Aplikasi Grey Wolf Optimizer dalam Pembelajaran Mesin Modern

Joko Siswantoro

DIREKTORAT PENERBITAN & PUBLIKASI ILMIAH UNIVERSITAS SURABAYA

Aplikasi *Grey Wolf Optimizer* dalam Pembelajaran Mesin Modern

Joko Siswantoro

Aplikasi *Grey Wolf Optimizer* dalam Pembelajaran Mesin Modern

Penulis:
Joko Siswantoro

Copy Editor:
Singgih Sugiarto

Desain Sampul dan Tata Letak:
Indah S. Rahayu

ISBN: 978-623-8038-80-0

Cetakan Pertama Juli 2026

Penerbit
Direktorat Penerbitan dan Publikasi Ilmiah
Universitas Surabaya
Anggota IKAPI & APPTI
Jl. Raya Kalirungkut Surabaya 60293
Telp. (62-31) 298-1344
E-mail: ppi@unit.ubaya.ac.id
Web: ppiubaya.ac.id

Kata Pengantar

Puji syukur penulis panjatkan ke hadirat Allah SWT atas rahmat dan karunia-Nya sehingga buku *Aplikasi Grey Wolf Optimizer dalam Pembelajaran Mesin Modern* ini dapat diselesaikan dengan baik. Buku ini disusun sebagai kontribusi akademik untuk memperkaya literatur mengenai pemanfaatan algoritma optimasi metaheuristik, khususnya *Grey Wolf Optimizer* (GWO), dalam mendukung berbagai proses penting pada pembelajaran mesin modern.

Pesatnya perkembangan pembelajaran mesin menuntut penggunaan metode optimasi yang mampu bekerja secara efektif pada ruang solusi yang kompleks, berdimensi tinggi, dan nonlinier. Dalam konteks tersebut, GWO menjadi salah satu algoritma yang menarik perhatian karena kesederhanaan strukturnya, kemampuannya menjaga keseimbangan antara eksplorasi dan eksploitasi, serta fleksibilitasnya dalam berbagai persoalan optimasi.

Buku ini bertujuan memberikan pemahaman yang utuh mengenai hubungan antara optimasi metaheuristik dan pembelajaran mesin, dengan menempatkan GWO sebagai fokus utama pembahasan. Pembaca diajak memahami landasan pembelajaran mesin, konsep dasar metaheuristik, prinsip kerja GWO, berbagai pengembangannya, serta penerapannya pada optimasi *hyperparameter*, pemilihan fitur, dan pelatihan model pembelajaran mesin.

Penulis menyadari bahwa buku ini masih memiliki keterbatasan. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan. Penulis berharap buku ini dapat memberikan manfaat akademik, memperluas wawasan pembaca, dan menjadi

rujukan dalam memahami pemanfaatan GWO dalam pembelajaran mesin modern.

Surabaya, Mei 2026

Penulis

Daftar Isi

Daftar Algoritma

Daftar Gambar

Daftar Tabel

01

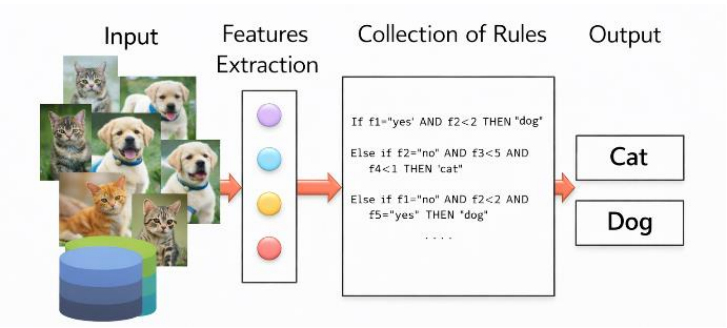
Mengenal Pembelajaran Mesin

1.1 Definis Pembelajaran Mesin

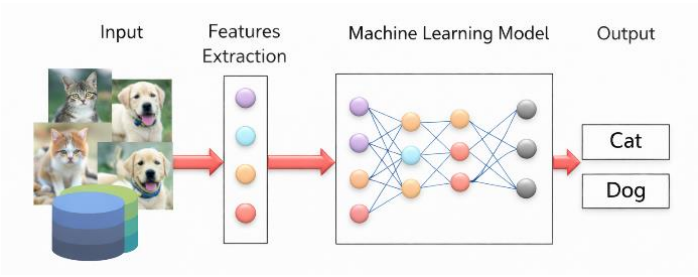
Pembelajaran mesin (*mechine learning*) adalah cabang dari kecerdasan artifisial yang bertujuan untuk membuat mesin/komputer mampu belajar tanpa harus diprogram secara eksplisit (Samuel, 1959). Maksudnya, komputer tidak harus diberikan satu per satu aturan "jika begini maka begitu" seperti pada kecerdasan artifisial tradisional, tetapi komputer belajar mengenali pola dari contoh yang diberikan, seperti diilustrasikan pada Gambar 1.1. Cara belajar ini mirip anak kecil belajar membedakan kucing dan anjing. Ketika orang tua menunjukkan beberapa contoh gambar kucing dan anjing sambil memberi tahu "ini kucing, ini anjing", anak mulai menangkap ciri-ciri umum, misalnya bentuk telinga, moncong, tekstur bulu, bukan menghafal satu gambar tertentu.

Dalam pembelajaran mesin, untuk membuat komputer bisa mengenali kucing dan anjing. Komputer diberi banyak contoh gambar yang sudah dilabeli "kucing" atau "anjing" sebagai data latih, kemudian algoritma menyesuaikan parameter model selama proses pelatihan (*training*) agar prediksinya makin benar. Ukuran belajarnya adalah kinerja yang membaik seiring pengalaman/data bertambah.

Tujuan pentingnya bukan sekadar benar pada gambar yang pernah dilihat, tetapi mampu mengenali kucing atau anjing lain yang baru, misalnya ras berbeda, sudut kamera berbeda, atau pencahayaan berbeda. Kemampuan tetap bisa mengenali pada contoh baru ini disebut generalisasi, dan menjadi konsep inti dalam pembelajaran mesin modern. Sehingga pembelajaran mesin dapat dipandang sebagai suatu proses membangun model yang kinerjanya meningkat melalui pengalaman (data), dengan tujuan utama mencapai generalisasi yang baik pada data baru (Tom M. Mitchell, 1997).



(a)



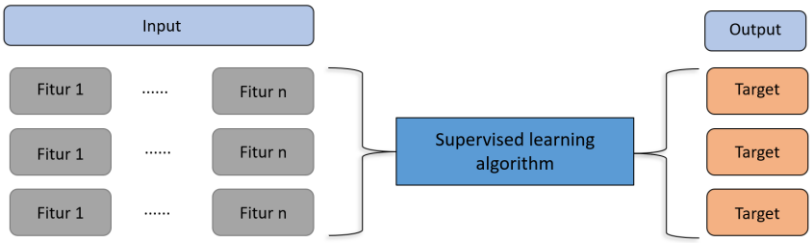
(b)

Gambar 1.1. Pendekatan: (a) kecerdasan artifisial tradisional (b) pembelajaran mesin.

1.2 Jenis-jenis Pembelajaran Mesin

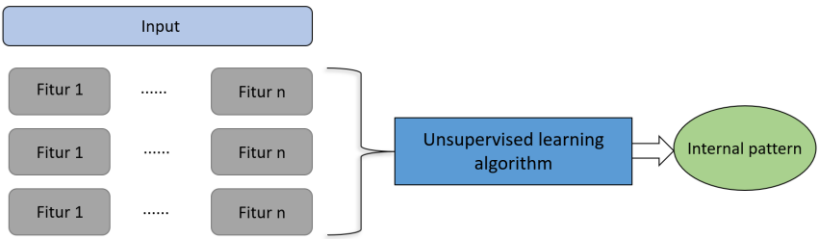
Secara umum, pembelajaran mesin dapat dibedakan menjadi empat kelompok utama, yaitu *supervised learning*, *unsupervised learning*, *semi-supervised learning*, dan *reinforcement learning*. Di luar empat kelompok utama tersebut, perkembangan mutakhir juga melahirkan pendekatan seperti *self-supervised learning*, *contrastive learning*, dan *generative learning*. Pembagian ini didasarkan pada jenis data yang tersedia, bentuk sinyal pembelajaran yang digunakan, serta tujuan yang ingin dicapai oleh model. Literatur dasar pembelajaran mesin umumnya menempatkan *supervised* dan *unsupervised learning* sebagai fondasi utama, sementara *semi-supervised* dan *reinforcement learning* berkembang sebagai kerangka yang menangani kondisi pembelajaran yang lebih khusus (Chapelle et al., 2006; Trevor Hastie et al., 2009; Sutton and Barto, 2018).

Supervised learning adalah jenis pembelajaran mesin yang menggunakan data berlabel, yaitu data yang setiap sampelnya sudah memiliki pasangan input dan target output, seperti diilustrasikan pada Gambar 1.2. Tujuan utamanya adalah mempelajari pemetaan dari input ke output sehingga model dapat melakukan prediksi pada data baru. Dalam konteks ini, model belajar dari contoh-contoh yang sudah diketahui jawabannya. Pendekatan ini umum digunakan pada tugas klasifikasi dan regresi. Contoh model yang termasuk *supervised learning* antara lain *linear regression, logistic regression, decision tree, random forest, support vector machine* (SVM), dan *artificial neural network* (ANN) untuk klasifikasi atau prediksi nilai (Bishop, 2010; Goodfellow et al., 2016; Trevor Hastie et al., 2009).



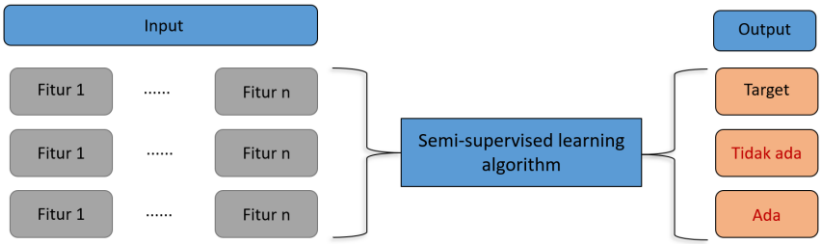
Gambar 1.2. Ilustrasi pembelajaran mesin jenis *supervised learning*

Unsupervised learning digunakan ketika data yang hanya mempunyai input tanpa target output, seperti diilustrasikan pada Gambar 1.3. Tujuan pembelajaran bukan memetakan input ke jawaban yang sudah diketahui, melainkan menemukan struktur, pola, hubungan, atau pengelompokan yang tersembunyi di dalam data. Pendekatan ini banyak digunakan untuk *clustering*, reduksi dimensi, dan eksplorasi struktur data. Contoh model yang termasuk *unsupervised learning* adalah *K-Means, hierarchical clustering, Gaussian mixture model*, dan *principal component analysis* (PCA). Dalam praktiknya, *unsupervised learning* berguna ketika peneliti ingin memahami karakteristik data sebelum masuk ke tahap prediksi atau ketika label memang belum tersedia (Trevor Hastie et al., 2009).



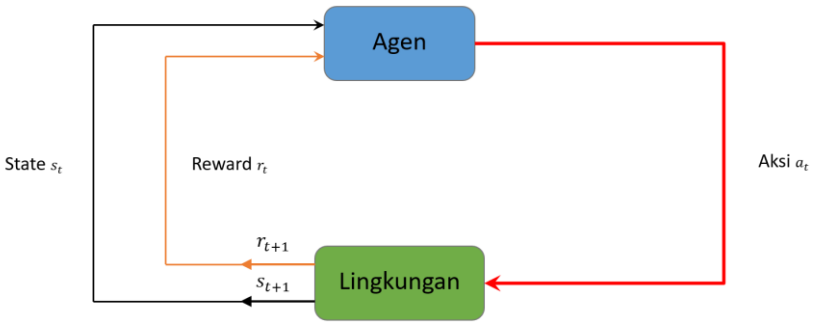
Gambar 1.3. Ilustrasi pembelajaran mesin jenis *unsupervised learning*

Semi-supervised learning berada di antara *supervised* dan *unsupervised learning*. Pendekatan ini digunakan ketika tersedia sedikit data berlabel dan jauh lebih banyak data tidak berlabel, seperti diilustrasikan pada Gambar 1.4. Ide utamanya adalah memanfaatkan informasi dari kedua jenis data tersebut agar model tetap dapat belajar dengan baik meskipun label terbatas. Pendekatan seperti ini sangat relevan pada banyak kasus nyata karena proses pelabelan data sering mahal, lambat, atau membutuhkan *human expert*. Contoh metode atau model yang umum dikaitkan dengan *semi-supervised learning* antara lain *self-training, co-training, graph-based label propagation, pseudo-labeling,* serta metode modern seperti *MixMatch* dan *FixMatch* (Chapelle et al., 2006; Zhu, 2010).



Gambar 1.4. Ilustrasi pembelajaran mesin jenis *semi-supervised learning*

Reinforcement learning berbeda dari tiga jenis sebelumnya karena pembelajaran berlangsung melalui interaksi dengan lingkungan. Dalam pendekatan ini, sebuah agen memilih aksi, menerima umpan balik dalam bentuk *reward*, lalu menyesuaikan kebijakannya untuk memaksimalkan akumulasi *reward* dalam jangka panjang, seperti diilustrasikan pada Gambar 1.5. Fokus utamanya bukan pada label target untuk setiap sampel, tetapi pada pengambilan keputusan bertahap. Contoh model atau algoritma yang terkenal dalam *reinforcement learning* adalah Q-*learning*, *State–Action–Reward–State–Action* (SARSA), *Deep* Q-*Network* (DQN), dan *Proximal Policy Optimization* (PPO). Pendekatan ini banyak digunakan pada *game*, robotika, kontrol, *autonomous vehicles*, dan sistem keputusan sekuensial lainnya (Mnih et al., 2015; Schulman et al., 2017; Sutton and Barto, 2018; Watkins and Dayan, 1992).



Gambar 1.5. Ilustrasi pembelajaran mesin jenis *reinforcement learning*

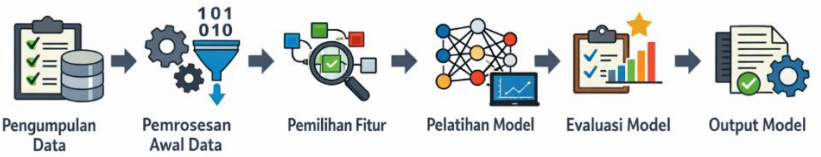
Seiring perkembangan bidang pembelajaran mesin, muncul pendekatan baru yang sering disebut sebagai perluasan atau evolusi dari kerangka pembelajaran mesin klasik. Salah satunya adalah *self-supervised learning*, yaitu pendekatan yang membangun target pembelajaran dari data itu sendiri tanpa memerlukan label manual dalam jumlah besar. Dalam domain bahasa, contoh yang sangat terkenal adalah *Bidirectional Encoder Representations from Transformers* (BERT), yang dilatih dengan tugas pretraining pada teks tak berlabel. Pendekatan ini menjadi penting karena mampu mempelajari representasi yang berguna sebelum model disesuaikan pada tugas spesifik (Devlin et al., 2019; Gui et al., 2024).

Salah satu cabang yang sangat menonjol dalam *self-supervised learning* adalah *contrastive learning*. Gagasan utamanya adalah mendorong representasi dari pasangan data yang saling berkaitan menjadi dekat, dan representasi dari pasangan yang tidak berkaitan menjadi jauh. Pendekatan ini sangat populer dalam pembelajaran representasi, khususnya pada visi komputer. Contoh model yang sangat dikenal adalah *Simple Framework for Contrastive Learning of Visual* **(**SimCLR). SimCLR memungkinkan model belajar fitur visual yang bermakna tanpa memerlukan label manual dalam jumlah besar (Chen et al., 2020).

Pendekatan lain yang juga berkembang pesat adalah *generative learning*. Pada pendekatan ini, model tidak hanya belajar memetakan input ke output, tetapi juga mempelajari distribusi data sehingga mampu menghasilkan sampel baru yang menyerupai data asli. Model generatif sangat penting dalam sintesis citra, pemodelan data, representasi laten, dan simulasi. Contoh model yang terkenal adalah *Variational Autoencoder* (VAE) dan *Generative Adversarial Network* (GAN). VAE mempelajari representasi probabilistik laten, sedangkan GAN menggunakan dua komponen yang saling bersaing, yaitu generator dan discriminator, untuk menghasilkan data sintetis yang semakin realistis (Goodfellow et al., 2014; Kingma and Welling, 2014).

1.3 Pelathan Model Pembelajaran Mesin

Secara umum, proses pelatihan model pembelajaran mesin terdiri atas enam tahapan utama, yaitu pengumpulan data, pemrosesan awal data, pemilihan fitur, pelatihan model, evaluasi model, dan outputnya berupa model pembelajaran mesin, seperti dapat dilihat pada Gambar 1.6. Susunan tahapan ini sejalan dengan kerangka proses dalam *data mining* dan *predictive modeling* yang menempatkan pemahaman data, persiapan data, pemodelan, evaluasi, dan penerapan hasil sebagai bagian yang saling berkaitan dalam satu alur kerja.



Gambar 1.6. Tahapan proses pelatihan model pembelajaran mesin.

Tahap pertama adalah pengumpulan data. Pada tahap ini, data diperoleh dari sumber yang relevan dengan masalah yang ingin diselesaikan, misalnya basis data, log sistem, dokumen, citra, sensor, atau sumber publik lainnya. Tahap ini tidak hanya menekankan jumlah data, tetapi juga relevansi, keterwakilan, dan kualitas data terhadap kondisi nyata yang ingin dipelajari. Data yang tidak representatif akan membatasi kemampuan model dalam mengenali pola yang benar dan menghasilkan prediksi yang dapat diandalkan (Wirth and Hipp, 2000).

Tahap kedua adalah pemrosesan awal data. Data yang baru dikumpulkan umumnya masih berada dalam bentuk mentah dan belum siap langsung digunakan untuk pelatihan model. Dalam praktiknya, data dapat mengandung *missing value*, perbedaan skala, format yang tidak konsisten, *noise*, atau variabel kategorikal yang belum diubah ke bentuk numerik. Oleh karena itu, tahap ini biasanya mencakup pembersihan data, transformasi, standardisasi atau normalisasi, serta

pembagian data ke dalam data latih dan data uji. Tahap ini penting karena kualitas representasi data sangat memengaruhi kualitas model yang akan dibangun (Bishop, 2006; Trevor Hastie et al., 2009; Kuhn and Johnson, 2013).

Tahap ketiga adalah pemilihan fitur. Tidak semua fitur atau atribut dalam data memiliki kontribusi yang sama terhadap performa model. Banyaknya fitur yang tersedia dalam dataset belum tentu menjamin kinerja model menjadi lebih baik. Sebagian fitur dapat sangat informatif, sebagian lainnya dapat bersifat redundan, tidak relevan, atau saling berkorelasi tinggi, sehingga justru menambah *noise*, meningkatkan kompleksitas model, dan mendorong terjadinya *overfitting*. *Overfitting* adalah kondisi ketika model belajar terlalu mengikuti detail, noise, atau pola kebetulan pada data latih, sehingga performanya tampak sangat baik pada data pelatihan tetapi menurun saat dihadapkan pada data baru yang belum pernah dilihat. Dengan kata lain, model menjadi terlalu menyesuaikan diri terhadap data latih dan kehilangan kemampuan generalisasi. Risiko ini cenderung meningkat ketika jumlah fitur terlalu banyak, terutama jika sebagian fitur tidak relevan atau redundan. Oleh karena itu, pemilihan fitur menjadi tahap penting untuk mempertahankan fitur yang paling bermakna, menyederhanakan model, mengurangi biaya komputasi, dan meningkatkan kemampuan generalisasi (Berrar and Dubitzky, 2013; Siswantoro et al., 2020b).

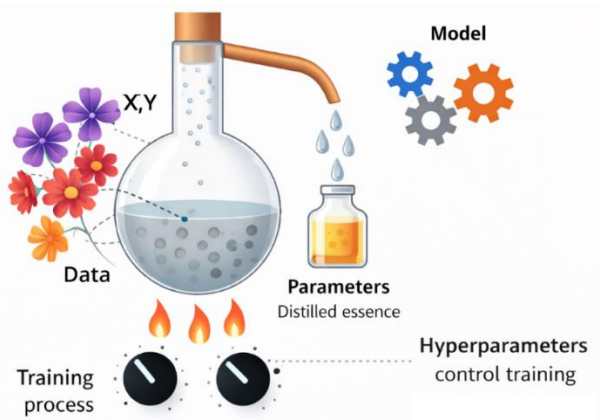
Secara umum, metode pemilihan fitur diklasifikasikan ke dalam tiga kelompok utama, yaitu *filter*, *wrapper*, dan *embedded* (Saeys et al., 2007). Metode *filter* menilai relevansi fitur berdasarkan karakteristik statistik data, misalnya korelasi, *mutual information*, atau skor uji tertentu, tanpa melibatkan model prediksi tertentu. Metode ini umumnya cepat dan efisien untuk data berdimensi tinggi, tetapi dapat mengabaikan interaksi antarfitur. Metode *wrapper* mengevaluasi subset fitur dengan melatih dan menguji model secara langsung, sehingga biasanya mampu menghasilkan subset yang lebih sesuai untuk model tertentu, walaupun biaya komputasinya lebih tinggi. Sementara itu, metode *embedded* melakukan seleksi fitur

sebagai bagian dari proses pelatihan model, misalnya melalui regularisasi atau mekanisme pemilihan internal model. Dari sudut pandang formulasi, pemilihan fitur juga dapat dipandang sebagai masalah optimasi, karena tujuannya adalah mencari subset fitur yang memaksimalkan atau meminimalkan fungsi objektif tertentu, misalnya akurasi klasifikasi, galat prediksi, atau kombinasi performa dan ukuran subset fitur (Wijaya and Siswantoro, 2025).

Selain melalui pemilihan fitur, pengurangan dimensi fitur yang diinputkan ke model pembelajaran mesin juga dapat dilakukan dengan mentransformasi ruang fitur asal ke ruang fitur baru yang dimensinya lebih rendah. Salah satu metode yang umum digunakan untuk pengurangan dimensi fitur adalah *Principal Component Analysis* (PCA). Berbeda dari pemilihan fitur yang mempertahankan sebagian fitur asli, PCA membentuk sekumpulan variabel baru yang disebut komponen utama yang merupakan kombinasi linear dari fitur-fitur awal. Komponen-komponen ini saling ortogonal dan disusun sedemikian rupa sehingga komponen pertama menangkap variasi terbesar dalam data, diikuti oleh komponen berikutnya dengan variasi yang semakin kecil. Dengan cara ini, PCA dapat mereduksi dimensi data sambil tetap mempertahankan sebagian besar informasi penting. Namun, karena PCA menghasilkan fitur baru hasil transformasi, interpretasinya biasanya tidak sesederhana pemilihan fitur yang tetap menggunakan atribut asli. Oleh sebab itu, dalam praktik pembelajaran mesin, pemilihan fitur dan PCA sering dipandang sebagai dua strategi yang berbeda tetapi sama-sama penting untuk menangani data berdimensi tinggi (Giordani, 2018).

Tahap keempat adalah pelatihan model. Pada tahap ini, data latih digunakan untuk membangun hubungan antara input dan output. Dalam *supervised learning*, proses ini bertujuan untuk menemukan fungsi atau pola yang dapat digunakan untuk memprediksi target pada data baru. Terdapat dua komponen penting yang perlu dibedakan dalam pelatihan model pembelajaran mesin, yaitu parameter dan *hyperparameter*. Proses pelatihan dapat dianalogikan seperti penyulingan minyak wangi dari bunga,

seperti diilustrasikan pada Gambar 1.7. Dalam analogi ini, data berperan sebagai bunga yang menjadi bahan utama, sementara proses pelatihan merupakan proses penyulingan yang bertujuan mengekstraksi esensi terbaik dari bahan tersebut. Parameter dapat dipandang sebagai hasil penyulingan yang diperoleh dari bunga melalui proses tersebut. Nilai parameter, seperti bobot dan bias pada jaringan saraf atau koefisien pada model regresi, diperbarui secara bertahap selama proses pelatihan berdasarkan data yang tersedia, sehingga model dapat menangkap pola yang terkandung di dalam data (Bishop, 2006).



Gambar 1.7. Ilustrasi proses pelatihan model pembelajaran mesin.

Sebaliknya, *hyperparameter*, seperti jumlah neuron pada jaringan saraf atau koefisien regularisasi pada *ridge regression*, merupakan nilai yang ditentukan oleh pengguna sebelum proses pelatihan dimulai. Dalam analogi penyulingan minyak wangi, *hyperparameter* dapat diibaratkan sebagai besarnya api, suhu pemanasan, tekanan, atau lamanya waktu penyulingan yang telah ditetapkan sebelumnya. *Hyperparameter* tidak dihasilkan dari data, tetapi ditentukan diawal untuk mengendalikan bagaimana proses ekstraksi berlangsung. Pengaturan ini sangat memengaruhi kualitas hasil akhir, karena menentukan dinamika proses belajar, stabilitas

pelatihan, serta kemampuan model untuk menghasilkan representasi yang mampu digeneralisasikan pada data baru. Dengan demikian, keberhasilan pembelajaran mesin tidak hanya bergantung pada kualitas data sebagai bahan baku, tetapi juga pada keseimbangan antara hasil ekstraksi (parameter) dan pengaturan proses (*hyperparameter*) yang tepat (Murphy, 2012).

Algoritma optimasi memiliki peran sentral dalam kedua aspek tersebut. Untuk parameter model, algoritma optimasi berfungsi sebagai mekanisme yang secara sistematis memperbarui nilai parameter agar *loss fuction* semakin kecil. Dalam proses pelatihan, model menerima data sebagai input, menghasilkan prediksi, kemudian membandingkan hasil prediksi tersebut dengan jawaban yang benar menggunakan sebuah fungsi yang disebut *loss function*. Fungsi ini digunakan sebagai indikator seberapa jauh hasil prediksi menyimpang dari target yang diharapkan. Secara sederhana, *loss function* memberi tahu model apakah hasil prediksi sudah dekat dengan jawaban yang benar atau masih jauh. Nilai *loss* yang besar menunjukkan bahwa prediksi model masih buruk, sedangkan nilai loss yang kecil menunjukkan bahwa prediksi model semakin baik (Bishop, 2006).

Untuk masalah regresi, salah satu *loss function* yang paling umum digunakan adalah *mean squared error* (MSE). MSE menghitung rata-rata kuadrat selisih antara nilai target aktual dan nilai prediksi model, seperti pada persamaan (1.1),

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \qquad (1.1)$$

dengan N menyatakan jumlah sampel, y_i menyatakan nilai target sebenarnya pada sampel ke-i, dan $\hat{y}_i$ menyatakan nilai prediksi model pada sampel ke-i. Penggunaan kuadrat pada selisih membuat kesalahan yang besar mendapatkan penalti lebih kuat, sehingga model

terdorong untuk mengurangi prediksi yang sangat meleset. *Squared loss* seperti ini merupakan pilihan yang sangat umum dalam regresi dan dibahas secara klasik dalam literatur *pattern recognition* dan *statistical learning* (Tibshirani Hastie et al., 2009).

Sementara itu, untuk masalah klasifikasi, *loss function* yang sangat umum digunakan adalah *cross entropy*. Fungsi ini mengukur perbedaan antara distribusi probabilitas target dan distribusi probabilitas yang diprediksi model, seperti dinyatakan pada persamaan (1.2),

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik}\log(\hat{p}_{ik}) \tag{1.2}$$

dengan N menyatakan jumlah sampel, K menyatakan jumlah kelas, y_{ik} menyatakan indikator target untuk sampel ke-i pada kelas ke-k (umumnya bernilai 1 jika sampel termasuk kelas tersebut dan 0 jika tidak), serta $\hat{p}_{ik}$ menyatakan probabilitas prediksi model bahwa sampel ke-i termasuk ke kelas ke-k. Pada kasus klasifikasi biner, *cross entropy* disederhanakan bentuknya seperti pada persamaan (1.3).

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i\log(\hat{p}_i) + (1-y_i)\log(1-\hat{p}_i)] \tag{1.3}$$

Cross entropy banyak digunakan karena sesuai untuk keluaran probabilistik, khususnya ketika model klasifikasi menghasilkan probabilitas melalui fungsi seperti *sigmoid* atau *softmax*. Dalam literatur *deep learning*, *cross entropy* dijelaskan sebagai fungsi objektif yang mendorong distribusi prediksi model agar semakin mendekati distribusi target pada data pelatihan (Goodfellow et al., 2016; Mao et al., 2023).

Proses pelatihan model pembelajaran mesin, khususnya model ANN dan *deep learning*, berlangsung secara iteratif dengan tujuan meminimalkan *loss function*, sehingga model secara bertahap meningkatkan kemampuannya dalam

menangkap pola data. Kerangka ini dikenal sebagai *empirical risk minimization*, yaitu upaya menemukan konfigurasi model yang memberikan kinerja terbaik berdasarkan pengalaman data (Bishop, 2006; Tibshirani Hastie et al., 2009). Salah satu pendekatan yang umum digunakan dalam pelatihan model pembelajaran mesin dalah metode optimasi berbasis gradien, seperti *gradient descent* pada persamaan (1.4). Pada persamaan tersebut, $\mathbf{W}$ adalah parameter model, η adalah *learning rate*, $\nabla Loss\left(\mathbf{W}^{(k)}\right)$ adalah gradien dari *loss fuction*, dan $k = 1,2,\dots,n$ adalah indeks iterasi. Pendekatan ini memanfaatkan informasi gradien untuk menentukan arah perubahan parameter yang paling efektif, sebagaimana diimplementasikan dalam metode *backpropagation* pada jaringan saraf (Santra et al., 2021). Detai algoritma pelatihan pembelajaran mesin berbasis *gradient descent* disajikan pada Algoritma 1.1.

$$\mathbf{W}^{(k+1)} = \mathbf{W}^{(k)} - \eta \nabla Loss\left(\mathbf{W}^{(k)}\right) \tag{1.4}$$

Gambar 1.8 mengilustrasikan proses pelatihan model pembelajaran mesin berbasis *gradient descent*. Proses ini berjalan dengan menuruni permukaan *loss function* secara bertahap menuju titik dengan nilai *loss* yang lebih rendah. Ilustrasi ini dapat dianalogikan seperti melepaskan sebuah bola di lereng bukit untuk mencari lembah. Bola akan bergerak dari posisi yang lebih tinggi menuju daerah yang lebih rendah. Dalam konteks pembelajaran mesin, arah pergerakan parameter ditentukan oleh arah negatif gradien, karena gradien menunjukkan arah kenaikan tercepat dari nilai fungsi objektif. Oleh sebab itu, agar nilai *loss* menurun, parameter model diperbarui ke arah yang berlawanan dengan gradien. Proses ini dilakukan berulang pada setiap iterasi hingga model semakin mendekati titik minimum. Besar kecilnya langkah perpindahan ditentukan oleh *learning rate*; jika nilainya terlalu besar, proses optimasi dapat melewati titik minimum dan menjadi tidak stabil, sedangkan jika terlalu kecil, konvergensi berlangsung lambat.

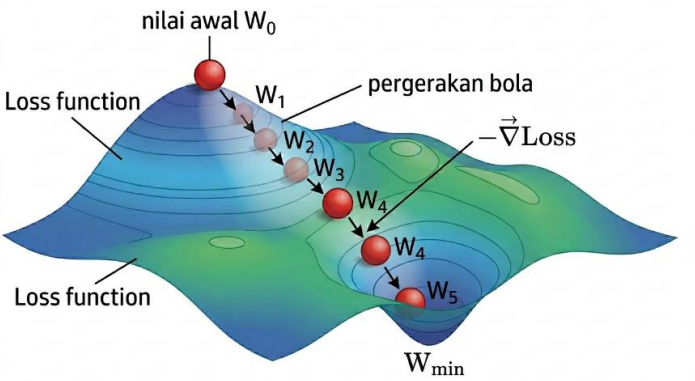
Algoritma 1.1. Pelatihan model pembelajaran mesin berbasis *gradient descent*

Input:
Data latih X, target y, model pembelajaran mesin, *learning rate* η, jumlah iterasi maksimum n, dan *loss function* $\text{Loss}(W)$.

Output:
Model terlatih dengan parameter akhir W^* dan nilai *loss function* akhir.

Prosedur:

1. Bangkitkan parameter awal model secara acak untuk memperoleh $W^{(0)}$.
2. Tetapkan indeks iterasi $k \leftarrow 0$.
3. Selama $k < n$ dan kriteria berhenti belum terpenuhi, lakukan langkah-langkah berikut:
   3.1. Hitung prediksi model menggunakan parameter saat ini, $W^{(k)}$, terhadap data latih X.
   3.2. Hitung nilai *loss function* berdasarkan selisih antara hasil prediksi dan target y.
   3.3. Hitung gradien *loss function* terhadap parameter model, yaitu $\nabla\text{Loss}\left(W^{(k)}\right)$.
   3.4. Perbarui parameter model ke arah negatif gradien dengan persamaan (4)
   3.5. Jika nilai *loss function* sudah cukup kecil atau perubahan parameter sudah sangat kecil, hentikan iterasi.
   3.6. Perbarui indeks iterasi menjadi $k \leftarrow k + 1$.
4. Tetapkan parameter akhir sebagai $W^* = W^{(k)}$.
5. Kembalikan model terlatih beserta nilai *loss function* akhirnya.



Gambar 1.8. Ilustrasi pelatihan model pembelajaran mesin berbasis gradien.

Di lain pihak, optimasi *hyperparameter* berfokus pada pencarian konfigurasi pengaturan awal model yang menghasilkan model dengan kinerja terbaik. Karena ruang pencarian *hyperparameter* sering kali kompleks, berdimensi tinggi, dan gradien dari fungsi objektifnya tidak dapat dihitung secara eksak, pendekatan berbasis gradien sulit diinplementasikan dalam optimasi *hyperparameter*. Oleh karena itu, metode optimasi tingkat lanjut, seperti *grid search* (Larochelle et al., 2007), *random search* (Bergstra and Bengio, 2012), optimasi Bayesian (Snoek et al., 2012), maupun pendekatan metaheuristik (Siswantoro, 2024) sering digunakan untuk menemukan kombinasi *hyperparameter* yang optimal. Dengan demikian, proses pelatihan pembelajaran mesin dapat dipahami sebagai proses optimasi berlapis yang melibatkan penyesuaian parameter dan pemilihan *hyperparameter* serta pencarian subset fitur secara simultan.

Tahap kelima adalah evaluasi model. Setelah model selesai dilatih, langkah berikutnya adalah menilai seberapa baik model tersebut bekerja. Evaluasi model dilakukan untuk mengukur kemampuan model dalam menghasilkan prediksi yang akurat dan konsisten, terutama ketika dihadapkan pada data yang tidak digunakan selama proses pelatihan. Tahap ini

penting karena model yang tampak sangat baik pada data latih belum tentu memiliki kinerja yang sama pada data baru. Oleh sebab itu, evaluasi digunakan untuk menilai kemampuan generalisasi model, yaitu sejauh mana model mampu menerapkan pola yang telah dipelajari ke data yang belum pernah dilihat sebelumnya. Tahap evaluasi tidak hanya menunjukkan apakah model sudah baik, tetapi juga membantu menentukan apakah model perlu diperbaiki, disetel ulang, atau bahkan diganti dengan pendekatan lain sebelum digunakan lebih lanjut. (Sokolova and Lapalme, 2009).

Evaluasi model dilakukan menggunakan data uji atau melalui teknik validasi tertentu, kemudian kinerjanya diukur dengan metrik yang sesuai dengan jenis permasalahan. Pada tugas klasifikasi, metrik yang umum digunakan antara lain akurasi, presisi, recall, dan F1– *score*, yang masing-masing dihitung menggunakan rumus pada persamaan (1.5) – (1.8),

$$\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.5}$$

$$\text{Presisi} = \frac{TP}{TP + FP} \tag{1.6}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{1.7}$$

$$\text{F1} - \text{score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{1.8}$$

dengan TP menyatakan *true positive*, TN adalah *true negative*, FP adalah *false positive*, dan FN adalah *false negative*. Akurasi memberikan gambaran proporsi prediksi yang benar secara keseluruhan, sedangkan presisi, *recall*, dan F1– *score* lebih membantu ketika distribusi kelas tidak seimbang atau ketika jenis kesalahan tertentu perlu mendapat perhatian khusus.

Sedangkan pada tugas regresi sering digunakan MSE, seperti pada persamaan (1.1), untuk sebagai metrik evaluasi. Selain MSE, evaluasi model regresi juga sering menggunakan koefisien determinasi, R^2, yang dihitung menggunakan rumus pada persamaan (1.9),

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \tag{1.9}$$

dengan N menyatakan jumlah sampel, y_i menyatakan nilai target sebenarnya pada sampel ke-i, dan $\hat{y}_i$ menyatakan nilai prediksi model pada sampel ke-i, dan $\bar{y}$ sebagai rata-rata nilai aktual. Metrik ini mengukur proporsi variasi pada data target yang dapat dijelaskan oleh model. Nilai R^2 yang semakin mendekati 1 menunjukkan bahwa model mampu menjelaskan variasi data dengan lebih baik, sedangkan nilai yang rendah menunjukkan bahwa model belum mampu menangkap hubungan yang memadai antara prediktor dan target.

Tahap keenam adalah output model pembelajaran mesin. Setelah model melalui proses pelatihan dan evaluasi, hasil akhirnya berupa model yang telah mempelajari pola atau hubungan dari data. Pada tahap ini, model tidak lagi hanya dipandang sebagai hasil komputasi, tetapi sebagai representasi fungsional yang dapat digunakan untuk menyelesaikan tugas tertentu. Dalam praktiknya, output model dapat berupa label kelas pada masalah klasifikasi, nilai prediksi pada masalah regresi, skor probabilitas, peringkat rekomendasi, atau bentuk keputusan lain sesuai dengan tujuan pembelajaran. Dengan demikian, keluaran dari keseluruhan proses pembelajaran mesin bukan hanya angka performa hasil evaluasi, melainkan sebuah model yang siap digunakan untuk melakukan inferensi terhadap data baru. Konteks ini sejalan dengan pandangan bahwa sistem pembelajaran mesin di dunia nyata tidak berhenti pada tahap pelatihan, tetapi harus dirancang agar model dapat dioperasikan secara berkelanjutan dalam sistem produksi (Paleyes et al., 2022).

Pada saat implementasi, model yang telah dilatih kemudian disimpan ke dalam file atau artefak model agar dapat digunakan

kembali tanpa harus dilatih ulang setiap kali aplikasi dijalankan. Proses ini sering disebut sebagai *model persistence*, *serialization*, atau *model artifact management*. Setelah disimpan, file model tersebut kemudian dapat dimuat kembali oleh aplikasi pada saat proses inferensi, misalnya ketika aplikasi menerima data baru dari pengguna dan perlu menghasilkan prediksi secara langsung. Pendekatan ini penting karena memisahkan tahap pelatihan dari tahap penggunaan, sehingga sistem menjadi lebih efisien, lebih mudah dipelihara, dan lebih sesuai untuk penerapan nyata, baik pada layanan web, sistem enterprise, maupun perangkat edge (Schlegel and Sattler, 2023).

02

Algoritma Optimasi *Metaheuristik Grey Wolf Optimizer*

2.1 Algoritma Optimasi Metaheuristik

Metaheuristik adalah strategi tingkat-atas untuk menyelesaikan masalah optimasi yang sulit, seperti ruang solusinya sangat besar, banyak jebakan optimum lokal, dan metode eksak terlalu mahal secara komputasi. Secara umum, masalah optimasi dapat dipahami sebagai upaya mencari nilai variabel keputusan (*decision variables*) yang membuat suatu fungsi objektif mencapai nilai terbaik, yaitu minimum atau maksimum, sesuai tujuan yang ditetapkan. Berbeda dari heuristik biasa yang sering spesifik untuk satu masalah, metaheuristik bersifat umum dan menyediakan kerangka pencarian yang dapat dipakai lintas masalah dengan cara mengatur kombinasi mekanisme eksplorasi (menjelajah banyak kemungkinan) dan eksploitasi (memperbaiki solusi yang sudah bagus). Karena itu metaheuristik dipakai ketika tujuan praktisnya adalah memperoleh solusi yang "sangat baik" dalam waktu wajar, meskipun tanpa jaminan selalu mencapai optimum global (Blum and Roli, 2003).

Algoritma optimasi metaheuristik telah dimanfaatkan secara luas dalam beragam bidang aplikasi. Luasnya penggunaan ini

setidaknya didukung oleh empat keunggulan utama, yaitu sederhana untuk dipahami, keluwesan untuk berbagai masalah, tidak bergantung pada informasi turunan, dan mampu mengurangi risiko terjebak pada optimum lokal (Mirjalili et al., 2014). Ditinjau dari aspek kesederhanaan, banyak algoritma optimasi metaheuristik dikembangkan berdasarkan gagasan yang intuitif, seperti pola perilaku makhluk hidup, gejala fisika, atau mekanisme evolusi. Landasan yang relatif sederhana tersebut membuat algoritma-algoritma ini lebih mudah dikembangkan, diimplementasikan, disesuaikan, maupun dipadukan dengan metode lain. Selain itu, karakter yang tidak terlalu rumit juga memudahkan peneliti dari berbagai latar belakang untuk mempelajari dan menerapkannya, sehingga pemanfaatannya tidak hanya berkembang di ilmu komputer, tetapi juga di banyak bidang lain yang memerlukan optimasi.

Dari sisi keluwesan, algoritma optimasi metaheuristik memiliki kemampuan untuk digunakan pada beragam jenis persoalan tanpa harus mengubah kerangka dasarnya secara signifikan. Dalam banyak penerapan, masalah diperlakukan sebagai suatu kotak hitam, dan pengguna hanya perlu menentukan variabel keputusan, fungsi objektif, dan fungsi kendala dari masalah optimasi yang akan diselesaikan. Selama suatu persoalan dapat dinyatakan dalam bentuk tersebut, pendekatan metaheuristik yang sama pada prinsipnya tetap dapat digunakan untuk menelusuri solusi yang baik. Karakter ini menjadikan metaheuristik sangat luwes dan mudah diterapkan ke berbagai bidang.

Keunggulan berikutnya adalah tidak bergantung pada informasi turunan. Sebagian besar algoritma optimasi metaheuristik tidak memerlukan informasi gradien (turunan parsial) dari fungsi objektif, berbeda dengan pendekatan berbasis gradien yang mensyaratkan fungsi objektif memiliki bentuk matematis yang jelas agar turunan parsialnya dapat dihitung. Algoritma optimasi metaheuristik bekerja secara stokastik dengan memulai pencarian dari solusi awal, kemudian memperbaikinya sedikit demi sedikit melalui aturan pembaruan tertentu. Karakter semacam ini sangat

bermanfaat pada permasalahan nyata yang fungsi objektifnya rumit, tidak mulus, bersifat implisit, mahal untuk dievaluasi, atau bahkan tidak memungkinkan dihitung turunannya. Karena tidak bergantung pada informasi gradien, metaheuristik menjadi lebih praktis untuk diterapkan pada masalah-masalah yang sulit dijangkau oleh optimasi konvensional.

Algoritma optimasi metaheuristik juga dikenal memiliki kemampuan yang baik dalam menghindari jebakan optimum lokal. Pada banyak persoalan optimasi dunia nyata, ruang solusi sering kali sangat kompleks dan dipenuhi banyak titik optimum lokal, sehingga algoritma dapat berhenti pada solusi yang tampak baik tetapi belum tentu merupakan solusi terbaik secara global. Melalui mekanisme pencarian yang bersifat stokastik dan didukung oleh keseimbangan antara eksplorasi serta eksploitasi, metaheuristik mampu menjelajahi ruang pencarian secara lebih luas dan tidak mudah berhenti di satu wilayah solusi saja. Oleh sebab itu, metaheuristik kerap dipandang sebagai pendekatan yang lebih tangguh untuk menangani persoalan optimasi yang kompleks dan kaya jebakan optimum lokal.

Dalam skala umum, penerapannya dapat ditemukan pada bidang *water management*, *geotechnical engineering*, *transport engineering*, *scheduling*, *renewable energy*, serta pembelajaran mesin. Dalam *water management*, algoritma optimasi metaheuristik telah diterapkan pada berbagai persoalan seperti inversi parameter model air tanah, optimasi jaringan distribusi air, dan operasi reservoir. Bidang ini umumnya melibatkan model nonlinier, kendala operasional yang banyak, serta ketidakpastian yang tinggi, sehingga optimasi konvensional sering sulit diterapkan secara langsung. Metaheuristik membantu mencari konfigurasi parameter dan kebijakan operasi yang lebih baik, misalnya untuk meningkatkan akurasi model air tanah, menekan kehilangan air dalam jaringan distribusi, atau menyesuaikan operasi waduk terhadap perubahan permintaan dan kondisi iklim (Chen and Dai, 2024; Chong et al., 2024; Djebedjian et al., 2021).

Pada *geotechnical engineering*, algoritma optimasi metaheuristik telah diterapkan untuk menyelesaikan berbagai persoalan seperti analisis kestabilan lereng**,** prediksi perpindahan longsor, dan optimasi parameter pada model geoteknik lainnya. Tinjauan di bidang rekayasa sipil menunjukkan bahwa metaheuristik sangat efektif dalam menangani masalah geoteknik yang rumit karena mampu mencari solusi mendekati optimum tanpa memerlukan turunan fungsi objektif. Penggunaan pada prediksi pergeseran longsor dan analisis kestabilan lereng memperlihatkan bahwa metaheuristik bukan hanya berguna pada optimasi abstrak, tetapi juga pada persoalan keselamatan dan perencanaan infrastruktur nyata (Ghaemifard and Ghannadiasl, 2024; Ma et al., 2022; Mishra et al., 2019).

Dalam *transport engineering*, algoritma optimasi metaheuristik banyak digunakan untuk masalah *vehicle routing*, *traveling salesman problem*, dan optimasi distribusi logistik yang lebih kompleks. Tinjauan taksonomis pada *vehicle routing problem* menunjukkan bahwa metaheuristik merupakan kelompok metode yang sangat dominan dalam menyelesaikan berbagai varian persoalan rute kendaraan. Peran ini semakin penting dalam konteks sistem distribusi modern, termasuk distribusi bertingkat (*multi-echelon distribution*), karena masalah transportasi sering melibatkan kombinasi tujuan, kendala, dan skala pencarian yang sulit ditangani metode eksak secara efisien (Elshaer and Awad, 2020; Nielsen et al., 2024).

Pada *scheduling*, algoritma optimasi metaheuristik digunakan untuk *project scheduling***,** *construction scheduling*, dan *task scheduling* pada lingkungan komputasi awan. Dalam manajemen proyek dan konstruksi, metaheuristik dipakai untuk mengoptimalkan waktu, biaya, dan alokasi sumber daya. Sementara itu, dalam komputasi awan, metaheuristik telah menjadi pendekatan penting untuk menjadwalkan tugas secara efisien pada sumber daya yang terbatas dan dinamis. Luasnya aplikasi pada penjadwalan menunjukkan bahwa metaheuristik sangat berguna ketika keputusan

harus dibuat di bawah banyak kendala dan kombinasi solusi yang sangat besar (Houssein et al., 2021; Liao et al., 2011).

Dalam bidang *renewable energy*, algoritma optimasi metaheuristik telah digunakan secara luas untuk menyelesaikan berbagai persoalan desain dan operasi yang kompleks. Aplikasinya mencakup *optimal sizing* pada *hybrid renewable energy systems* seperti kombinasi surya–angin–baterai, perencanaan dan integrasi sumber energi terbarukan pada sistem tenaga, optimasi operasi dan efisiensi pada jaringan energi, serta identifikasi parameter dan pengaturan model pada perangkat energi terbarukan. Selain itu, algoritma optimasi metaheuristik juga digunakan pada ranah *bioenergy*, misalnya untuk optimasi proses produksi biodiesel, termasuk optimasi campuran bahan baku dan kondisi proses transesterifikasi agar rendemen dan kualitas bahan bakar meningkat (Bouaouda and Sayouti, 2022; Gusain et al., 2023; Nassef et al., 2023; Ong et al., 2019; Rezk et al., 2024; Sebayang et al., 2023; Silitonga et al., 2020).

Algoritma optimasi metaheuristik juga banyak diaplikasikan dalam pembelajaran mesin. Hal ini karena masalah optimasi dalam pembelajaran mesin memiliki ruang pencarian yang besar dan fungsi objektifnya tidak mudah dioptimalkan secara analitik. Dalam bidang ini, algoritma optimasi metaheuristik banyak dimanfaatkan untuk seleksi fitur, optimasi *hyperparameter*, *data augmentation*, melatih model *artificial neural netwok*, dan optimasi model *clustering* (Kumar et al., 2017; Siswantoro, 2024; Siswantoro et al., 2026; Sunaryono et al., 2025, 2023, 2022; Zhang et al., 2014).

2.2 Jenis Algoritma Optimasi Metaheuristik

Secara umum, metaheuristik dapat dibedakan menjadi dua kelompok besar, yaitu *single-solution based* dan *population-based* (Ahmed et al., 2021; Van Thieu and Mirjalili, 2023). Pada kelompok pertama, proses pencarian berpusat pada satu solusi yang diperbaiki terus-menerus dari iterasi ke iterasi. Pendekatan ini relatif sederhana,

tetapi lebih rentan terhadap bias inisialisasi dan lebih sulit menjaga keseimbangan eksplorasi serta eksploitasi karena hanya mengandalkan satu titik pencarian. Salah satu contoh yang paling dikenal adalah *Simulated Annealing* (Van Laarhoven and Aarts, 1987). Sebaliknya, pada kelompok *population-based*, algoritma menggunakan sekumpulan solusi sekaligus pada setiap iterasi untuk mencari solusi optimum. Setiap solusi bertindak sebagai agen pencarian yang dapat saling bertukar informasi atau dipengaruhi oleh solusi terbaik yang ada, sehingga pencarian menjadi lebih kaya dan lebih tangguh dalam menjelajahi ruang solusi. Pendekatan berbasis populasi inilah yang kemudian menjadi bentuk metaheuristik yang paling banyak dikembangkan dalam literatur modern.

Dalam perkembangannya, algoritma optimasi metaheuristik berbasis populasi dapat dikatgorikan lebih rinci berdasarkan sumber inspirasi yang melandasi mekanisme pencariannya (Van Thieu and Mirjalili, 2023). Kategorisasi ini penting karena cara sebuah algoritma membangkitkan solusi baru, mempertukarkan informasi antarsolusi, dan menyeimbangkan eksplorasi serta eksploitasi sangat dipengaruhi oleh metafora dasar yang digunakan. Meskipun semua kategori bertujuan mencari solusi terbaik, masing-masing memiliki sudut pandang yang berbeda dalam merancang aturan pergerakan solusi di ruang pencarian. Kategori tersebut meliputi:

1) *Evolutionary-based optimozation*
2) *Swarm-based optimozation*
3) *Physics-based optimozation*
4) *Human-based optimozation*
5) *Biology-based optimozation*
6) *System-based optimozation*
7) *Math-based optimozation*
8) *Music-based optimozation*.

Evolutionary-based optimozation merupakan salah satu kategori algoritma optimasi metaheuristik yang paling awal dan paling dikenal. Kelompok ini dibangun dari gagasan evolusi alam, terutama prinsip seleksi alam, reproduksi, pewarisan sifat, dan perubahan populasi dari satu generasi ke generasi berikutnya. Dalam ketegori ini, setiap solusi dipandang sebagai individu dalam populasi, lalu kualitas solusi menentukan peluangnya untuk dipertahankan, dikombinasikan, atau dimodifikasi pada iterasi berikutnya. Contoh algoritma yang termasuk dalam kategori ini antara lain *Genetic Algorithm* (GA) (Holland, 1992), *Differential Evolution* (DE) (Storn, 1996), *Evolution Strategies* (ES) (Rechenberg, 1984), *Evolutionary Programming* (EP) (Fogel, 1999), *Flower Pollination Algorithm* (FPA) (Yang et al., 2014), serta *Memetic Algorithm* (MA) (Moscato, 1989).

Swarm-based optimozation mengambil inspirasi dari perilaku kolektif sekawanan agen sederhana yang berinteraksi di dalam suatu sistem. Agen-agen ini dapat dianalogikan sebagai kawanan burung, koloni semut, gerombolan ikan, atau hewan sosial lain yang bergerak bersama dan saling memengaruhi. Gagasan utamanya adalah bahwa kerja sama antarbanyak agen dapat menghasilkan pencarian solusi yang lebih kuat dibanding pencarian oleh satu agen saja. Contoh algoritma yang banyak dikenal dalam kategori ini adalah *Particle Swarm Optimization* (PSO) (Kennedy and Eberhart, 1995), *Ant Colony Optimization* (ACO) (Dorigo et al., 2006), *Grey Wolf Optimizer* (GWO) (Mirjalili et al., 2014), *Whale Optimization Algorithm* (WOA) (Mirjalili and Lewis, 2016), *Harris Hawks Optimization* (HHO) (Heidari et al., 2019), *Honey Badger Algorithm* (HBA) (Hashim et al., 2022), *Hunger Games Search* (HGS) (Yang et al., 2021), *Manta Ray Foraging Optimization* (MRFO) (Zhao et al., 2020b), *Marine Predators Algorithm* (MPA) (Faramarzi et al., 2020), serta *Moth-Flame Optimization* (MFO) (Mirjalili, 2015a).

Physics-based optimozation dibangun dari hukum-hukum fisika dan fenomena alam semesta. Dalam kelompok ini, pergerakan

solusi dimodelkan menggunakan analogi seperti gaya, energi, gravitasi, lubang hitam, atau teori multisemesta. Solusi diperlakukan seolah-olah bergerak mengikuti dinamika suatu sistem fisik. Contoh algoritma yang sering dikaitkan dengan kategori ini adalah *Energy Valley Optimizer* (EVO) (Azizi et al., 2023) dan *Fick's Law Optimization* (FLA) (Hashim et al., 2023).

Human-based optimozation mengambil inspirasi dari perilaku manusia, baik dalam belajar, bekerja sama, berkompetisi, maupun bertukar gagasan. Dalam kelompok ini, solusi diperbarui dengan meniru proses interaksi manusia dalam kehidupan sosial atau pembelajaran. Contoh algoritma yang dikenal dalam kategori ini adalah *Teaching-Learning-Based Optimization* (TLBO) (Rao and Patel, 2012) dan *Child Drawing Development Optimization* (CDDO) (Abdulhameed and Rashid, 2022). Pada algoritma semacam ini, pencarian solusi digambarkan sebagai proses belajar-mengajar atau proses interaksi antar indvidu manusia, sehingga mekanisme perbaikannya berangkat dari perilaku manusia, bukan dari evolusi biologis atau gerakan kawanan.

Biology-based optimozation mengambil ide dari makhluk hidup atau mikroorganisme dan berbagai proses biologis yang tidak selalu identik dengan perilaku kawanan. Berbeda dengan *swarm-based optimozation* yang lebih menekankan interaksi kolektif, *biology-based optimozation* ide dasarnya lebih luas karena dapat mencakup pertumbuhan, infeksi, adaptasi, dan respons biologis lainnya. Contoh algoritma yang termasuk dalam kategori ini adalah *Virus Colony Search* (VCS) (Li et al., 2016) dan *Bacterial Colony Optimization* (BCO) (Niu and Wang, 2012). Kategori ini memperlihatkan bahwa inspirasi biologis dalam metaheuristik dapat datang bukan hanya dari hewan sosial, tetapi juga dari organisme lain dan proses kehidupan pada tingkat yang lebih umum.

System-based optimozation mengambil inspirasi dari sistem yang lebih besar dan terorganisasi, seperti sistem ekologi, sistem imun, atau sistem jaringan. Algoritma ini tidak hanya fokus pada agen individu, tetapi pada cara komponen-komponen dalam suatu sistem

saling berhubungan dan menjaga kestabilan. Pada pendekatan ini, proses optimasi dianalogikan sebagai dinamika sebuah sistem, sehingga solusi berkembang mengikuti pola interaksi dan keseimbangan sistem tersebut. Contoh algoritma dalam kategori ini adalah *Artificial Ecosystem-based Optimization (*AEO) (Zhao et al., 2020a) dan *Water Cycle Algorithm* (WCA) (Eskandar et al., 2012).

Math-based optimozation dikembangkan dari bentuk, fungsi, atau hukum matematika. Inspirasi utamanya bukan makhluk hidup, melainkan pola matematis yang dapat dijadikan aturan pembaruan solusi. Contoh yang termasuk dalam kategori ini adalah *Sine Cosine Algorithm* (SCA) (Mirjalili, 2016) dan *Arithmetic Optimization Algorithm* (AOA) (Abualigah et al., 2021). Pada algoritma semacam ini, perpindahan solusi dibangun dari fungsi atau operator matematis tertentu, sehingga mekanisme pencariannya cenderung lebih langsung dan lebih mudah dijelaskan secara analitis dibanding pendekatan yang sepenuhnya berbasis metafora biologis.

Music-based optimozation merupakan kategori yang lebih khusus, yaitu algoritma yang mengambil inspirasi dari instrumen musik, harmoni, atau improvisasi musikal. Dalam kategori ini, proses pencarian solusi dianalogikan sebagai upaya membangun harmoni yang semakin baik. Contoh algoritma yang paling dikenal dalam kategori ini adalah *Harmony Search* (HS) (Geem et al., 2001). Walaupun algoritma dalam kategori ini tidak sebanyak kategori lain, kehadiran kelompok ini menunjukkan bahwa desain metaheuristik terus berkembang dan dapat memanfaatkan berbagai metafora selama metafora tersebut mampu diterjemahkan menjadi mekanisme eksplorasi dan eksploitasi yang efektif.

2.2 Inspirasi *Grey Wolf Optimizer*

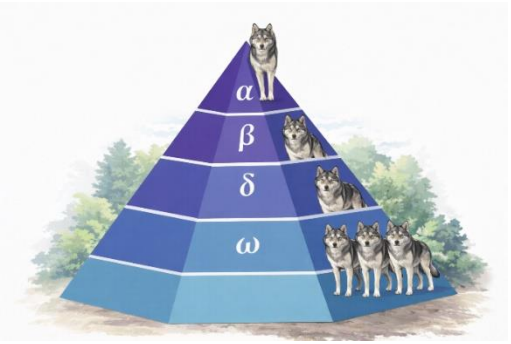
Grey Wolf Optimizer (GWO) adalah algoritma optimasi metaheuristik yang mengambil inspirasi dari perilaku sosial dan strategi berburu serigala abu-abu (*Canis lupus*) di alam. Inspirasi ini tidak muncul semata-mata dari gerakan serigala sebagai hewan

pemburu, tetapi dari dua aspek yang sangat menonjol dalam kehidupan mereka, yaitu hirarki sosial yang ketat dan mekanisme berburu secara berkelompok. Dalam konteks optimasi, kedua aspek tersebut dipandang menarik karena memperlihatkan bagaimana sekelompok individu dapat bekerja secara terstruktur untuk menemukan sasaran secara efektif. Gagasan inilah yang kemudian menjadi dasar konseptual GWO sebagai salah satu algoritma metaheuristik berbasis populasi (Mirjalili et al., 2014).

Dalam kehidupan alami, serigala abu-abu cenderung hidup dalam struktur sosial, yaitu kelompok kecil yang sangat terorganisir. Jumlah anggota kelompok umumnya berkisar antara lima sampai dua belas ekor. Di dalam kelompok tersebut berlaku hirarki dominasi yang jelas, seperti pada Gambar 2.1. Hirarki sosial serigala abu-abu. (Mech, 1999). Pada tingkat tertinggi terdapat serigala α, yaitu pemimpin utama yang berperan dalam pengambilan keputusan penting, seperti waktu berburu, tempat beristirahat, dan arah gerak kelompok. Posisi ini menunjukkan bahwa keberhasilan kelompok tidak hanya ditentukan oleh kekuatan fisik, tetapi juga oleh kemampuan memimpin dan mengatur anggota lain. Dengan kata lain, peran pemimpin menjadi sangat sentral dalam menjaga koordinasi seluruh kelompok.

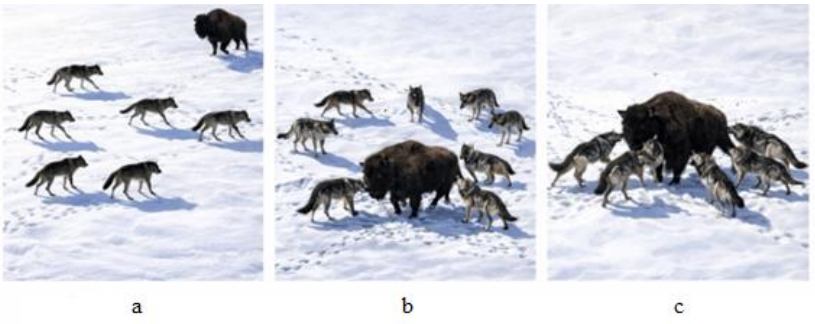
Di bawah serigala α terdapat serigala β, yaitu serigala yang berfungsi sebagai pendamping sekaligus penasihat bagi serigala α. Serigala β membantu menegakkan keputusan pemimpin dan menjaga kedisiplinan di dalam kelompok. Pada tingkat berikutnya terdapat serigala δ, yaitu anggota yang masih memiliki peran penting dalam kelompok, misalnya sebagai pemburu, penjaga, pengintai, atau perawat anggota lain. Sementara itu, tingkat paling bawah adalah serigala ω, yaitu anggota yang berada pada posisi subordinat. Walaupun serigala ω menempati tingkat terendah dalam hirarki, keberadaannya tetap penting dalam menjaga stabilitas sosial kelompok. Struktur hirarki sosial semacam ini menunjukkan adanya organisasi internal yang rapi di dalam kelompok serigala abu-abu, di mana setiap anggota memiliki fungsi yang berbeda namun tetap

saling terkait. Hirarki sosial inilah yang menjadi salah satu inspirasi utama dalam perancangan GWO.



Gambar 2.1. Hirarki sosial serigala abu-abu.

Selain struktur sosial, inspirasi penting lainnya berasal dari perilaku berburu serigala. Serigala abu-abu dikenal sebagai pemburu kelompok yang efektif. Proses berburu berlangsung melalui beberapa fase utama, yaitu melacak, mengejar, dan mendekati mangsa, kemudian mengepung dan mengganggu mangsa hingga geraknya terbatas, lalu diakhiri dengan menyerang mangsa, seperti pada Gambar 2.2 (Muro et al., 2011). Pola berburu seperti ini menunjukkan adanya perpaduan antara pencarian area yang luas, koordinasi antaranggota, dan pemusatan gerak saat target mulai teridentifikasi.



Gambar 2.2. Setrategi berburu serigala abu-abu: (a) melacak, mengejar, dan mendekati mangsa, (b) kemudian mengepung dan mengganggu mangsa hingga geraknya terbatas, (c) menyerang mangsa.

Jika dilihat lebih dalam, strategi berburu serigala abu-abu juga memperlihatkan bahwa keberhasilan kelompok tidak bergantung pada satu individu saja. Meskipun serigala α berperan sebagai pemimpin, proses berburu tetap melibatkan dukungan serigala β dan δ, sementara anggota lain mengikuti arah gerak kelompok secara kolektif. Ini memberi gambaran bahwa pencarian sasaran dilakukan melalui kerja sama, bukan melalui tindakan individual yang terpisah. Dalam optimasi, gagasan ini penting karena solusi terbaik sering kali tidak diperoleh dari satu kandidat yang bekerja sendiri, melainkan dari interaksi antar kandidat yang saling memengaruhi dan saling mengarahkan. Oleh sebab itu, GWO mengambil inspirasi dari cara serigala membangun keputusan kolektif berdasarkan posisi anggota-anggota teratas dalam hirarki.

Aspek lain yang menarik dari inspirasi ini adalah adanya keseimbangan alami antara pencarian dan penyerangan. Pada awal perburuan, serigala perlu menyebar untuk mencari keberadaan mangsa. Fase ini mencerminkan kebutuhan untuk melakukan penelusuran yang luas terhadap berbagai kemungkinan. Setelah mangsa terdeteksi, serigala mulai mempersempit gerakan, mengepung, dan mengarahkan serangan. Fase ini menunjukkan pergeseran dari pencarian luas menuju tindakan yang lebih terfokus. Dalam bahasa optimasi, pola tersebut sejalan dengan dua kebutuhan utama algoritma pencarian, yaitu eksplorasi untuk menelusuri ruang solusi dan eksploitasi untuk memperdalam pencarian di sekitar solusi yang menjanjikan. Dengan demikian, perilaku alami serigala menyediakan metafora yang kuat untuk menjelaskan bagaimana proses pencarian solusi dapat dirancang secara bertahap dan terarah.

2.3 Pemodelan Matematis *Grey Wolf Optimizer*

GWO memandang setiap kandidat solusi sebagai seekor serigala yang bergerak di dalam ruang pencarian. Posisi seekor serigala merepresentasikan satu kemungkinan solusi, sedangkan kualitas solusi dinilai melalui nilai fungsi objektif atau *fitness*.

Dengan kerangka ini, proses optimasi dapat dipahami sebagai pergerakan sekumpulan serigala menuju posisi yang semakin mendekati solusi terbaik. Pemodelan matematis GWO dibangun dari lima komponen utama, yaitu hirarki sosial**,** mengepung mangsa**,** berburu mangsa**,** menyerang mangsa sebagai eksploitasi**,** dan mencari mangsa sebagai eksplorasi**.**

2.3.1 Pemodelan Hirarki Sosial

Dalam GWO, solusi terbaik pada suatu iterasi dianggap sebagai serigala α, solusi terbaik kedua dianggap sebagai serigala β, dan solusi terbaik ketiga dianggap sebagai serigal δ. Semua solusi lain diperlakukan sebagai serigala ω. Dengan demikian, hirarki biologis serigala diterjemahkan ke dalam hirarki kualitas solusi. Serigala α, β, dan δ menjadi tiga acuan utama yang memandu proses pencarian, sedangkan serigala ω mengikuti arah yang ditentukan oleh ketiga solusi terbaik tersebut. Inti gagasan ini adalah bahwa pencarian tidak hanya bergantung pada satu solusi terbaik, tetapi pada tiga solusi terbaik sekaligus, sehingga arah gerak populasi menjadi lebih stabil dan tidak terlalu mudah bias terhadap satu kandidat saja.

2.3.2 Pemodelan Perilaku Mengepung Mangsa

Dalam perilaku alami, serigala akan mengitari mangsa sebelum menyerang. Dalam GWO, perilaku ini dimodelkan seperti pada persamaan (2.1) dan (2.2) berikut.

$$\vec{D} = \left|\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)\right| \quad (2.1)$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D} \quad (2.2)$$

Pada persamaan tersebut, $\vec{X}(t)$adalah posisi serigala saat iterasi ke-t, $\vec{X}_p(t)$adalah posisi mangsa, dan $\vec{D}$ menyatakan vektor jarak antara serigala dan mangsa. Simbol $|.|$ menyatakan nilai mutlak per komponen. Secara intuitif, persamaan (10) menghitung seberapa jauh serigala berada dari mangsa, sedangkan persamaan (11) memperbarui posisi serigala agar bergerak menuju area di sekitar mangsa. Dengan kata lain, solusi kandidat tidak langsung melompat ke posisi terbaik, tetapi bergerak relatif terhadap estimasi lokasi target.

Agar pergerakan serigala tidak bersifat kaku, GWO menggunakan dua vektor koefisien, yaitu $\vec{A}$ dan $\vec{C}$, yang didefinisikan seperti pada persamaan (2.3) dan (2.4).

$$\vec{A} = 2a \cdot \vec{r}_1 - a \tag{2.3}$$

$$\vec{C} = 2 \cdot \vec{r}_2 \tag{2.4}$$

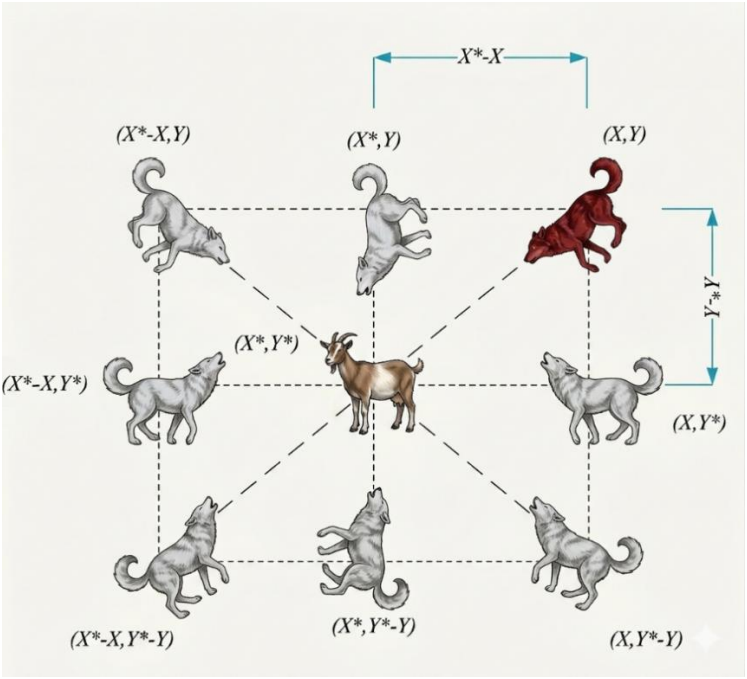
Dimana $\overrightarrow{\boldsymbol{r}_1}$ dan $\overrightarrow{\boldsymbol{r}_2}$ adalah vektor acak dalam rentang [0,1], dan a adalah parameter kontrol yang nilainya berkurang secara linear dari 2 menuju 0 sepanjang iterasi, seperti pada persamaan (2.5),

$$a = 2\left(1 - \frac{t}{T}\right) \tag{2.5}$$

dengan t adalah iterasi saat ini dan T adalah maksimum iterasi. Peran $\vec{A}$ sangat penting karena menentukan arah dan skala langkah pergerakan serigala. Sementara itu, $\vec{C}$ memberikan bobot acak terhadap posisi mangsa sehingga pergerakan tidak selalu simetris atau seragam. Secara sederhana, $\vec{A}$ mengendalikan apakah serigala lebih cenderung mendekat atau menjauh, sedangkan $\vec{C}$ memperkenalkan variasi acak agar pencarian tidak monoton. Dengan dua komponen ini, GWO dapat membangun mekanisme pencarian yang adaptif.

Visualiasi dari pemodelan mengepung mangsa dalam GWO dapat dilihat pada .Gambar 2.3 berikut. Gambar tersebut menunjukkan bahwa mekanisme pemodelan mengepung mangsa pada GWO tidak memindahkan serigala secara langsung tepat ke posisi mangsa, tetapi ke berbagai titik di sekeliling mangsa. Dalam hal ini, mangsa menjadi pusat referensi, sedangkan serigala membentuk wilayah gerak di sekitarnya. Seekor serigala yang semula berada pada posisi (X, Y), menggunakan acuan posisi mangsa (X^*, Y^*) untuk bergerak menuju posisi baru. Karena perpindahan ini dikendalikan oleh koefisien $\vec{A}$ dan $\vec{C}$, maka posisi baru yang dihasilkan bisa berada di sisi atas, bawah, kiri, kanan, maupun

diagonal dari mangsa. Keberadaan vektor acak $\vec{r}_1$ dan $\vec{r}_2$ membuat perpindahan tersebut tidak kaku, sehingga serigala dapat menempati banyak kemungkinan titik dalam area pengepungan. Dengan cara ini, proses pembaruan posisi pada GWO dapat dipahami sebagai pembentukan ruang pencarian lokal di sekitar solusi terbaik, bukan sekadar gerak lurus menuju target.



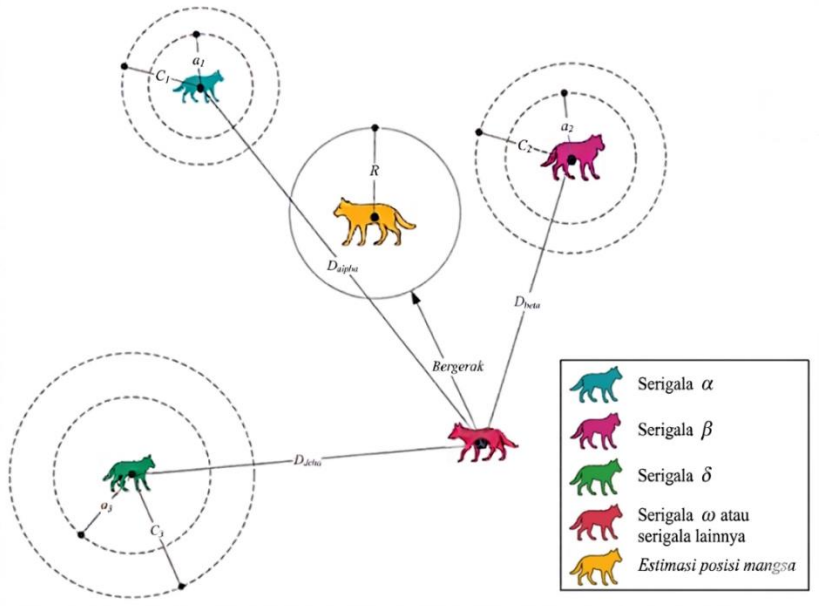
Gambar 2.3. Visualisai pemodelan perilaku mengepung mangsa

2.3.3 Pemodelan Berburu Mangsa

Dalam perilaku alaminya, serigala mampu memperkirakan posisi mangsa dan kemudian bergerak secara terkoordinasi untuk mengepungnya. Proses berburu mangsa dalam kawanan serigala abu-abu dipimpin oleh serigala α, dan kadang-kadang dibantu oleh serigala β dan δ. Ketika konsep ini dipindahkan ke dalam ruang pencarian optimasi yang abstrak, istilah mangsa tidak lagi dimaknai sebagai hewan yang diburu, melainkan sebagai posisi solusi optimum yang ingin ditemukan oleh algoritma.

Karena posisi optimum tersebut belum diketahui secara pasti, maka lokasi mangsa juga tidak dapat diamati secara langsung. Untuk mengatasi hal ini, GWO mengasumsikan bahwa tiga solusi terbaik yang ditemukan pada setiap iterasi, yaitu serigala α, β, dan δ, merupakan kandidat yang paling dekat dalam memperkirakan posisi optimum tersebut. Oleh sebab itu, ketiga solusi ini dijadikan acuan utama dalam proses pencarian, sedangkan solusi-solusi lainnya memperbarui posisinya dengan mengikuti arah yang dibentuk oleh ketiga pemimpin tersebut. Dengan cara ini, proses berburu dalam GWO dimodelkan sebagai gerakan kolektif populasi menuju wilayah solusi yang diperkirakan paling menjanjikan, sampai akhirnya mendekati posisi solusi optimum.

Gambar 2.4 memvisualisasikan pemodelan berburu mangsa pada ruang pencarian dua dimensi. Gambar tersebut memperlihatkan bahwa posisi baru serigala tidak ditentukan hanya oleh satu pemimpin, melainkan oleh kombinasi pengaruh serigala α, β, dan δ. Ketiga serigala terbaik ini bertindak sebagai titik acuan untuk memperkirakan letak mangsa, yaitu solusi optimum yang sedang dicari. Berdasarkan ketiga acuan tersebut, serigala lain kemudian memperbarui posisinya ke suatu titik baru yang berada secara acak di dalam area yang dibentuk oleh posisi serigala α, β, dan δ. Dengan demikian, perpindahan agen tidak bersifat deterministik ke satu titik tertentu, tetapi terjadi di sekitar wilayah yang diperkirakan paling menjanjikan, sehingga pencarian tetap memiliki unsur keragaman sekaligus arah yang jelas.



Gambar 2.4. Perubahan posisi serigala abu-abu dalam mencari solusi optimal.

Untuk memodelkan proses berburu mangsa, terlebih dahulu dihitung jarak setiap serigala dalam kawanan terhadap ketiga pemimpin menggunakan persamaan (2.6)-(2.8).

$$\overrightarrow{D_\alpha} = \left|\overrightarrow{C_1} \cdot \overrightarrow{X_\alpha} - \vec{X}\right| \quad (2.6)$$

$$\overrightarrow{D_\beta} = \left|\overrightarrow{C_2} \cdot \overrightarrow{X_\beta} - \vec{X}\right| \quad (2.7)$$

$$\overrightarrow{D_\delta} = \left|\overrightarrow{C_3} \cdot \overrightarrow{X_\delta} - \vec{X}\right| \quad (2.8)$$

Selanjutnya berdasarkan ketiga jarak tersebut dihitung tiga estimasi posisi serigala menggunakan persamaan (2.9)-(2.11),

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha} - \overrightarrow{A_1} \cdot \overrightarrow{D_\alpha} \quad (2.9)$$

$$\overrightarrow{X_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2} \cdot \overrightarrow{D_\beta} \quad (2.10)$$

$$\overrightarrow{X_3} = \overrightarrow{X_\delta} - \overrightarrow{A_3} \cdot \overrightarrow{D_\delta} \quad (2.11)$$

dengan $\overrightarrow{A_j}$ dan $\overrightarrow{C_j}$, $j = \alpha, \beta, \delta$ adalah vektor koefisien yang masing didefinisikan seperti pada persamaan (2.3) dan (2.4). Posisi baru

serigala pada iterasi berikutnya kemudian dihitung sebagai rata-rata dari ketiga estimasi pada persamaan (2.9)-(2.11) seperti pada persamaan (2.1).

$$\vec{X}(t+1) = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3} \tag{2.12}$$

2.3.4 Pemodelan Menyerang Mangsa (Eksploitasi)

Pada tahap akhir perburuan, serigala mulai memperkecil jarak dan memusatkan geraknya untuk menyerang mangsa. Dalam pemodelan matematis GWO, perilaku mendekati mangsa ini direpresentasikan dengan menurunkan nilai a secara bertahap sepanjang iterasi. Karena nilai $\vec{A}$ dibentuk dari a, maka rentang variasi $\vec{A}$ juga ikut menyempit. Secara lebih spesifik, $\vec{A}$ merupakan nilai acak pada interval $[-2a, 2a]$, sedangkan a nilainya berkurang secara linear dari 2 hingga 0 selama proses optimasi berlangsung. Ketika $|\vec{A}| < 1$, posisi baru agen pencari akan berada di antara posisi saat ini dan posisi mangsa, sehingga geraknya menjadi semakin dekat dan semakin terfokus pada target. Kondisi ini mencerminkan fase eksploitasi, yaitu saat populasi mulai menyerang wilayah solusi yang diperkirakan optimum. Namun, jika hanya mengandalkan mekanisme ini, pencarian berisiko terlalu cepat berkonsentrasi pada satu wilayah dan akhirnya terhenti pada solusi lokal. Oleh sebab itu, selain mekanisme pengepungan dan penyerangan, GWO juga memerlukan komponen lain yang dapat memperkuat eksplorasi agar pencarian tetap mampu menjangkau wilayah solusi yang lebih luas.

2.3.5 Pemodelan Mencari Mangsa (Eksplorasi)

Pada fase tertentu, populasi serigala dapat saling menjauh untuk memperluas pencarian, sedangkan pada fase lain akan saling mendekat ketika wilayah solusi yang menjanjikan mulai ditemukan. Untuk memodelkan perilaku menyebar tersebut, GWO memanfaatkan nilai $\vec{A}$ yang lebih besar dari 1 atau lebih kecil dari -1. Ketika $|\vec{A}| > 1$, agen pencari terdorong untuk bergerak menjauh dari posisi mangsa, sehingga pencarian tidak langsung terfokus pada satu

area saja. Mekanisme ini memperkuat eksplorasi global, karena populasi dipaksa menelusuri wilayah lain yang mungkin menyimpan solusi yang lebih baik.

Selain itu, komponen lain yang juga mendukung eksplorasi adalah vektor $\vec{C}$, yang elemennya berupa bilangan acak pada interval $[0,2]$. Vektor ini berfungsi memberikan bobot acak terhadap pengaruh mangsa dalam perhitungan jarak, sehingga posisi mangsa kadang diperkuat dan kadang justru dilemahkan. Akibatnya, gerak populasi menjadi lebih bervariasi dan tidak terlalu mudah terkonsentrasi pada satu titik. Berbeda dari $\vec{A}$, nilai $\vec{C}$ tidak diturunkan secara bertahap, karena unsur acak ini tetap dibutuhkan sepanjang proses optimasi, termasuk pada iterasi-iterasi akhir, untuk membantu populasi keluar dari risiko terjebak pada optimum lokal. Dalam interpretasi intuitif, $\vec{C}$ dapat dipandang sebagai representasi adanya hambatan di jalur perburuan, yang kadang membuat mangsa tampak lebih sulit dijangkau dan kadang justru lebih mudah didekati.

Secara garis besar proses pencarian solusi optimum menggunakan algoritma GWO dapat diringkas sebagai berikut. Mula-mula dibangkitkan populasi serigala secara acak. Pada setiap iterasi, tiga solusi terbaik disimpan sebagai serigala α, β, dan δ. Selanjutnya, setiap solusi lain memperbarui posisinya berdasarkan jaraknya terhadap tiga serigala pemimpin tersebut. Dua parameter acak, yaitu $\vec{A}$ dan $\vec{C}$, mengatur apakah populasi harus menjelajah lebih luas atau mulai memusatkan diri ke wilayah yang lebih menjanjikan. Parameter a yang menurun secara linear memastikan bahwa proses optimasi bergerak secara bertahap dari eksplorasi ke eksploitasi. Dengan struktur seperti ini, GWO dapat dipandang sebagai model pencarian kolektif yang sederhana, mudah diimplementasikan, dan memiliki mekanisme adaptif untuk menemukan solusi optimal atau mendekati optimal. Detail tahapan algoritma GWO dapat dilihat pada Algoritma 2.1.

Algoritma 2.1. *Grey wolf optimizer*.

Input:

Fungsi objektif $f(\vec{X})$, jumlah serigala dalam populasi N, jumlah iterasi maksimum T, dimensi masalah D, batas bawah $\overrightarrow{lb}$, dan batas atas $\overrightarrow{ub}$.

Output:

Solusi terbaik $\vec{X}_\alpha$ dan nilai *fitness* terbaik $f(\vec{X}_\alpha)$.

Prosedur:

1. Bangkitkan populasi awal serigala secara acak sebanyak N di dalam ruang pencarian berdimensi D.
2. Hitung nilai *fitness* setiap serigala menggunakan fungsi objektif $f(\vec{X})$.
3. Tentukan tiga serigala terbaik dalam populasi sebagai $\vec{X}_\alpha$, $\vec{X}_\beta$, dan $\vec{X}_\delta$.
4. Tetapkan iterasi awal $t \leftarrow 1$.
5. Hitung nilai a, $\vec{A}$, dan $\vec{C}$ menggunakan persamaan (2.5), (2.3), dan (2.4).
6. Selama $t \leq T$, lakukan langkah-langkah berikut:

6.1. Untuk setiap serigala ke-$i = 1,2, \dots, N$, lakukan:

   a. Perbarui posisi serigala ke-i menggunakan persamaan (2.12).

   b. Jika posisi baru berada di luar batas pencarian, lakukan penyesuaian agar tetap berada pada rentang $[\overrightarrow{lb}, \overrightarrow{ub}]$.

6.2. Hitung kembali nilai *fitness* seluruh serigala.

6.3. Perbarui nilai a, $\vec{A}$, dan $\vec{C}$ menggunakan persamaan (2.5), (2.3), dan (2.4).

6.4. Perbarui $\vec{X}_\alpha$, $\vec{X}_\beta$, dan $\vec{X}_\delta$ berdasarkan tiga nilai *fitness* terbaik pada populasi.
6.5. Set $t \leftarrow t + 1$.

7. Kembalikan $\vec{X}_\alpha$ sebagai solusi terbaik akhir dan nilai *fitness* $f(\vec{X}_\alpha)$.

03

Evolusi *Grey Wolf Optimizer*

Perkembangan *Grey Wolf Optimizer* (GWO) tidak berhenti pada formulasi awalnya sebagai algoritma *swarm intelligence* yang meniru hirarki sosial dan strategi berburu serigala abu-abu. Sejak diperkenalkan, GWO cepat menarik perhatian karena strukturnya sederhana, mudah diimplementasikan, dan mampu memberikan kinerja yang kompetitif pada berbagai masalah optimasi. Namun, seperti banyak metaheuristik lainnya, versi dasarnya juga menunjukkan sejumlah keterbatasan ketika dihadapkan pada persoalan yang lebih menantang, terutama yang melibatkan ruang solusi yang sangat luas, permukaan fungsi yang rumit, atau kebutuhan untuk menjaga keseimbangan pencarian global dan lokal secara lebih presisi. Kondisi inilah yang mendorong lahirnya berbagai pengembangan dan varian baru dari GWO (Makhadmeh et al., 2023).

Dalam perkembangannya, evolusi GWO terutama diarahkan untuk menjawab tiga tuntutan utama, yaitu meningkatkan kemampuan eksplorasi**,** mempercepat konvergensi, dan mengurangi risiko terjebak pada optimum lokal**.** Pada masalah optimasi modern, solusi yang baik tidak cukup hanya dicapai dengan bergerak cepat

menuju kandidat terbaik sementara, karena strategi seperti itu justru dapat menyebabkan konvergensi prematur. Oleh sebab itu, banyak penelitian berupaya memperbaiki dinamika pencarian GWO agar populasi tetap cukup beragam pada fase awal, tetapi juga mampu menjadi lebih fokus pada fase akhir. Arah pengembangan ini terlihat jelas pada berbagai studi yang menekankan penyempurnaan mekanisme pencarian dan strategi adaptif untuk meningkatkan performa GWO pada persoalan teknik maupun numerik berskala besar (Yu et al., 2024).

Salah satu jalur evolusi yang paling menonjol adalah modifikasi terhadap mekanisme pengendali pencarian. Pada GWO dasar, keseimbangan antara eksplorasi dan eksploitasi banyak ditentukan oleh parameter kontrol yang berubah secara linear sepanjang iterasi. Dalam praktiknya, mekanisme ini tidak selalu cukup lentur untuk menangani persoalan yang bersifat sangat kompleks atau multimodal. Karena itu, berbagai pengembangan kemudian memperkenalkan strategi yang lebih adaptif, termasuk pengaturan parameter secara nonlinier, pendekatan berbasis chaos, serta mekanisme penyesuaian yang responsif terhadap kondisi populasi selama optimasi berlangsung. Arah ini menunjukkan bahwa evolusi GWO tidak hanya berfokus pada penambahan komponen baru, tetapi juga pada penyempurnaan inti perilaku pencariannya agar lebih adaptif terhadap karakter masalah yang dihadapi (Makhadmeh et al., 2023).

Jalur evolusi lainnya tampak pada upaya hybridisasi dan rekayasa struktur populasi. Banyak pengembang melihat bahwa kekuatan GWO dapat diperluas dengan menggabungkannya dengan operator, strategi, atau mekanisme dari algoritma lain, sehingga kelemahan pada satu sisi dapat ditutup oleh kelebihan pendekatan lain. Di samping itu, perhatian juga diarahkan pada cara populasi dibentuk, diperbarui, dan dipertahankan agar kualitas interaksi antarsolusi semakin baik. Pendekatan seperti ini menjadi penting terutama ketika GWO diterapkan pada persoalan berdimensi tinggi, karena pada kondisi tersebut algoritma dasar sering menghadapi

penurunan efisiensi pencarian, lambat dalam konvergensi, atau kurang robust dalam menjaga keragaman populasi. Dengan kata lain, evolusi GWO pada tahap ini bergerak dari sekadar meniru perilaku alami menuju perancangan mekanisme optimasi yang lebih terstruktur dan problem-aware (Zhang et al., 2024).

Perkembangan GWO juga meluas ke ranah masalah yang semakin kompleks, termasuk optimasi multimodal, multi-objektif, *many-objective*, dan aplikasi spesifik seperti pembelajaran mesin, rekayasa, IoT, bioinformatika, serta sistem perencanaan. Hal ini menunjukkan bahwa evolusi GWO bukan hanya soal memperbaiki performa numerik, tetapi juga soal memperluas kemampuan algoritma agar tetap relevan pada kelas masalah yang lebih beragam. Karena itu, pembahasan mengenai evolusi GWO menjadi penting sebagai landasan untuk memahami mengapa begitu banyak varian terus dikembangkan. Pada bagian-bagian berikutnya, berbagai strategi modifikasi tersebut akan dibahas lebih rinci agar terlihat dengan jelas bagaimana GWO berevolusi dari algoritma dasar menjadi keluarga metode optimasi yang semakin kaya dan adaptif (Kalita et al., 2025; Makhadmeh et al., 2023). Beberapa modifikasi GWO yang diaplikasikan dalam pembelajaran mesin dirangkum pada Tabel 3.1

Tabel 3.1. Rangkuman beberapa modifikasi GWO yang diaplikasikan dalam pembelajaran mesin.

Varian	**Tujuan Modifikasi**	**Ide Modifikasi Utama**
Random Walk Grey Wolf Optimizer (Gupta and Deep, 2019)	Meningkatkan eksplorasi dan mengurangi risiko konvergensi prematur.	- Menambahkan *random walk* pada serigala pemimpin agar dapat bergerak acak namun terarah. - Serigala lain tetap mengikuti pemimpin seperti pada GWO standar.

Varian	Tujuan Modifikasi	Ide Modifikasi Utama
Modified Grey Wolf Optimizer (Jain et al., 2025)	Memperkaya diversitas populasi sehingga pencarian tidak mudah terjebak pada optimum lokal.	- Menginjeksikan *chaotic map* (*logistic*) ke koefisien pengendali gerak $\vec{A}$ dan $\vec{C}$. - Menambahkan operator mutasi Differential Evolution (DE) untuk membangkitkan kandidat solusi yang lebih eksploratif.
Exponential Neighborhood Grey Wolf Optimizer (Mohakud and Dash, 2022a)	Menyeimbangkan eksplorasi–eksploitasi secara lebih efektif dan memperkaya pencarian lokal.	- Mengganti penurunan linier parameter *a* menjadi penurunan nonlinier. - Menambahkan *neighborhood-based learning*: membangkitkan kandidat tetangga dan menyeleksi solusi terbaik antara hasil GWO dan hasil kandidat tetangga.
Multi-strategy Improved Grey Wolf Optimizer (Huang et al., 2025)	Meningkatkan kualitas pembaruan posisi dan memperkuat eksplorasi global.	- Pembaruan posisi dibuat berbobot dinamis berdasarkan nilai fungsi objektif dan kemajuan antargenerasi. - Eksplorasi diperkuat melalui hibridisasi dengan DE/best/2/bin dan lompatan Levy pada serigala pemimpin untuk mengurangi jebakan optimum lokal.
Hibrida *Golden Jackal Optimzer* dengan *Grey Wolf Optimzer*	konvergensi cepat GJO dengan eksplorasi GWO agar tidak terjebak optimum	- Memasukkan struktur hierarki dan peran serigala α ke dalam dinamika GJO. - Pembaruan populasi menjadi fungsi nonlinier dari tiga

Varian	Tujuan Modifikasi	Ide Modifikasi Utama
(Chopra and Mohsin Ansari, 2022)	Menggabungkan lokal.	posisi utama (*male jackal*, *female jackal*, serigala α) menggunakan interpolasi Lagrange tiga titik.
Hibrida *Grey Wolf Optimizer* dengan *Grasshopper Optimization Algorithm* (Mirjalili et al., 2018)	Menjaga keseimbangan eksplorasi–eksploitasi dan mengurangi jebakan lokal.	- Menggabungkan eksploitasi terarah GWO (mengikuti pemimpin) dengan dinamika kawanan GOA. - Pembaruan posisi memadukan arahan dua pemimpin (α dan β), lalu ditambah operator *crossover* dan *mutation* untuk meningkatkan keragaman populasi.
Hibrida *Grey Wolf Optimizer* dengan *Whale Optimization Algorithm* (Obadina et al., 2022)	Meningkatkan eksploitasi, memperhalus konvergensi, dan mengurangi risiko terjebak pada optimum lokal.	- Menyisipkan mekanisme spiral bubble-net dari WOA ke pembaruan posisi serigala α agar eksploitasi di sekitar solusi terbaik menjadi lebih halus. - Memodifikasi laju penurunan parameter *a* dari skema GWO standar agar kemampuan gerak agen tidak menyusut terlalu cepat pada iterasi akhir.

3.1 Ran*dom Walk Grey Wolf Optimizer*

Salah satu arah pengembangan GWO adalah menambahkan mekanisme *random walk* pada proses pencariannya yang dikenal dengan *Random Walk Grey Wolf Optimizer* (RW-GWO) (Gupta and Deep, 2019). Gagasan ini didasari fakta bahwa pada GWO standar,

seluruh serigala selalu memperbarui posisi berdasarkan tiga pemimpin tersebut, yaitu serigala α, β, dan δ. Pembaruan posisi tesrsebut juga termasuk pembaruan posisi serigala α yang harus mengikuti posisi serigala β dan δ serta pembaruan posisi serigala β mengikuti posisi serigala δ. Mekanisme tersebut memang efektif untuk menjaga arah pencarian, tetapi juga memiliki kelemahan: jika para pemimpin belum benar-benar berada pada wilayah solusi yang baik, maka seluruh populasi cenderung mengikuti arah yang kurang optimal. Kondisi ini dapat menyebabkan proses optimasi mudah mengalami konvergensi prematur, terutama ketika ruang solusi sangat kompleks, berdimensi tinggi, dan penuh jebakan optimum lokal. Oleh karena itu, modifikasi berbasis *random walk* diperkenalkan untuk memperkuat kemampuan eksplorasi para pemimpin, sehingga arah pencarian yang diberikan kepada populasi menjadi lebih baik.

Secara matematis, *random walk* adalah proses stokastik yang tersusun dari langkah-langkah acak berurutan. Jika setiap langkah dinyatakan sebagai s_i, maka posisi *random walk* setelah N langkah dapat ditulis seperti pada persamaan (3.1).

$$W_N = \sum_{i=1}^{N} s_i \tag{3.1}$$

Sehingga hubungan antara dua posisi berurutan dapat dinyatakan seperti pada persamaan (3.2).

$$W_N = W_{N-1} + s_N \tag{3.2}$$

Bentuk pada persamaan (3.2) menunjukkan bahwa posisi baru ditentukan oleh posisi saat ini ditambah satu langkah acak. Untuk kasus pergerakan serigala, *random walk* juga dapat dinyatakan seperti pada persamaan (3.3),

$$\vec{X}(t+1) = \vec{X}(0) + \sum_{i=1}^{N} \alpha_i \, s_i \tag{3.3}$$

dengan $\alpha_i > 0$ sebagai pengendali besar langkah pada iterasi ke-i. Dalam optimasi, persamaan ini memberi arti bahwa perpindahan solusi tidak sepenuhnya mengikuti arah yang sudah ada, tetapi juga

diberi peluang untuk melakukan langkah acak yang dapat membuka jalur pencarian baru.

Pada RW-GWO, *random walk* diterapkan khusus pada serigala pemimpin, yaitu serigala α, β, dan δ. Modifikasi ini tidak mengubah seluruh struktur GWO, tetapi hanya memperbaiki kualitas agen-agen yang paling berpengaruh dalam populasi, seperti diilustrasikan pada Gambar 3.1. Jika posisi salah satu pemimpin dinyatakan sebagai $\vec{X}_i$, maka kandidat posisi baru hasil *random walk* dapat ditulis secara konseptual seperti pada persamaan (3.4),

$$\vec{Y}_i = \vec{X}_i + \lambda_i s_i \tag{3.4}$$

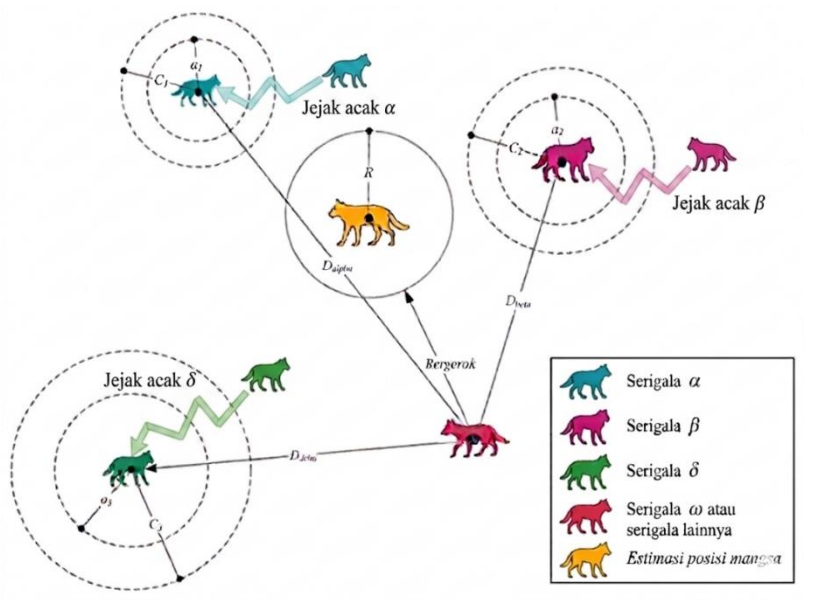
dengan $i \in \{\alpha, \beta, \delta\}$. Parameter λ_i berfungsi sebagai pengendali langkah, dan nilainya diturunkan secara bertahap dari 2 ke 0 sepanjang iterasi. Strategi ini penting karena pada tahap awal optimasi para pemimpin diberi kesempatan melakukan langkah yang lebih besar untuk menjelajahi ruang solusi secara luas, sedangkan pada tahap akhir langkahnya diperkecil agar pencarian menjadi lebih halus dan terfokus. Dengan demikian, *random walk* pada RW-GWO berperan sebagai mekanisme tambahan untuk memperkaya eksplorasi tanpa menghilangkan arah pencarian utama dari GWO.

Ukuran langkah acak pada RW-GWO diambil dari distribusi Cauchy. Pemilihan ini sangat penting karena distribusi Cauchy memungkinkan munculnya langkah yang sesekali sangat panjang. Dalam proses optimasi, lompatan seperti ini bermanfaat ketika populasi mulai stagnan di sekitar solusi lokal, karena pemimpin dapat terdorong keluar dari wilayah tersebut dan menjelajahi area lain yang mungkin lebih menjanjikan. Setelah kandidat posisi baru $\vec{\boldsymbol{Y}}_i$ dihasilkan, mekanisme *greedy selection* seperti pada persamaan (3.5) digunakan untuk menentukan posisi serigala pemimpin terbaru.

$$\text{Jika } f\left(\vec{Y}_i\right) < f\left(\vec{X}_i\right), \text{ maka } \vec{X}_i \leftarrow \vec{Y}_i \tag{3.5}$$

Maksud persamaan (3.5) adalah posisi baru serigala pemimpin diterima jika memberikan nilai fungsi objektif yang lebih baik dari pada posisi serigala pemimpin sebelumnya. Dengan skema ini, *random walk* tidak sekadar menambah keacakan, tetapi tetap

diarahkan untuk meningkatkan kualitas pemimpin. Setelah tiga pemimpin diperbarui dengan *random walk*, serigala lain tetap memperbarui posisinya menggunakan mekanisme dasar GWO seperti pada persamaan (2.12). Detail tahapan algoritma RW-GWO dapat dilihat pada Algoritma 3.1.



Gambar 3.1. Ilustrasi pergerakan serigala pada *Random Walk Grey Wolf Optimizer*.

Algoritma 3.1. *Random walk grey wolf optimizer*.

Input:

Fungsi objektif $f(\vec{X})$, jumlah serigala dalam populasi N, jumlah iterasi maksimum T, dimensi masalah D, batas bawah $\overrightarrow{lb}$, dan batas atas $\overrightarrow{ub}$.

Output:

Solusi terbaik $\vec{X}_\alpha$ dan nilai *fitness* terbaik $f(\vec{X}_\alpha)$.

Prosedur:

1. Bangkitkan populasi awal serigala secara acak sebanyak N di dalam ruang pencarian berdimensi D.

2. Hitung nilai *fitness* setiap serigala menggunakan fungsi objektif $f(\vec{X})$.

3. Tentukan tiga serigala terbaik dalam populasi sebagai $\vec{X}_\alpha$, $\vec{X}_\beta$, dan $\vec{X}_\delta$.

4. Tetapkan iterasi awal $t \leftarrow 1$.

5. Hitung nilai a, $\vec{A}$, dan $\vec{C}$ menggunakan persamaan (2.5), (2.3), dan (2.4).

6. Selama $t \leq T$, lakukan langkah-langkah berikut:

6.1. Untuk setiap serigala pemimpin $\vec{X}_i$, $i \in \{\alpha, \beta, \delta\}$ lakukan:

   a. Bangkitkan kandidat posisi baru $\vec{Y}_i$ menggunakan *random walk* pada persamaan (3.4).

   b. Jika $f(\vec{Y}_i) < f(\vec{X}_i)$, maka $\vec{X}_i \leftarrow \vec{Y}_i$.

6.2. Untuk setiap serigala ω lakukan:

   a. Perbarui posisi serigala ω menggunakan persamaan (2.12).

   b. Jika posisi baru berada di luar batas pencarian, lakukan penyesuaian agar tetap berada pada rentang $[\overrightarrow{lb}, \overrightarrow{ub}]$.

6.3. Hitung kembali nilai *fitness* seluruh serigala.

6.4. Perbarui nilai a, $\vec{A}$, dan $\vec{C}$ menggunakan persamaan (2.5), (2.3), dan (2.4).

6.5. Perbarui $\vec{X}_\alpha$, $\vec{X}_\beta$, dan $\vec{X}_\delta$ berdasarkan tiga nilai *fitness* terbaik pada populasi.

6.6. Set $t \leftarrow t + 1$.

7. Kembalikan $\vec{X}_\alpha$ sebagai solusi terbaik akhir dan nilai *fitness* $f(\vec{X}_\alpha)$.

3.2 *Modified Grey Wolf Optimizer*

Modified Grey Wolf Optimizer (MGWO) (Jain et al., 2025) merupakan pengembangan dari GWO yang dirancang untuk memperbaiki keterbatasan GWO standar, terutama pada masalah optimasi yang semakin kompleks, berdimensi tinggi, dan kaya optimum lokal. Pada GWO standar, pencarian solusi dilakukan dengan memanfaatkan tiga serigala terbaik sebagai acuan utama, sedangkan agen lain menyesuaikan posisinya mengikuti ketiga pemimpin tersebut. Meskipun mekanisme ini efektif dan sederhana, pada persoalan yang lebih sulit populasi dapat kehilangan keragaman terlalu cepat, sehingga pencarian cenderung mengarah ke wilayah tertentu sebelum ruang solusi dijelajahi secara memadai. Untuk mengatasi kelemahan tersebut, MGWO memperkuat kemampuan pencarian melalui dua modifikasi utama, yaitu penyisipan *chaotic map* ke dalam koefisien kendali GWO dan penambahan *mutation strategy* dari *Differential Evolution*.

Modifikasi pertama dilakukan pada koefisien pengendali posisi, yaitu $\vec{A}$ dan $\vec{C}$ seperti didefinisikan pada persamaan (2.3) dan (2.4). Dalam MGWO, unsur chaos dimasukkan ke dalam kedua koefisien tersebut, seperti telihat pada persamaan (3.6) dan (3.7),

$$\vec{A} = 2a \cdot \vec{r}_1 \cdot y_t - a \tag{3.6}$$

$$\vec{C} = 2 \cdot \vec{r}_2 \cdot y_t \tag{3.7}$$

dengan $y_t \in (0,1)$ adalah variabel *chaos* pada iterasi ke-*t*. Perubahan ini menyebabkan gerak agen pencari tidak lagi hanya dikendalikan oleh parameter acak biasa, tetapi juga oleh variabel *chaos*. Akibatnya, pola pergerakan populasi menjadi lebih dinamis dan lebih sulit terjebak pada pola pencarian yang sempit atau terlalu teratur.

Variabel chaos y_t pada MGWO ini dibangkitkan menggunakan *logistic map*, seperti pada persamaan (3.8),

$$y_t = r\, y_{t-1}(1 - y_{t-1}) \tag{3.8}$$

dengan parameter kontrol r umumnya diambil 4 untuk menghasilkan tingkat *chaos* maksimum. Umumhya nilai awal y_t dipilih secara acak dalam rentang 0.0001 hingga 0.9999, dengan menghindari beberapa nilai tertentu, seperti 0.25, 0.5, dan 0.75, agar pola yang dihasilkan benar-benar tetap *chaotic* ketika dikombinasikan dengan parameter r. Selain itu, pengaruh *chaos* juga disesuaikan terhadap perkembangan iterasi seperti pada persamaan (3.9),

$$y_t^{\text{scaled}} = y_t \left(1 - \frac{\text{t}}{\text{T}}\right) \tag{3.9}$$

dengan *t* adalah iterasi saat ini dan *T* adalah jumlah iterasi masksimum. Persamaan ini menunjukkan bahwa intensitas *chaos* dibuat lebih besar pada tahap awal iterasi untuk mendukung eksplorasi yang luas, kemudian secara bertahap diperkecil pada tahap akhir agar pencarian menjadi lebih terfokus. Dengan demikian, *chaos* pada MGWO tidak digunakan secara konstan, tetapi dikendalikan secara adaptif agar sesuai dengan kebutuhan fase pencarian.

Selain itu, MGWO juga menambahkan operator mutasi *differential evolution* (DE) untuk membangkitkan kandidat solusi baru yang lebih eksploratif. Strategi mutasi DE umumnya didefinisikan seperti pada persamaan (3.10),

$$\vec{v} = \vec{z}_{r1} + F(\vec{z}_{r2} - \vec{z}_{r3}) \tag{3.10}$$

dimana $\vec{z}_{r1}, \vec{z}_{r2}, \vec{z}_{r3}$ adalah vektor acak dalam populasi dan $F \in [0,2]$ adalah faktor yang mengontrol variasi diferensial $(\vec{z}_{r2} - \vec{z}_{r3})$. Dalam MGWO strategi mutasi yang digunakan adalah **DE/best/1,** seperti didefinsikan pada Persamaan (3.11), untuk membangkitkan mutasi serigala.

$$\vec{v} = \vec{z}_{best} + F(\vec{z}_{r2} - \vec{z}_{r3}) \tag{3.11}$$

Dimana, $\vec{z}_{best}$ adalah serigala yang paling dekat dengan solusi optimum, sedangkan $\vec{z}_{r2}$ dan $\vec{z}_{r3}$

serigala yang dipilih secara acak dari kawanan. Strategi ini memperkenalkan arah pencarian baru yang berasal dari kombinasi solusi terbaik dengan selisih dua solusi acak lainnya. Dengan cara tersebut, agen tidak hanya mengikuti pemimpin secara langsung, tetapi juga memperoleh dorongan tambahan untuk menjelajah wilayah solusi lain yang mungkin lebih baik. Detail tahapan algoritma MGWO dapat dilihat pada Algoritma 3.2. *Modified grey wolf optimizer*..

Algoritma 3.2. *Modified grey wolf optimizer*.

Input:

Fungsi objektif $f(\vec{X})$, jumlah serigala dalam populasi N, jumlah iterasi maksimum T, dimensi masalah D, batas bawah $\overrightarrow{lb}$, dan batas atas $\overrightarrow{ub}$, parameter chaos awal y_0, parameter kontrol *logistic map* r, dan faktor skala diferensial F.

Output:

Solusi terbaik $\vec{X}_\alpha$ dan nilai *fitness* terbaik $f(\vec{X}_\alpha)$.

Prosedur:

1. Bangkitkan populasi awal serigala secara acak sebanyak N di dalam ruang pencarian berdimensi D.
2. Hitung nilai *fitness* setiap serigala menggunakan fungsi objektif $f(\vec{X})$.
3. Tentukan tiga serigala terbaik dalam populasi sebagai $\vec{X}_\alpha$, $\vec{X}_\beta$, dan $\vec{X}_\delta$.
4. Tetapkan iterasi awal $t \leftarrow 1$.
5. Hitung nilai a menggunakan persamaan (2.5).
6. Hitung nilai y_t mengunakan persamaan (3.8) kemudian skalakan menggunakan persamaan (3.9).

---

7. Hitung nilai $\vec{A}$ dan $\vec{C}$ menggunakan persamaan (3.6) dan (3.7).

8. Selama $t \leq T$, lakukan langkah-langkah berikut:

8.1. Untuk setiap serigala ke-$i = 1,2, \ldots, N$ lakukan:

a. Perbarui posisi serigala ke-i menggunakan persamaan (2.12).

b. Bangkitkan kandidat mutasi *differential evolution* menggunakan persamaan (3.11)

c. Evaluasi nilai *fitness* hasil langkah a dan b, pilih hasil terbaik sebagai posisi baru.

d. Jika posisi baru berada di luar batas pencarian, lakukan penyesuaian agar tetap berada pada rentang $[\overrightarrow{lb}, \overrightarrow{ub}]$.

8.2. Hitung kembali nilai *fitness* seluruh serigala.

8.3. Perbarui nilai a menggunakan persamaan (2.5).

8.4. Perbarui nilai y_t mengunakan persamaan (3.8) kemudian skalakan menggunakan persamaan (3.9).

8.5. Perbarui nilai $\vec{A}$ dan $\vec{C}$ menggunakan persamaan (3.6) dan (3.7).

8.6. Perbarui $\vec{X}_\alpha$, $\vec{X}_\beta$, dan $\vec{X}_\delta$ berdasarkan tiga nilai *fitness* terbaik pada populasi.

8.7. Set $t \leftarrow t + 1$.

9. Kembalikan $\vec{X}_\alpha$ sebagai solusi terbaik akhir dan nilai *fitness* $f(\vec{X}_\alpha)$.

3.3 *Exponential Neighborhood Grey Wolf Optimizer*

Exponential Neighborhood Grey Wolf Optimizer (EN-GWO) (Mohakud and Dash, 2022a) merupakan varian GWO yang

dikembangkan untuk memperbaiki dua kelemahan utama pada GWO standar, yaitu keseimbangan eksplorasi–eksploitasi yang belum optimal serta kecenderungan populasi kehilangan keragaman terlalu cepat. Pada GWO standar, posisi agen diperbarui berdasarkan tiga posisi serigala pemimpin. Meskipun formulasi ini sederhana dan efektif, pembagian eksplorasi dan eksploitasi pada GWO standar dianggap belum cukup lentur untuk berbagai masalah optimasi yang lebih menantang. EN GWO melakukan memodifikasi melalui dua mekanisme utama, yaitu pengaturan ulang parameter adaptif a dan penambahan strategi pencarian berbasis tetangga (*neighborhood-based learning*).

Modifikasi pertama pada EN-GWO dilakukan dengan mengganti penurunan linear parameter adaptif a menjadi penurunan eksponensial. Dalam GWO sandar, nilai a berkurang secara linear dari 2 ke 0 pada setiap iterasi, seperti pada persamaan (2.5), sedangkan dalam EN-GWO digunakan penurunan eksponensial seperti pada persamaan (3.12),

$$a = 2\left(1-\frac{t^2}{T^2}\right) \tag{3.12}$$

dengan t adalah iterasi saat ini dan T adalah jumlah iterasi maksimum. Bentuk ini dirancang agar dinamika pencarian tidak membagi eksplorasi dan eksploitasi secara setengah–setengah, melainkan memberi porsi yang lebih besar pada eksploitasi, yaitu sekitar rasio 30:70. Secara intuitif, pada awal iterasi populasi masih diberi ruang untuk menjelajah, tetapi penurunan a yang lebih tajam membuat algoritma lebih cepat memasuki fase pencarian yang terfokus. Dengan demikian, perubahan sederhana pada fungsi penurun a ini menjadi mekanisme penting untuk mengatur transisi yang lebih adaptif antara pencarian global dan lokal.

Modifikasi kedua dimulai dengan menghitung kandidat posisi hasil GWO $\vec{X}_{i,GWO}(t)$ menggunakan persamaan (2.12). Dari hasil ini kemudian dihitung radius antara posisi saat ini $\vec{X}_i(t)$ dengan

$\vec{X}_{i,GWO}(t)"$" menggunakan jarak Euclidean seperti pada persamaan (3.13).

$$R_i(t) = \| \vec{X}_i(t) - \vec{X}_{i,GWO}(t) \| \quad (3.13)$$

Radius $R_i(t)$ kemudian digunakan untuk membentuk himpunan tetangga dari $\vec{X}_i(t)$, seperti pada persamaan (3.14). Himpunan tetangga ini memuat semua agen lain yang berjarak tidak lebih besar dari radius $R_i(t)$. Ide utama modifikasi ini adalah bahwa pencarian tidak lagi hanya bergantung pada tiga pemimpin global, tetapi juga memanfaatkan informasi dari lingkungan lokal di sekitar agen saat ini. Dengan cara ini, EN-GWO berusaha menambah keragaman arah gerak dan memperluas domain pencarian tanpa melepaskan panduan global dari serigala α, β, dan δ.

$$N_i(t) = \left\{ \vec{X}_j(t) \;\middle|\; D\left(\vec{X}_j(t), \vec{X}_i(t)\right) \leq R_i(t) \right\} \quad (3.14)$$

Setelah himpunan tetangga diperoleh, EN-GWO membentuk kandidat posisi *neighborhood learning* menggunakan persamaan (3.15),

$$\vec{X}_{NL}(t) = \overrightarrow{X_i}(t) + rand \cdot \left(\vec{X}_n(t) - \vec{X}_r(t)\right) \quad (3.15)$$

dengan $\vec{X}_n(t)$ adalah tetangga yang dipilih acak dari $N_i(t)$ dan $\vec{X}_r(t)$ adalah serigala yang dipilih acak dari populasi. Persamaan ini memperlihatkan bahwa posisi baru tidak hanya ditentukan oleh arahan tiga pemimpin, tetapi juga oleh perbedaan antara informasi lokal dan global. Tetangga acak memberi sinyal pencarian di sekitar wilayah yang dekat dengan agen, sementara agen acak dari populasi memasukkan unsur variasi yang lebih luas. Kombinasi ini membangun mekanisme pembelajaran lokal yang lebih kaya dibanding GWO standar. Secara konseptual, $\vec{X}_{NL}(t)$ dapat dipandang sebagai posisi alternatif yang lahir dari interaksi antara pengalaman lokal agen dengan keberagaman populasi secara keseluruhan. Langkah berikutnya pada EN-GWO adalah seleksi antara dua kandidat posisi**,** yaitu kandidat dari mekanisme GWO standar dan kandidat dari *neighborhood learning*. Pemilihan ini dilakukan

melalui persamaan (3.16). Detail tahapan algoritma EN-GWO dapat dilihat pada Algoritma 3.3.

$$\vec{X}_i(t+1) = \begin{cases} \vec{X}_{i,GWO}(t), & \text{jika } f\left(\vec{X}_{i,GWO}\right) < f\left(\vec{X}_{NL}\right) \\ \vec{X}_{NL}(t), & \text{untuk yang lainnya} \end{cases} \quad (3.16)$$

Algoritma 3.3. *Exponential neighborhood grey wolf optimizer*.

Input:

Fungsi objektif $f(\vec{X})$, jumlah serigala dalam populasi N, jumlah iterasi maksimum T, dimensi masalah D, batas bawah $\overrightarrow{lb}$, dan batas atas $\overrightarrow{ub}$.

Output:

Solusi terbaik $\vec{X}_\alpha$ dan nilai *fitness* terbaik $f(\vec{X}_\alpha)$.

Prosedur:

1. Bangkitkan populasi awal serigala secara acak sebanyak N di dalam ruang pencarian berdimensi D.
2. Hitung nilai *fitness* setiap serigala menggunakan fungsi objektif $f(\vec{X})$.
3. Tentukan tiga serigala terbaik dalam populasi sebagai $\vec{X}_\alpha$, $\vec{X}_\beta$, dan $\vec{X}_\delta$.
4. Tetapkan iterasi awal $t \leftarrow 1$.
5. Hitung nilai a, $\vec{A}$, dan $\vec{C}$ menggunakan persamaan (3.12), (2.3), dan (2.4).
6. Selama $t \leq T$, lakukan langkah-langkah berikut:
   6.1. Untuk setiap serigala ke-$i = 1,2, \ldots, N$ lakukan:
   a. Bangkitkan kandidat posisi baru $\vec{X}_{i,GWO}$ menggunakan persamaan (2.12).
   b. Hitung radius $R_i(t)$ antara posisi serigala saat ini $\vec{X}_i$

dengan $\vec{X}_{i,GWO}$ menggunakan persamaan (3.13).

c. Bentuk himpunan tetangga $N_i(t)$ menggunakan persamaan (3.14).

d. Hitung kandidat posisi berbasis *neighborhood learning* $\vec{X}_{NL}(t)$ menggunakan persamaan (3.15).

e. Seleksi posisi terbaik menggunakan persamaan (3.16).

f. Jika posisi baru berada di luar batas pencarian, lakukan penyesuaian agar tetap berada pada rentang $[\overrightarrow{lb}, \overrightarrow{ub}]$.

6.2. Hitung kembali nilai *fitness* seluruh serigala.

6.3. Perbarui nilai a, $\vec{A}$, dan $\vec{C}$ menggunakan persamaan (3.12), (2.3), dan (2.4).

6.4. Perbarui $\vec{X}_{\alpha}$, $\vec{X}_{\beta}$, dan $\vec{X}_{\delta}$ berdasarkan tiga nilai *fitness* terbaik pada populasi.

6.5. Set $t \leftarrow t + 1$.

7. Kembalikan $\vec{X}_{\alpha}$ sebagai solusi terbaik akhir dan nilai *fitness* $f(\vec{X}_{\alpha})$.

3.4 *Multi-strategy Improved Grey Wolf Optimizer*

Multi-strategy Improved Grey Wolf Optimizer (MIGWO) (Huang et al., 2025) merupakan pengembangan GWO yang dirancang untuk mengatasi kesulitan optimasi pada ruang solusi berdimensi tinggi. Pengembangannya bertumpu pada tiga arah perbaikan yang saling melengkapi, yaitu strategi inisialisasi berbasis ReliefF**,** strategi pembaruan posisi yang diperbaiki, dan strategi penguatan eksplorasi ganda. Inti gagasannya adalah bahwa GWO dasar sering kehilangan keragaman populasi sejak awal, lalu pada fase akhir ketiga serigala elit cenderung berkumpul pada solusi yang sama sehingga banyak iterasi menjadi tidak efektif. Karena itu, MIGWO tidak hanya memodifikasi satu komponen, tetapi mengubah beberapa bagian

penting sekaligus agar proses pencarian menjadi lebih cepat, lebih beragam, dan lebih tahan terhadap stagnasi pada optimum lokal.

Modifikasi pertama dilakukan pada tahap inisialisasi populasi**.** Pada MIGWO, tahap inisialisasi populasi tidak dilakukan sepenuhnya secara acak seperti pada GWO standar. Karena algoritma ini dirancang untuk menyelesaikan masalah pemilihan fitur, maka populasi awal perlu diarahkan sejak awal ke wilayah solusi yang lebih relevan. Untuk itu, terlebih dahulu dihitung korelasi antara setiap fitur X dan label Y menggunakan algoritma ReliefF (Robnik-Šikonja and Kononenko, 2003), kemudian seluruh fitur diurutkan berdasarkan tingkat nilai korelasinya.

Setelah fitur-fitur tersebut diberi peringkat, proses inisialisasi dilanjutkan dengan pendekatan *mini-initialization* (Xue et al., 2014), yaitu dengan memilih secara acak sekitar 20%–30% fitur yang paling penting untuk membentuk solusi awal dalam populasi. Strategi ini membuat populasi awal tidak lagi tersebar merata di seluruh ruang pencarian, melainkan lebih terkonsentrasi pada fitur-fitur yang sejak awal sudah dinilai memiliki kontribusi lebih besar terhadap kualitas solusi. Dengan demikian, proses pencarian dapat dimulai dari titik awal yang lebih informatif, sehingga mempercepat konvergensi dan mengurangi pengaruh fitur-fitur yang kurang relevan atau redundan.

Modifikasi kedua terdapat pada strategi pembaruan posisi oleh serigala pemimpin. Pada GWO standar, tiga serigala terbaik α, β, dan δ berkontribusi dengan bobot yang sama ketika memandu populasi. MIGWO menganggap asumsi ini kurang realistis, karena dalam praktiknya ketiga serigala pemimpin tidak selalu memiliki kualitas dan kemajuan yang sama. Untuk itu, MIGWO memberikan bobot dinamis berdasarkan dua komponen, yaitu *fitness* saat ini dan kemajuan *fitness* dari iterasi sebelumnya. Bobot berbasis fitness didefinisikan seperti pada persamaan (3.17).

$$w_i^{fitness} = \frac{1/f\left(\vec{X}_i\right)}{1/f(\vec{X}_\alpha) + 1/f(\vec{X}_\beta) + 1/f(\vec{X}_\delta)}, i \in (\alpha, \beta, \delta) \quad (3.17)$$

Sedangkan bobot berbasis kemajuan *fitness* didefinisikan seperti pada persamaan (3.18).

$$w_i^{advance} = \begin{cases} \dfrac{\Delta f(\vec{X}_i)}{\Delta f(\vec{X}_\alpha) + \Delta f(\vec{X}_\beta) + \Delta f(\vec{X}_\delta)}, & \text{jika semua } \Delta f \neq 0 \\ 0, & \text{lainnya.} \end{cases}, i \in (\alpha, \beta, \delta) \quad (3.18)$$

Kemudian bobot total setiap serigala pemimpin dihitung menggunajan persamaan (3.19).

$$w_i = \frac{w_i^{fitness} + w_i^{advance}}{\sum_i \left(w_i^{fitness} + w_i^{advance} \right)}, i \in (\alpha, \beta, \delta) \quad (3.19)$$

Posisi baru populasi lalu diperbarui menggunakan kombinasi berbobot dari tiga kandidat $\vec{X}_1, \vec{X}_2, \vec{X}_3$ seperti pada persamaan (3.20)

$$\vec{X}(t+1) = w_\alpha \vec{X}_1 + w_\beta \vec{X}_2 + w_\delta \vec{X}_3 \quad (3.20)$$

Dengan demikian, serigala pemimpin yang lebih baik dan yang menunjukkan kemajuan lebih besar akan memiliki pengaruh lebih kuat dalam memandu pencarian.

Modifikasi ketiga adalah penguatan eksplorasi melalui *Differential Evolution* (DE). MIGWO memanfaatkan keunggulan DE untuk memperluas pencarian global pada ruang berdimensi tinggi. MIGWO menggunakan strategi **DE/best/2/bin**, sehingga kandidat mutan dibentuk menggunakan persamaan (3.21),

$$\vec{v}_i = \vec{X}_\alpha + F \cdot \left(\vec{X}_{r1} - \vec{X}_{r2} \right) + F \cdot \left(\vec{X}_{r3} - \vec{X}_{r4} \right) \quad (3.21)$$

dengan $\vec{X}_{r1}$ dan $\vec{X}_{r3}$ dipilih dari serigala β dan δ, sedangkan $\vec{X}_{r2}$ dan $\vec{X}_{r4}$ dipilih dari serigala ω. Selanjutnya dilakukan operasi *crossover* pada ruang biner untuk mempertahankan lebih banyak informasi penting dari $\vec{X}_\alpha$ yang didefinisikan pada persamaan (3.22),

$$\vec{u}_i^j = \begin{cases} \vec{v}_i^j, & \text{jika } rand < 0.5 \text{ atau } j = j_{rand} \\ \vec{X}_\alpha^j, & \text{lainnya.} \end{cases} \quad (3.22)$$

di mana $i = 1,2, \dots N$ menyatakan individu dalam populasi, $j = 1,2, \dots, D$ menyatakan posisi pada ruang biner, D adalah dimensi dari masalah optimasi, dan j_{rand} adalah bilangan acak dari 1 sampai D untuk memastikan bahwa $\vec{u}_i^j$ tetap mempertahankan informasi dari $\vec{X}_\alpha$. Mekanisme ini memperluas cakupan pencarian karena kandidat baru dibentuk dari kombinasi solusi serigala pemimpin dan solusi umum, bukan hanya mengikuti arah pemimpin secara langsung.

Selain DE, MIGWO juga menambahkan strategi *Levy-flight* untuk memperbesar peluang keluar dari jebakan optimum lokal. *Levy-flight* adalah salah satu bentuk *random walk*, yaitu pergerakan acak yang tersusun dari langkah-langkah pendek dan sesekali lompatan yang jauh. Berbeda dari *random walk* biasa yang cenderung memiliki langkah relatif seragam, *Levy-flight* memungkinkan perpindahan yang kadang sangat panjang, sehingga agen pencari dapat keluar dari wilayah solusi yang sempit dan menjelajah area lain yang lebih jauh. Karena sifat ini, *Levy-flight* sering digunakan untuk memperkuat eksplorasi global dan membantu algoritma keluar dari jebakan optimum lokal.

Secara matematis, *Levy-flight* didefinisikan seperti pada persamaan (3.23),

$$Levy(\theta) = \frac{u}{|v|^{\frac{1}{\theta}}} \tag{3.23}$$

dengan u dan v masing-masing adalah variabel acak berdistribusi $N(0, \sigma_u^2)$ dan $N(0,1)$. σ_u dihitung menggunakan rumus pada persamaan (3.24),

$$\sigma_u = \left(\frac{\Gamma(1+\theta)\sin(\theta/2)}{\Gamma[(1+\theta)/2]\theta 2^{(\theta-1)/2}} \right)^{\frac{1}{\theta}} \tag{3.24}$$

dengan $\Gamma(1+\theta)$ adalah fungsi gamma yang didefinisikan seperti pada persamaan (3.25).

$$\Gamma(1+\theta) = \int_0^\infty t^\theta \, e^{-t} \, dt \tag{3.25}$$

Dalam MIGWO, Levy-flight hanya diterapkan pada serigala α, β, dan δ. Bentuk pembaruan ketiga serigala tersebut dilakukan penggunakan persamaan (3.26),

$$\vec{X}_{i,new}^{\,j} = \begin{cases} \vec{X}_i^{\,j} \times Levy(\theta), & \text{jika } rand < p_l \\ \vec{X}_i^{\,j}, & \text{lainnya.} \end{cases} \quad (3.26)$$

dengan probabilitas $p_l = 0.8$ dan $\theta = 1.5$. Strategi ini memberi kesempatan bagi serigala pemimpin untuk sesekali melakukan lompatan acak yang lebih jauh ketika diperlukan, tanpa membuat seluruh populasi bergerak terlalu tidak stabil. Tujuannya adalah agar pencarian tidak mudah terjebak pada wilayah yang sama dalam waktu terlalu lama. Detail tahapan algoritma MGWO dapat dilihat padaAlgoritma 3.4.

Algoritma 3.4. *Multi-strategy improved grey wolf optimizer*.

Input:

Fungsi objektif $f(\vec{X})$, jumlah serigala dalam populasi N, jumlah iterasi maksimum T, dimensi masalah D, batas bawah $\overrightarrow{lb}$, dan batas atas $\overrightarrow{ub}$, faktor skala diferensial F**,** probabilitas *Levy-flight* p_l, dan parameter *Levy-flight* θ.

Output:

Solusi terbaik $\vec{X}_\alpha$ dan nilai *fitness* terbaik $f(\vec{X}_\alpha)$.

Prosedur:

1. Bangkitkan populasi awal serigala sebanyak N di dalam ruang pencarian berdimensi D menggunakan strategi *ReliefF-based initialization*.
2. Hitung nilai *fitness* setiap serigala menggunakan fungsi objektif $f(\vec{X})$.
3. Tentukan tiga serigala terbaik dalam populasi sebagai $\vec{X}_\alpha$, $\vec{X}_\beta$, dan $\vec{X}_\delta$.
4. Hitung bobot serigala terbaik menggunakan persamaan (3.19).

5. Tetapkan iterasi awal $t \leftarrow 1$.
6. Hitung nilai a, $\vec{A}$, dan $\vec{C}$ menggunakan persamaan (2.5), (2.3), dan (2.4).
7. Selama $t \leq T$, lakukan langkah-langkah berikut:
   7.1. Untuk setiap serigala ke-$i = 1,2, \dots, N$ lakukan:
   a. Perbarui posisi serigala ke-*i* menggunakan persamaan (3.20).
   b. Jika posisi baru berada di luar batas pencarian, lakukan penyesuaian agar tetap berada pada rentang $[\overrightarrow{lb}, \overrightarrow{ub}]$.

   7.2. Hitung kembali nilai *fitness* seluruh serigala

   7.3. Tentukan tiga serigala terbaik dalam populasi sebagai $\vec{X}_\alpha$, $\vec{X}_\beta$, dan $\vec{X}_\delta$.

   7.4. Untuk setiap serigala ke-$i = 1,2, \dots, N$ lakukan:
   a. Hitung $\vec{u}_i$ menggunakan persamaan (3.22).
   b. Jika posisi $\vec{u}_i$ berada di luar batas pencarian, lakukan penyesuaian agar tetap berada pada rentang $[\overrightarrow{lb}, \overrightarrow{ub}]$.

   7.5. Hitung nilai *fitness* seluruh $\vec{u}_i$

   7.6. Tentukan tiga serigala terbaik dalam populasi dari $\vec{u}_i$ sebagai $\vec{X}_\alpha$, $\vec{X}_\beta$, dan $\vec{X}_\delta$.

   7.7. Untuk setiap serigala ke-$i = \alpha, \beta, \delta$ lakukan:
   a. Hitung $\vec{X}_{i,new}$ menggunakan persamaan (3.26).
   b. Jika posisi $\vec{X}_{i,new}$ berada di luar batas pencarian, lakukan penyesuaian agar tetap berada pada rentang $[\overrightarrow{lb}, \overrightarrow{ub}]$.

   7.8. Perbarui $\vec{X}_\alpha$, $\vec{X}_\beta$, dan $\vec{X}_\delta$ berdasarkan tiga nilai *fitness* $\vec{X}_{i,new}$.

7.9. Perbarui bobot serigala terbaik menggunakan persamaan (3.20).

7.10. Perbarui nilai a, $\vec{A}$, dan $\vec{C}$ menggunakan persamaan (2.5), (2.3), dan (2.4).

7.11. Set $t \leftarrow t + 1$.

10. Kembalikan $\vec{X}_\alpha$ sebagai solusi terbaik akhir dan nilai *fitness* $f(\vec{X}_\alpha)$.

3.5 Hibrida *Golden Jackal Optimzer* dengan *Grey Wolf Optimzer*

Golden Jackal Optimizer (GJO) (Chopra and Mohsin Ansari, 2022) adalah algoritma optimasi metaheuristik yang memodelkan perilaku sepasang *golden jackal* (*Canis aureus*), jantan dan betina, saat berburu mangsa. *Golden Jackal–Grey Wolf Hybrid Optimization Algorithm* (GJO–GWO) (Liu et al., 2024) merupakan algoritma hibrida yang menggabungkan dua sumber inspirasi utama, yaitu perilaku berburu *golden jackal* dan struktur kepemimpinan *grey wolf*. Tujuan utamanya adalah memperbaiki keterbatasan GJO dasar, khususnya ketika algoritma memerlukan keseimbangan yang lebih baik antara kemampuan menemukan area solusi yang menjanjikan dan kemampuan memperhalus pencarian di sekitar area tersebut. Dalam rancangan hibrida ini, mekanisme pencarian *jackal* tetap dipertahankan sebagai kerangka utama, tetapi kemudian diperkaya dengan strategi kepemimpinan serigala abu-abu serta mekanisme pembaruan posisi yang lebih kooperatif. GJO–GWO bukan sekadar menempelkan dua algoritma secara berdampingan, melainkan menyusun ulang dinamika pencarian agar lebih kaya secara struktural dan lebih stabil selama iterasi.

Pada bagian GJO, pencarian solusi didasarkan pada dua agen utama, yaitu *male jackal* dan *female jackal*, yang bergerak mengikuti posisi mangsa. Posisi kedua agen selama proses pencarian dimodelkan dalam persamaan (3.27) dan (3.28).

$$Y_1(t) = Y_M(t) - E \cdot |Y_M(t) - rl \cdot Prey(t)| \qquad (3.27)$$

$$Y_2(t) = Y_{FM}(t) - E \cdot |Y_{FM}(t) - rl \cdot Prey(t)| \quad (3.28)$$

Pada kedua persamaan tersebut, t menyatakan iterasi saat ini. $Prey(t)$ menunjukkan posisi mangsa pada iterasi ke-t. $Y_M(t)$ dan $Y_{FM}(t)$ masing-masing merepresentasikan posisi *male jackal* dan *female jackal* pada iterasi ke-t. $Y_1(t)$ dan $Y_2(t)$menyatakan posisi baru hasil pembaruan untuk *male jackal* dan *female jackal* setelah mekanisme pencarian pada iterasi tersebut dilakukan. Sedangkan E merupakan fungsi energi mangsa yang terus berkurang selama iterasi dan dimodelkan seperti pada persamaan (3.29), (3.30), dan (3.31),

$$E = E_1 \cdot E_0 \quad (3.29)$$

$$E_0 = 2r - 1 \quad (3.30)$$

$$E_1 = c_1 \left(1 - \frac{t}{T}\right) \quad (3.31)$$

dengan E_1 menggambarkan proses penurunan energi mangsa selama iterasi berlangsung yang nilainya berkurang secara linier dari c_1 sampai 0 selama proses iterasi. Sedangkan E_0 menyatakan kondisi energi awal mangsa. Nilai r adalah bilangan acak pada rentang $[0, 1]$, c_1 merupakan konstanta yang bernilai 1.5, dan T menunjukkan jumlah iterasi maksimum yang digunakan oleh algoritma.

Pada persamaan (3.27) dan (3.28) rl merupakan bilangan acak yang dibangkitkan dari distribusi Levy seperti pada persamaan (3.32) dan (3.33),

$$rl = 0.05\ LF(y) \quad (3.32)$$

$$LF(\beta) = \frac{0.01(\mu\sigma)}{|v|^{\frac{1}{\beta}}} \quad (3.33)$$

dengan u dan v adalah bilangan acak pada rentang $[0, 1]$, β adalah konstanta yang nilainya 1.5, dan σ adalah konstanta yang dihitung menggunakan persamaan (3.24).

Formulasi ini menunjukkan bahwa GJO sejak awal sudah memiliki unsur eksplorasi yang cukup kuat karena geraknya dipengaruhi oleh energi mangsa dan bilangan acak Levy. Namun, dua agen utama saja kadang belum cukup untuk menghasilkan dinamika pencarian yang stabil pada masalah yang lebih kompleks.

Perbaikan pertama pada algoritma hibrida GJO-GWO ini adalah memasukkan peran serigala α pada GWO ke dalam struktur GJO. Dengan tambahan ini, proses pencarian tidak lagi hanya bergantung pada pasangan *jackal*, tetapi juga dibantu oleh satu agen elit tambahan yang bertindak sebagai pemimpin tingkat kedua. Pada fase pencarian dan pelacakan, posisi agen ketiga tersebut dimodelkan seperti pada persamaan (3.34).

$$Y_3(t) = Y_\alpha(t) - A \cdot |Y_\alpha(t) - C \cdot Prey(t)| \quad (3.34)$$

Sementara pada fase pengepungan dan serangan dimodelkan seperti pada persamaan (3.35),

$$Y_3(t) = Y_\alpha(t) - A \cdot |C \cdot Y_\alpha(t) - Prey(t)| \quad (3.35)$$

dengan A dan C adalah koefisien pada GWO yang didefinisikan pada persamaan (2.3) dan (2.4), serta $Y_\alpha(t)$ adalah posisi serigala α.

Namun menjaga agar model tidak menjadi terlalu kompleks karena masuknya parameter baru A dan C, formulasi ini kemudian disederhanakan dengan mengganti pengaruh A dan C menggunakan komponen energi E dan bilangan acak rl yang sudah ada pada GJO dasar, sehingga pemodelan posisi ketiga agen pada fase pencarian dan pelacakan menjadi seperti pada persamaan (3.27), (3.28), dan (3.36).

$$Y_3(t) = Y_\alpha(t) - E \cdot |Y_\alpha(t) - rl \cdot Prey(t)| \quad (3.36)$$

Sedangkan pada fase pengepungan dan serangan dimodelkan seperti pada persamaan (3.37), (3.38), dan (3.39).

$$Y_1(t) = Y_M(t) - E \cdot |rl \cdot Y_M(t) - Prey(t)| \quad (3.27)$$

$$Y_2(t) = Y_{FM}(t) - E \cdot |rl \cdot Y_{FM}(t) - Prey(t)| \quad (3.28)$$

$$Y_3(t) = Y_\alpha(t) - E \cdot |rl \cdot Y_\alpha(t) - Prey(t)| \quad (3.39)$$

Dengan penambahan ini, ruang pencarian populasi menjadi lebih luas pada tahap awal, sementara kemampuan mengepung mangsa juga meningkat pada tahap akhir.

Masuknya serigala α juga menuntut perubahan pada mekanisme pembaruan posisi populasi. Pada GJO dasar, posisi baru

dihitung hanya sebagai rata-rata dua agen *jackal*, seperti pada persamaan (3.40).

$$Y(t+1) = \frac{Y_1(t) + Y_2(t)}{2} \tag{3.40}$$

Setelah serigala α diperkenalkan, rumus ini tidak lagi memadai karena pengaruh agen ketiga belum tercakup. Untuk itu, algoritma hibrida ini menggunakan interpolasi Lagrange tiga titik agar *male jackal*, *female jackal*, dan serigala α semuanya terlibat langsung dalam pembaruan solusi. Persamaan pembaruan populasinya dinyatakan pada persamaan (3.41).

$$Y(t+1) = \sum_{i=1}^{3} \frac{Y_i(t)}{3} \prod_{\substack{j=1 \\ j \neq i}}^{3} \frac{Y(t) - Y_j(t)}{Y_i(t) - Y_j(t)} \tag{3.41}$$

Rumus ini dirancang agar pembaruan posisi tidak sekadar berupa rata-rata sederhana, tetapi benar-benar mempertimbangkan hubungan antar-tiga agen utama yang saling bekerja sama. Dengan kata lain, interpolasi Lagrange dipakai sebagai mekanisme untuk membangun pembaruan posisi yang lebih halus, lebih kooperatif, dan lebih sensitif terhadap perubahan arah pencarian.

Secara algoritmik, struktur hibrida GJO–GWO dapat dipahami sebagai proses yang berulang. Mula-mula populasi *prey* diinisialisasi, lalu nilai *fitness* setiap individu dievaluasi untuk menentukan *male jackal*, *female jackal*, dan serigala α. Setelah itu, pada setiap iterasi dihitung bilangan acak *Levy* serta energi mangsa, kemudian agen-agen utama memperbarui posisinya sesuai kondisi eksplorasi atau eksploitasi. Jika $|E| < 1$, proses bergerak pada fase eksploitasi yang lebih terfokus. Sebaliknya, jika $|E| \geq 1$, proses masih berada pada fase eksplorasi yang lebih luas. Setelah posisi $Y_1(t)$, $Y_2(t)$, dan $Y_3(t)$ diperoleh, populasi diperbarui menggunakan persamaan (3.41), lalu dilakukan penanganan batas pencarian sebelum iterasi berikutnya dimulai. Detail algoritma hibrida GJO-GWO dapat dilihat pada .

Algoritma 3.5. *Golden jackal–grey wolf hybrid optimization algorithm.*

Input:
Fungsi objektif $f(Y)$, jumlah serigala dalam populasi N, jumlah iterasi maksimum T, dimensi masalah D, batas bawah $\overrightarrow{lb}$, dan batas atas $\overrightarrow{ub}$.

Output:
Solusi terbaik Y_M dan nilai *fitness* terbaik $f(Y_M)$.

Prosedur:

1. Bangkitkan populasi awal *prey* secara acak sebanyak N di dalam ruang pencarian berdimensi D.

2. Hitung nilai *fitness* setiap individu dalam populasi menggunakan fungsi objektif $f(Y)$.

3. Tentukan tiga individu terbaik dalam populasi sebagai Y_M, Y_{FM}, dan Y_α.

4. Tetapkan iterasi awal $t \leftarrow 1$.

5. Hitung nilai E dan rl menggunakan persamaan (3.29) dan (3.32).

6. Selama $t \leq T$, lakukan langkah-langkah berikut:
6.1. Untuk setiap individu ke-$i = 1,2, \ldots, N$ dalam populasi, lakukan:

   a. Jika $|E| < 1$ perbarui posisi Y_1, Y_2, dan Y_3. menggunakan persamaan (3.37), (3.38), dan (3.39).

   b. Jika $|E| \geq 1$ perbarui posisi Y_1, Y_2, dan Y_3. menggunakan persamaan (3.27), (3.28), dan (3.36).

   c. Hitung posisi baru individu dengan mekanisme kolaboratif berbasis interpolasi

Lagrange tiga titik menggunakan persamaan (3.41).

d. Jika posisi baru berada di luar batas pencarian, lakukan penyesuaian agar tetap berada pada rentang $[\overrightarrow{lb}, \overrightarrow{ub}]$.

6.2. Hitung kembali nilai *fitness* seluruh individu dalam populasi.

6.3. Perbarui nilai nilai E dan rl menggunakan persamaan (3.29) dan (3.32).
6.4. Perbarui Y_M, Y_{FM}, dan Y_α berdasarkan tiga nilai *fitness* terbaik pada populasi.
6.5. Set $t \leftarrow t + 1$.

7. Kembalikan Y_M sebagai solusi terbaik akhir dan nilai *fitness* $f(Y_M)$.

3.6 Hibrida *Grey Wolf Optimizer* dengan *Grasshopper Optimization Algorithm*

Grasshopper Optimization Algorithm (GOA) (Mirjalili et al., 2018) adalah algoritma optimasi metaheuristik yang meniru perilaku kawanan belalang, yaitu interaksi sosial saat bergerak. Hibrida *Gray Wolf Optimization* dengan *Grasshopper Optimization Algorithm* (GWO–GOA) (Purushothaman et al., 2020) dirancang untuk memadukan dua karakter pencarian yang berbeda. GWO dikenal memiliki kecenderungan eksploitasi yang kuat dan mampu bergerak cepat menuju kandidat solusi yang baik, tetapi pada beberapa kasus eksplorasinya belum cukup kuat sehingga masih berisiko gagal menemukan solusi global terbaik. Sebaliknya, GOA memiliki mekanisme adaptif yang baik dalam menyeimbangkan eksplorasi dan eksploitasi, serta relatif lebih mampu menangani ruang pencarian yang kompleks. Atas dasar itu, keduanya digabungkan agar kelemahan salah satu algoritma dapat ditutupi oleh keunggulan algoritma lainnya, sehingga diharapkan diperoleh proses pencarian

yang lebih stabil, lebih cepat konvergen, dan tidak mudah terjebak pada optimum lokal.

Pada sisi GOA, pergerakan agen didasarkan pada tiga komponen utama, yaitu interaksi sosial, gaya gravitasi, dan adveksi angin. Secara matematis, posisi *grasshopper* ke-k dinyatakan seperti pada persamaan (3.42),

$$X_k = S_k + G_k + A_k \tag{3.42}$$

dengan S_k menyatakan interaksi sosial, G_k adalah gaya gravitasi, dan A_k adalah adveksi angin. Komponen interaksi sosial dimodelkan seperti pada persamaan (3.43),

$$S_k = \sum_{l=1,\, l\neq 1}^{N} s(d_{kl})\hat{d}_{kl} \tag{3.43}$$

di mana $d_{kl} = |x_l - x_k|$ adalah jarak antara *grasshopper* ke-k dan ke-l, sedangkan $\hat{d}_{kl} = \frac{x_l - x_k}{d_{kl}}$ adalah vektor satuan yang merupakan arah dari *grasshopper* ke-k menuju *grasshopper* ke-l, dan *N* adalah banyaknya *grasshopper*. Sedangkan *s* adalah fungsi sosial yang didefinisikan seperti pada persamann (3.44),

$$s(r) = fe^{-\frac{r}{l}} - e^{-r} \tag{3.44}$$

dengan f adalah intensitas gaya tarik dan l adalah skala panjang gaya tarik. Selanjutnya, gaya gravitasi dan angin masing-masing dimodelkan seperti pada persamaan (3.45) dan (3.46),

$$G_k = -g\hat{e}_g \tag{3.45}$$

$$A_k = u\hat{e}_w \tag{3.46}$$

dengan g adalah adalah konstanta grafitasi, $\hat{e}_g$ adalah vektor satuan yang mengarah ke pusat bumi, u adalah konstanta penyimpangan, dan $\hat{e}_w$ adalah vektor satuan arah angin. Formulasi ini menunjukkan bahwa GOA pada dasarnya membangun gerak populasi melalui interaksi kolektif antarsolusi, bukan hanya mengikuti satu atau dua pemimpin terbaik.

Dalam bentuk yang telah diarahkan untuk optimasi, GOA memperbarui posisi agen dengan mempertimbangkan posisi agen lain

sekaligus target saat ini. Model matematika untuk pembaruan tersebut dinyatakan dalam persamaan (3.47)

$$X_i^d = c\left(\sum_{j=1,\ j\neq 1}^{N} c\frac{ub_d - lb_d}{2}\, s\left(\left|x_j^d - x_i^d\right|\right)\hat{d}_{kl}\right) + \hat{T}^d \qquad (3.47)$$

dengan ub_d dan lb_d masing-masing adalah batas atas dan bawah pada dimensi ke-d. Fungsi s menyatakan gaya interaksi sosial yang didefinisikan pada persamaan (3.44), $\hat{T}_d$ adalah nilai pada dimensi ke-d dari solusi terbaik yang telah ditemukan sejauh ini, dan $c$merupakan koefisien yang menurun secara bertahap untuk memperkecil area nyaman (comfort area), area tolak-menolak, dan area tarik-menarik.

Pada algoritma hibrida GWO–GOA, mekanisme pembaruan posisi tidak lagi sepenuhnya mengikuti pola GOA asli. Posisi agen diperbarui dengan memanfaatkan dua pemimpin utama yang diambil dari struktur GWO, yaitu serigala α dan β. Pembaharuan dilakukan dengan menghitung rata-rata pergeseran posisi saat ini terhadap posisi serigala α dan β, seperti pada persamaan (3.48),

$$X(t+1) = X + f_1 \cdot rand \cdot \frac{(X_\alpha - X) + (X_\beta - X)}{2} \qquad (3.48)$$

di mana X adalah posisi saat ini, sedangkan X_α dan X_β mewakili arahan dari dua agen terbaik. Persamaan (3.48) memperlihatkan bahwa solusi baru dihasilkan dari posisi saat ini yang didorong menuju rata-rata arah dua pemimpin, lalu dikalikan faktor acak agar geraknya tidak terlalu deterministik. Dengan demikian, struktur kepemimpinan dari GWO dimanfaatkan untuk memberi arah pencarian yang lebih jelas, sementara karakter interaksi dan penyesuaian gerak dari GOA tetap dipertahankan.

Sedangkan pembaruan posisi serigala α dan β hanya ditentukan oleh serigala α, seperti pada persamaan (3.49).

$$X(t+1) = X + f_1 \cdot rand \cdot (X_\alpha - X) \qquad (3.49)$$

Strategi ini diposisikan sebagai mekanisme yang membantu mengurangi risiko pencarian terlalu cepat berkumpul pada satu wilayah yang sama. Dengan kata lain, walaupun GWO–GOA memanfaatkan arahan elit, algoritma ini tetap berusaha menjaga keseimbangan agar peran pemimpin tidak justru menyebabkan stagnasi dini. Langkah ini juga dipandang sebagai bentuk strategi penurunan bertahap agar proses pencarian tidak mudah jatuh pada optimum lokal.

Keseimbangan antara eksplorasi dan eksploitasi pada GWO–GOA dikendalikan lebih lanjut oleh parameter kontrol l yang didefinisikan pada persamaan (3.50),

$$l = 2 - \left(\cos\left(rand(\quad)\right) \cdot \frac{1}{M_{itera}}\right) \tag{3.50}$$

yang berfungsi mengatur arah dominan pencarian. Nilai $l$yang lebih besar cenderung mendorong fase eksplorasi, sedangkan nilai yang lebih kecil membuat pencarian bergerak ke fase eksploitasi. Selain itu, algoritma ini juga menambahkan operator *crossover* dan *mutation* untuk memperkaya keragaman solusi. Operasi *crossover* dinyatakan pada persamaan (3.51).

$$z_{k,l} = \begin{cases} z_{v,l}, & \text{jika } rand < 0.5 \\ z_{k,l}, & \text{lainnya} \end{cases} \tag{3.51}$$

Sedangkan operasi mutation dinyatakan pada persamaan (3.52).

$$z_{k,l} = \begin{cases} z_{Sl} + \mu\left(z_{a,l} - z_{b,l}\right), & \text{jika } rand < 0.07 \\ z_{k,l}, & \text{lainnya} \end{cases} \tag{3.52}$$

Kehadiran dua operator ini menunjukkan bahwa GWO–GOA tidak hanya mengandalkan pembaruan posisi utama, tetapi juga memberi peluang munculnya kombinasi solusi baru yang lebih beragam untuk menghindari konvergensi prematur. Detail algoritma hibrida GWO-GOA dapat dilihat pada Algoritma 3.6.

Algoritma 3.6. Hibrida *gray wolf optimization* dengan *grasshopper optimization algorithm*.

Input:
Fungsi objektif $f(Y)$, jumlah serigala dalam populasi N, jumlah iterasi maksimum M_{itera}, batas bawah lb, batas atas ub, parameter f, g, u,

Output:
Solusi terbaik X_α dan nilai *fitness* terbaik $f(X_\alpha)$.

Prosedur:

1. Inisialisasi populasi agen pencari secara acak.
2. Inisialisasi parameter algoritma GWO–GOA.
3. Hitung nilai *fitness* seluruh agen pencari menggunakan fungsi objektif.
4. Tentukan tiga individu terbaik dalam populasi sebagai X_α, X_β, dan X_δ.
5. Tetapkan iterasi awal $t \leftarrow 1$.
6. Selama $t \leq M_{itera}$, lakukan langkah-langkah berikut:

   6.1. Hitung parameter kontrol $l$menggunakan persamaan (3.50).

   6.2. Untuk setiap agen pencari X_i**,** lakukan pembaruan posisi:

   a. Hitung interaksi sosial antaragen menggunakan Persamaan (3.43) dan fungsi sosial pada Persamaan (3.44).

   b. Hitung komponen gravitasi menggunakan Persamaan (3.45).

   c. Hitung komponen adveksi angin menggunakan Persamaan (3.46).

d. Hitung posisi baru agen berdasarkan model optimasi GOA menggunakan Persamaan (3.47).

6.3. Lakukan pembaruan posisi berbasis elit GWO

a. Perbarui posisi agen biasa dengan memanfaatkan arah dua pemimpin utama menggunakan Persamaan (3.48).

b. Perbarui posisi serigala α dan β menggunakan Persamaan (3.49).

6.4. Untuk setiap elemen solusi, lakukan crossover menggunakan Persamaan (3.51).

6.5. Untuk setiap elemen solusi, lakukan mutation menggunakan Persamaan (3.52).

6.6. Jika ada posisi yang keluar dari batas pencarian, kembalikan ke rentang $[lb, ub]$.

6.7. Hitung kembali nilai *fitness* seluruh agen dan perbarui X_α, X_β, dan X_δ.

6.8. Set $t \leftarrow t + 1$.

7. Kembalikan X_α sebagai solusi terbaik akhir dan nilai *fitness* $f(X_\alpha)$.

3.7 Hibrida *Grey Wolf Optimizer* dengan *Whale Optimization Algorithm*

Whale Optimization Algorithm (WOA) (Mirjalili and Lewis, 2016) adalah algoritma optimasi metaheuristik berbasis populasi yang meniru perilaku berburu paus bungkuk, khususnya strategi *bubble-net feeding*. Pada strategi ini, paus membentuk gelembung dalam lintasan melingkar atau spiral untuk mengepung kawanan mangsa sebelum menyerang. Dari inspirasi biologis tersebut, WOA dimodelkan ke dalam tiga mekanisme utama, yaitu mengepung

mangsa, mendekati mangsa dengan lintasan spiral, dan mencari mangsa secara acak. Karena itu, WOA memiliki dua kemampuan penting sekaligus, yakni menjelajah ruang solusi secara global dan memperhalus pencarian di sekitar solusi terbaik yang telah ditemukan.

Pada WOA, fase pengepungan mangsa dinyatakan dimodelkan seperti pada persamaan (3.53) dan (3.54),

$$\vec{D} = \left|\vec{C} \cdot \vec{X}^*(t) - \vec{X}(t)\right| \tag{3.53}$$

$$\vec{X}(t+1) = \vec{X}^*(t) - \vec{A} \cdot \vec{D} \tag{3.54}$$

dengan $\vec{X}^*(t)$ adalah solusi terbaik saat ini dan $\vec{X}(t)$ adalah posisi paus. Dua koefisien $\vec{A}$ dan $\vec{C}$ masing-masing dihitung menggunakan persamaan (3.55) dan (3.56),

$$\vec{A} = 2\vec{a} \cdot \vec{r} - \vec{a} \tag{3.55}$$

$$\vec{C} = 2\vec{r} \tag{3.56}$$

di mana $\vec{a}$ nilainya diturunkan secara linear dari 2 ke 0 sepanjang iterasi seperti pada persamaan (2.5), sedangkan $\vec{r}$ adalah bilangan acak pada interval $[0,1]$.

Selain gerak pengepungan biasa, WOA juga menggunakan pendekatan pergerakan spiral untuk meniru gerakan berbentuk heliks paus bungkuk yang dimodelkan seperti pada persamaan (3.57),

$$\vec{X}(t+1) = \vec{D}'e^{bl}\cos(2\pi l) + \vec{X}^*(t) \tag{3.57}$$

dengan $\vec{D}' = \left|\vec{X}^*(t) - \vec{X}(t)\right|$ menyatakan jarak menunjukkan jarak paus ke-i ke mangsa (solusi terbaik yang diperoleh sejauh ini) , b adalah konstanta untuk mendefinisikan bentuk spiral logaritmik, dan l adalah bilangan acak pada interval $[-1,1]$. Pada setiap iterasi, kedua mekanisme ini dipilih secara bergantian dengan peluang 50%, seperti pada persamaan (3.58), sehingga eksploitasi lokal tidak hanya berlangsung secara lurus, tetapi juga dapat mengikuti lintasan spiral di sekitar target.

$$\vec{X}(t+1) = \begin{cases} \vec{X}^*(t) - \vec{A} \cdot \vec{D}, & \text{jika } p < 0.5 \\ \vec{D}' e^{bl} \cos(2\pi l) + \vec{X}^*(t), & \text{jika } p \geq 0.5 \end{cases} \quad (3.58)$$

Di sisi lain, GWO dikenal kuat dalam mengarahkan populasi menuju area solusi yang menjanjikan melalui struktur kepemimpinan serigala α, β, dan δ. Karena itu, hibrida GWO–WOA (Obadina et al., 2022) dikembangkan untuk menggabungkan kekuatan keduanya. GWO memberi kerangka eksplorasi dan arah pencarian yang kuat, sedangkan WOA menyumbang mekanisme eksploitasi yang lebih lentur melalui gerak spiral. Tujuan penggabungan ini adalah memperoleh keseimbangan yang lebih baik antara eksplorasi dan eksploitasi, sehingga proses pencarian menjadi lebih akurat, lebih cepat konvergen, dan lebih kecil kemungkinan terjebak pada optimum lokal.

Inti pertama dari hibrida ini terletak pada modifikasi pembaruan posisi serigala α. Karena serigala α adalah solusi terbaik pada populasi, kualitas pembaruan posisinya akan sangat menentukan arah pencarian seluruh kawanan. Pada GWO standar, serigala α bergerak dengan mekanisme pengepungan biasa. Dalam GWO–WOA, fase ini diperbaiki dengan menyisipkan konsep spiral dari WOA ke dalam jarak pengepungan serigala α pada persamaan (2.6) menjadi seperti pada persamaan (3.59),

$$D_\alpha = \begin{cases} r\left|\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}\right|, & \text{jika } p < 0.5 \\ \rho e^{bl} \cos(2\pi l) \left|\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}\right|, & \text{jika p} \geq 0.5 \end{cases} \quad (3.59)$$

dengan p, r, dan ρ adalah bilangan acak pada interval $[0,1]$, serta l adlah bilangan acak pada interval $[-1,1]$. Bentuk ini menunjukkan bahwa serigala α tidak selalu mendekati target secara langsung, tetapi kadang menggunakan lintasan spiral agar eksploitasi di sekitar solusi terbaik menjadi lebih halus.

Penggunaan dua mode gerak pada persamaan (3.59) didasarkan pada asumsi yang juga dipakai dalam WOA, yaitu adanya peluang 50% untuk bergerak dengan lingkaran menyusut dan 50% untuk bergerak dengan lintasan spiral. Jika $p < 0.5$, pembaruan

mengikuti mekanisme mendekat biasa. Jika $p \geq 0.5$, jarak dimodifikasi dengan faktor spiral $\rho e^{bl}\cos(2\pi l)$. Dengan strategi ini, eksploitasi tidak berlangsung terlalu kaku. Populasi tetap bergerak menuju solusi terbaik, tetapi diberi variasi lintasan agar pencarian di sekitarnya lebih kaya dan tidak cepat stagnan.

Modifikasi kedua terletak pada vektor $\vec{a}$ yang pada GWO berfungsi mengatur fase serangan atau eksploitasi. Pada GWO biasa, vektor ini menurun linear dari 2 ke 0 seperti pada persamaan (2.5).

Sedangkan pada GWO–WOA vektor ini diubah sehingga nilainya menurun dari 2 ke 1 seperti pada persamaan (3.60),

$$\vec{a} = 2\left(1-\frac{t}{2T}\right) \tag{3.60}$$

dengan t adalah iterasi saat ini dan T adalah maksimum iterasi. Perubahan ini membuat kemampuan gerak agen tidak menyusut terlalu cepat pada iterasi akhir. Dengan demikian, populasi tetap diarahkan untuk mengeksploitasi wilayah yang menjanjikan, tetapi masih memiliki fleksibilitas yang cukup untuk memperbaiki solusi. Modifikasi ini dimaksudkan untuk meningkatkan akurasi konvergensi sekaligus menurunkan risiko konvergensi prematur.

Yang menarik, hibridisasi ini tidak mengubah seluruh struktur GWO. Posisi serigala β dan δ, serta bagian-bagian umum lain dari GWO, tetap dipertahankan. Artinya, algoritma ini bersifat selektif: hanya komponen yang dianggap paling kritis terhadap kualitas solusi akhir yang diperbaiki, yakni pembaruan posisi α dan laju penurunan $\vec{a}$. Pendekatan ini menunjukkan bahwa tujuan hibrida bukan membangun algoritma yang sepenuhnya baru, melainkan memperkuat titik lemah GWO dengan memanfaatkan mekanisme spiral dari WOA.

Secara operasional, GWO–WOA dimulai dengan menginisialisasi populasi agen pencari dan mengevaluasi fitness seluruh agen. Setelah itu dipilih tiga agen terbaik sebagai serigala α, β, dan δ. Pada setiap iterasi, parameter $\vec{a}$, $\vec{A}$, dan $\vec{C}$ diperbarui masing-masing menggunakan persamaan (3.60), (2.3), dan (2.4), lalu posisi

semua agen dihitung. Perlakuan khusus diberikan pada D_α melalui Persamaan (3.59), sedangkan untuk agen lain tetap mengikuti mekanisme GWO. Konstanta acak seperti p, l, dan ρ juga diperbarui pada setiap iterasi. Setelah itu seluruh populasi dievaluasi kembali, dan serigala α, β, serta δ diperbarui jika ditemukan solusi yang lebih baik. Dengan kata lain, hibridisasi berlangsung terutama pada tahap pembaruan posisi, bukan pada tahap evaluasi fitness atau pembentukan populasi. Detail algoritma hibrida GWO-WOA dapat dilihat pada Algoritma 3.1.

Algoritma 3.7. *Hibrida grey wolf optimizer dengan whale optimization algorithm*.

Input:

Fungsi objektif $f(\vec{X})$, jumlah serigala dalam populasi N, jumlah iterasi maksimum T, dimensi masalah D, batas bawah $\overrightarrow{lb}$, dan batas atas $\overrightarrow{ub}$.

Output:

Solusi terbaik $\vec{X}_\alpha$ dan nilai *fitness* terbaik $f(\vec{X}_\alpha)$.

Prosedur:

1. Bangkitkan populasi awal serigala secara acak sebanyak N di dalam ruang pencarian berdimensi D.
2. Hitung nilai fitness setiap serigala menggunakan fungsi objektif $f(\vec{X})$.
3. Tentukan tiga serigala terbaik dalam populasi sebagai $\vec{X}_\alpha$, $\vec{X}_\beta$, dan $\vec{X}_\delta$.
4. Tetapkan iterasi awal $t \leftarrow 1$.
5. Selama $t \leq T$, lakukan langkah-langkah berikut:

   5.1. Hitung nilai a, $\vec{A}$, dan $\vec{C}$ menggunakan persamaan (3.60), (2.3), dan (2.4).

5.2. Bangkitkan bilangan acak p, r, ρ, dan l.

5.3. Untuk setiap serigala ke-$i = 1,2, \ldots, N$, lakukan:

a. Hitung jarak terhadap serigala α menggunakan bentuk hibrida pada persamaan (3.59).

b. Hitung jarak terhadap serigala β dan δ menggunakan persamaan (2.7) dan (2.8)

c. Hitung tiga estimasi posisi berdasarkan α, β, dan δ menggunakan Persamaan (2.9), (2.10), dan (2.11).

d. Perbarui posisi serigala ke-i menggunakan persamaan (2.12).

e. Jika posisi baru berada di luar batas pencarian, lakukan penyesuaian agar tetap berada pada rentang $[\overrightarrow{lb}, \overrightarrow{ub}]$.

5.4. Hitung kembali nilai fitness seluruh serigala.

5.5. Perbarui $\vec{X}_\alpha$, $\vec{X}_\beta$, dan $\vec{X}_\delta$ berdasarkan tiga nilai fitness terbaik pada populasi.

5.6. Set $t \leftarrow t + 1$.

6. Kembalikan $\vec{X}_\alpha$ sebagai solusi terbaik akhir dan nilai *fitness* $f(\vec{X}_\alpha)$.

04

Aplikasi *Grey Wolf Optimizer* dalam Optimasi *Hyperparameter*

Optimasi *hyperparameter* merupakan tahap penting dalam pengembangan model pembelajaran mesin karena kualitas model sangat dipengaruhi oleh nilai hyperparameter yang digunakan selama proses pelatihan. Berbeda dari parameter model yang dipelajari langsung dari data, *hyperparameter* ditetapkan sebelum pelatihan dimulai dan berfungsi mengendalikan perilaku model. Pemilihan hyperparameter yang kurang tepat dapat menyebabkan model mengalami *underfitting*, *overfitting*, pelatihan yang lambat, atau kemampuan generalisasi yang buruk pada data baru. Oleh sebab itu, optimasi hyperparameter dipahami sebagai upaya untuk menemukan konfigurasi hyperparameter yang memberikan performa terbaik bagi model pada tugas tertentu.

Dalam praktiknya, pencarian *hyperparameter* yang baik bukanlah tugas yang sederhana. Ruang pencarian *hyperparameter* sering kali berdimensi banyak, melibatkan variabel kontinu, diskrit, bahkan kategorikal, serta memiliki hubungan antarkomponen yang tidak selalu linier. Pada model SVM, performa dipengaruhi oleh hyperparameter seperti C, γ, dan jenis kernel (Siswantoro, 2024; Siswantoro et al., 2024). Pada CNN, ruang pencarian menjadi lebih

kompleks lagi karena dapat melibatkan jumlah *layer*, ukuran kernel, jumlah filter, ukuran *pooling*, *dropout rate*, *learning rate*, *optimizer*, *batch size*, dan *epoch* (Jain et al., 2025; Mohakud and Dash, 2022a, 2022b). Akibatnya, penentuan *hyperparameter* secara manual melalui *trial and error* menjadi mahal, memakan waktu, dan tidak menjamin bahwa konfigurasi yang diperoleh mendekati optimum.

Metode pencarian konvensional seperti *grid search* dan *random search* memang dapat digunakan untuk mengatasi masalah ini, tetapi keduanya memiliki keterbatasan. *Grid search* mengevaluasi kombinasi *hyperparameter* secara sistematis, namun menjadi sangat tidak efisien ketika ruang pencarian membesar. *Random search* lebih fleksibel, tetapi tetap berisiko melewatkan kombinasi yang baik karena proses pencarian sangat bergantung pada sampel acak yang dihasilkan. Dalam kondisi seperti ini, optimasi *hyperparameter* menjadi masalah optimasi yang sangat sesuai ditangani oleh pendekatan metaheuristik, karena metaheuristik tidak perlu memeriksa semua kombinasi secara menyeluruh, melainkan menelusuri ruang solusi secara bertahap melalui sekumpulan agen pencari yang terus diperbarui (Siswantoro, 2024).

GWO merupakan salah satu metaheuristik berbasis *swarm intelligence* yang relevan untuk optimasi *hyperparameter* karena strukturnya sederhana, jumlah parameter internalnya sedikit, dan mekanisme pencariannya relatif mudah diimplementasikan. GWO meniru perilaku sosial dan strategi berburu serigala abu-abu melalui tiga agen elit, yaitu serigala α, β, dan δ, yang memandu agen lain menuju wilayah solusi terbaik. Dalam konteks optimasi *hyperparameter*, ketiga agen elit ini dapat dipandang sebagai tiga konfigurasi *hyperparameter* terbaik yang ditemukan pada suatu iterasi, lalu digunakan sebagai acuan untuk membentuk kandidat konfigurasi berikutnya. Dengan demikian, GWO menyediakan mekanisme pencarian yang intuitif sekaligus efektif untuk menyusuri ruang *hyperparameter* yang kompleks.

Walaupun demikian, GWO standar tidak sepenuhnya bebas dari keterbatasan. Pada ruang *hyperparameter* yang sangat besar,

algoritma dapat mengalami stagnasi lokal atau konvergensi prematur. Oleh sebab itu, selain menggunakan GWO standar (Ahmed et al., 2023; Mohakud and Dash, 2022b; Siswantoro, 2024; Wang et al., 2024), sejumlah peneliti mengembangkan varian GWO yang diperkaya dengan *neighborhood strategy* (Mohakud and Dash, 2022a), *random walk* (Siswantoro and Yuhana, 2023), serta *chaos* dan *differential evolution mutation* (Jain et al., 2025) untuk memperkuat kemampuan pencarian *hyperparamter* terbaik. Meski demikian, keterbatasan tersebut tidak mengurangi relevansi GWO sebagai fondasi. Justru karena formulasi dasarnya sederhana dan modular, GWO mudah diperluas sehingga tetap kompetitif untuk optimasi *hyperparameter* pada berbagai model pembelajaran mesin. Rangkuman aplikasi GWO dan variannya dalam optimasi *hyperparameter* dapat dilihat pada Tabel 4.1.

Tabel 4.1. Rangkuman aplikasi GWO dan variannya dalam optimasi *hyperparameter*.

Tugas	**Varian GWO**	**Model**	**Hyperparameter**	**Fungsi Objektif**
Deteksi kanker cervic (Jain et al., 2025)	MGWO	CNN	jumlah filter, ukuran kernel, *dropout rate, learning rate*	*cross entropy*
Klasifikasi citra buah (Siswantoro, 2024)	GWO standar	SVM	ambang untuk *variant threshold pemilihan fitur*, koefisien regularisasi, koefisien kernel	$1 - akurasi$
Deteksi kecacatan perangkat lunak (Siswantoro et al., 2024)	RW-GWO	SVM	jumlah komponen PCA, koefisien regularisasi, koefisien kernel, kernel	$-akurasi$

Tugas	Varian GWO	Model	Hyperparameter	Fungsi Objektif
Deteksi keruskan *Wheelset Bearings* (Wang et al., 2024)	GWO standar	SVM	koefisien regularisasi, koefisien kernel, orde kernel polinomial	*akurasi*
Deteksi anomali PLTS (Ahmed et al., 2023)	GWO standar	SVM	koefisien regularisasi, koefisien kernel	*akurasi*
Segmentasi citra kanker kulit (Mohakud and Dash, 2022a)	EN-GWO	FCEDN	jumlah filter, ukuran kernel, *dropout rate*	*Jaccard coefficient*
Deteksi kanker kulit (Mohakud and Dash, 2022b)	GWO standar	CNN	jumlah lapisan konvolusi, jumlah filter, ukuran kernel, ukuran pooling, *dropout rate*	*akurasi*

Secara umum, tahapan optimasi *hyperparameter* menggunakan GWO dimulai dengan mengodekan setiap kombinasi *hyperparameter* sebagai vektor solusi, sehingga satu serigala merepresentasikan satu kandidat konfigurasi hyperparameter. Setelah itu, populasi awal serigala dibangkitkan dalam ruang pencarian yang telah ditentukan untuk setiap *hyperparameter*. Setiap kandidat *hyperparameter* kemudian dievaluasi menggunakan fungsi objektif yang sesuai dengan model dan tugas yang dihadapi. Evaluasi biasanya dilakukan dengan melatih dan menguji model pembelajaran mesin dengan kandidat *hyperparameter* pada data latih dan data uji, lalu menghitung nilai fungsi objektif berdasarkan performa model.

Berdasarkan nilai fungsi objektif tersebut, tiga solusi terbaik ditetapkan sebagai serigala α, β, dan δ, lalu digunakan untuk memandu pembaruan posisi serigala lain sesuai mekanisme GWO. Proses evaluasi dan pembaruan posisi ini dilakukan secara iteratif hingga jumlah iterasi maksimum tercapai, dan solusi terbaik pada akhir proses dipilih sebagai konfigurasi *hyperparameter* optimal. (Mohakud and Dash, 2022b).

4.1 Representasi Serigala dalam Optimasi *Hyperparameter*

Untuk menentukan kombinasi *hyperparameter* terbaik dari sebuah model pembelajaran mesin, GWO bekerja dengan gagasan sederhana namun sangat kuat yaitu serigala bukan lagi seekor hewan tapi berupa vektor dengan komponen *hyperparameter* dari model pembelajaran mesin. Jika suatu model memiliki D hyperparameter yang ingin dioptimasi, maka posisi seekor serigala dapat dinyatakan sebagai vektor berdimensi D seperti pada persamaan (4.1),

$$\vec{\boldsymbol{X}}(t) = (x_1, x_2, \dots, x_D) \tag{4.1}$$

dengan setiap komponen merepresentasikan satu *hyperparameter*. Dengan demikian, serigala tidak lagi dimaknai sebagai agen pada ruang fisik, melainkan sebagai satu kandidat konfigurasi yang dapat langsung digunakan untuk membangun atau melatih model. Representasi ini mirip dengan representasi solusi pada optimasi kontinu, hanya saja makna setiap dimensi ditentukan oleh jenis *hyperparameter* yang dioptimasi.

Tahapan awal dalam optimasi *hyperparameter* menggunakan GWO adalah pengodean *hyperparameter* menjadi vektor posisi. Proses pengodean *hyperparameter* adalah mengubah *hyperparameter* menjadi vektor posisi $\vec{\boldsymbol{X}}(t)$. Sebagai contoh pada optimasi *hyperparameter* Support Vector Machine (SVM), vektor posisi $\vec{\boldsymbol{X}}(t)$ berisi koefisien regularisasi C, koefisien kernel γ, jenis kernel K, atau orde kernel polinomial r. Jika sebelum klasifikasi terdapat tahap pemilihan fitur atau pengurangan dimensi maka parameter dari tahap tersebut juga dapat dioptimasi menggunakan GWO (Siswantoro,

2024; Siswantoro et al., 2024; Wang et al., 2024). Selain *hyperparameter* model SVM, GWO juga diaplikasikan untuk mengoptimasi *hyperparameter* dari model *deep learning* seperti model *Convolutional Neural Network* (CNN). Untuk model CNN vektor posisi $\vec{\boldsymbol{X}}(t)$ dapat berisi jumlah lapisan konvolusi, jumlah filter, ukuran kernel, ukuran pooling, *dropout rate*, maupun *learning rate* (Jain et al., 2025; Mohakud and Dash, 2022b, 2022a).

Representasi tersebut memiliki implikasi metodologis yang penting. Dalam optimasi *hyperparameter*, perubahan posisi serigala dimaknai sebagai perubahan konfigurasi model. Artinya, ketika posisi serigala bergeser, yang sesungguhnya berubah adalah kombinasi nilai *hyperparameter* yang akan digunakan pada model. Satu langkah pembaruan posisi dapat berarti *learning rate* menjadi lebih kecil, *dropout rate* meningkat, jumlah filter bertambah, atau parameter regularisasi SVM berubah. Karena itu, kualitas solusi sangat bergantung pada bagaimana ruang *hyperparameter* didefinisikan, termasuk rentang nilai setiap dimensi serta jenis variabel yang terlibat, apakah kontinu, diskrit, atau kategorikal.

Pada tahap inisialisasi, populasi serigala dibangkitkan secara acak di dalam rentang nilai yang telah ditentukan untuk setiap hyperparameter. Misalnya, *learning rate* dapat dibatasi pada interval tertentu, jumlah filter pada interval diskrit tertentu, ukuran kernel pada himpunan nilai yang diizinkan, dan dropout rate pada interval antara 0 dan 1. Pada CNN, strategi ini menghasilkan sekumpulan konfigurasi awal yang beragam. Pada SVM, rentang nilai untuk C dan γ biasanya ditentukan sesuai kebutuhan eksperimen. Inisialisasi semacam ini penting karena menentukan wilayah awal pencarian. Jika rentang terlalu sempit, solusi yang baik bisa tidak pernah dicapai. Jika terlalu lebar, pencarian dapat menjadi lebih lambat dan kurang efisien.

4.2 Fungsi Objektif dalam Optimasi *Hyperparameter*

Dalam optimasi *hyperparameter*, fungsi objektif digunakan untuk menilai seberapa baik satu konfigurasi *hyperparameter* ketika diterapkan pada model. Secara umum, fungsi objektif pada masalah ini lebih sederhana dibandingkan pada pemilihan fitur, karena sasaran utamanya biasanya adalah memaksimalkan performa model atau meminimalkan kesalahan prediksi. Dengan kata lain, setiap serigala akan dinilai berdasarkan hasil model yang dibangun menggunakan *hyperparameter* yang direpresentasikannya. Oleh sebab itu, hubungan antara GWO dan model pembelajaran mesin pada optimasi *hyperparameter* bersifat sangat langsung, yaitu diawali dengan konfigurasi *hyperparameter* dan diakhiri dengan perhitungan nilai fungsi objektif.

Evaluasi fungsi objektif pada optimasi *hyperparameter* berbasis GWO dilakukan dengan mengaitkan setiap posisi serigala sebagai satu konfigurasi *hyperparameter* terhadap kinerja model pada data yang terpisah. Pertama, dataset dibagi menjadi data latih dan data uji. Selanjutnya, untuk setiap serigala, nilai-nilai pada vektor posisinya diinterpretasikan sebagai *hyperparameter* model, kemudian model dilatih menggunakan data latih dengan konfigurasi *hyperparameter* tersebut sehingga diperoleh model terlatih. Setelah itu, model dievaluasi menggunakan data uji untuk menghitung nilai fungsi objektif.

Bentuk fungsi objektif yang paling umum adalah berbasis akurasi pada masalah klasifikasi. Beberapa peneliti menggunakan akurasi klasifikasi secara langsung, yang didefinisikan pada persamaan (4.2), sebagai fungsi objektif yang dimaksimumkan (Mohakud and Dash, 2022b; Wang et al., 2024). Selain itu, karena banyak implementasi GWO dirumuskan sebagai masalah minimisasi, akurasi sering ditransformasikan menjadi bentuk ekuivalen seperti (Siswantoro, 2024) atau $-akurasi$ (Siswantoro et al., 2024) sebagai fungsi objektif.

$$fitness = akurasi = \frac{Banyaknya\ data\ yang\ tepat\ diklasifikasikan}{Banyaknya\ data\ uji} \quad (4.2)$$

$$fitness = 1 - akurasi \tag{4.3}$$

$$fitness = -akurasi \tag{4.4}$$

Selain akurasi, *cross entropy* pada persamaan (4.5), juga digunakan sebagai fungsi objektif pada optimasi *hyperparameter* model CNN (Jain et al., 2025),

$$fitness = -\sum_{i=1}^{N} y_i \log p_i \tag{4.5}$$

dengan N menyatakan jumlah kelas, y_i adalah label sebenarnya pada kelas ke-i, dan p_i adalah probabilitas prediksi model untuk kelas ke-i. Pada klasifikasi, nilai y_i biasanya berbentuk *one-hot encoding*, sehingga hanya kelas yang benar yang bernilai 1, sedangkan kelas lainnya bernilai 0. Akibatnya, penjumlahan tersebut pada dasarnya hanya mengambil nilai $-\log(p_i)$ dari kelas yang benar. Jika model memberikan probabilitas tinggi pada kelas yang benar, maka nilai *cross entropy* akan kecil, sebaliknya jika probabilitas untuk kelas yang benar rendah, maka nilai cross entropy akan besar. Sehingga pada kasus optimasi *hyperparameter* fungsi ini menjadi fungsi objektif yang harus diminimumkan.

Pada optimasi *hyperparameter Fully Convolution Encoder Decoder Network* (FCEDN) untuk segmentasi citra (Mohakud and Dash, 2022a), *Jaccard coefficient* pada persamaan (4.6) digunakan sebagai fungsi objektif,

$$fitness = \frac{1}{N}\sum_{i=1}^{N} \frac{\varepsilon + \sum_{i=1}^{r}\sum_{j=1}^{c} y(i,j)\hat{y}(i,j)}{\varepsilon + \sum_{i=1}^{r}\sum_{j=1}^{c} y(i,j) + \sum_{i=1}^{r}\sum_{j=1}^{c} \hat{y}(i,j) - \sum_{i=1}^{r}\sum_{j=1}^{c} y(i,j)\hat{y}(i,j)} \tag{4.6}$$

dengan N menyatakan banyaknya citra, r dan c masing-masing menyatakan banyaknya baris dan kolom piksel pada citra. $y(i,j)$ adalah nilai *ground truth* pada piksel (i,j), dan $\hat{y}(i,j)$ adalah nilai prediksi model pada piksel yang sama. Serta ε adalah parameter *smoothing* yang dibangkitkan secara random pada interval (0,1). Karena nilai *Jaccard coefficient* dihitung untuk setiap citra lalu dirata-ratakan terhadap N citra, maka nilai akhir pada persamaan (4.6)

merepresentasikan kualitas segmentasi model secara keseluruhan pada satu dataset evaluasi. Semakin tinggi rata-rata *Jaccard coefficient*, semakin baik *hyperparameter* yang digunakan oleh model segmentasi tersebut. Itulah sebabnya, pada optimasi *hyperparameter* FCEDN, *Jaccard coefficient* dimaksimumkan sebagai fungsi objektif. Detail algoritma pemilihan fitung menggunakan GWO pada masalah klasifikasi dapat dilihat pada Algoritma 4.1.

Algoritma 4.1. Optimasi *hyperparameter* menggunakan GWO

Input:

Dataset X, label kelas y, ukuran populasi serigala N, jumlah iterasi maksimum T, model pembelajaran mesin yang akan dioptimasi, jumlah hyperparameter D, batas bawah $\overrightarrow{lb}$, dan batas atas $\overrightarrow{ub}$.

Output:

Konfigurasi hyperparameter terbaik $\vec{X}_\alpha$ dan nilai *fitness* terbaik.

Prosedur:

1. Bangkitkan populasi awal sebanyak N serigala. Setiap serigala direpresentasikan sebagai vektor berdimensi D, dengan setiap komponen merepresentasikan satu *hyperparameter* model.
2. Hitung nilai *fitness* setiap serigala menggunakan langkah-langkah berikut:
   2.1. Bagi dataset X dan label kelas y menjadi data latih dan data uji.
   2.2. Interpretasikan vektor posisi serigala sebagai satu konfigurasi hyperparameter model.
   2.3. Latih model menggunakan data latih dengan konfigurasi hyperparameter tersebut.
   2.4. Uji model menggunakan data uji.

2.5. Hitung *fitness* menggunakan salah satu fungsi pada persamaan (4.2) – (4.6)

3. Tentukan tiga serigala terbaik dalam populasi sebagai $\vec{X}_\alpha$, $\vec{X}_\beta$, dan $\vec{X}_\delta$.
4. Tetapkan iterasi awal $t \leftarrow 1$.
5. Hitung nilai a, $\vec{A}$, dan $\vec{C}$ menggunakan persamaan (2.5), (2.3), dan (2.4).
6. Selama $t \leq T$, lakukan langkah-langkah berikut:

   6.1. Untuk setiap serigala ke-$i = 1,2, \ldots, N$, lakukan:

      a. Perbarui posisi serigala ke-i menggunakan persamaan (2.12).
      b. Jika posisi baru berada di luar batas pencarian, lakukan penyesuaian agar tetap berada pada rentang $[\overrightarrow{lb}, \overrightarrow{ub}]$.

   6.2. Hitung kembali nilai *fitness* setiap serigala menggunakan langkah 2.

   6.3. Perbarui nilai a, $\vec{A}$, dan $\vec{C}$ menggunakan persamaan (2.5), (2.3), dan (2.4).

   6.4. Perbarui $\vec{X}_\alpha$, $\vec{X}_\beta$, dan $\vec{X}_\delta$ berdasarkan tiga nilai *fitness* terbaik pada populasi.

   6.5. Set $t \leftarrow t + 1$.

7. Kembalikan konfigurasi *hyperparameter* terbaik $\vec{X}_\alpha$ dan nilai *fitness* terbaik.

4.3 Studi Kasus Optimasi *Hyperparameter* SVM untuk Klasifikasi Citra Buah-Buahan

Studi kasus ini membahas penerapan GWO untuk mengoptimasi *hyperparameter Support Vector Machine* (SVM) pada masalah klasifikasi citra buah-buahan (Siswantoro, 2024). Studi kasus ini penting karena menunjukkan bagaimana konsep optimasi *hyperparameter* yang telah dijelaskan sebelumnya dapat diterapkan pada persoalan nyata dalam bidang visi komputer dan pembelajaran mesin. Klasifikasi citra buah-buahan merupakan salah satu contoh penerapan pembelajaran mesin dalam bidang visi komputer. Permasalahan ini banyak ditemukan pada bidang pertanian, industri pangan, retail, dan sistem rekomendasi konsumsi makanan. Sistem klasifikasi berbasis citra dapat membantu proses identifikasi jenis buah secara otomatis berdasarkan karakteristik visualnya. Dengan cara ini, proses pengenalan buah tidak harus selalu dilakukan secara manual oleh manusia.

Meskipun terlihat sederhana, klasifikasi citra buah-buahan memiliki tantangan yang cukup besar. Setiap jenis buah dapat memiliki variasi warna, bentuk, ukuran, tekstur, tingkat kematangan, dan kondisi permukaan yang berbeda. Selain itu, citra yang diambil dari lingkungan nyata juga dapat dipengaruhi oleh pencahayaan, sudut pengambilan gambar, warna latar belakang, bayangan, jumlah objek, dan posisi buah. Variasi tersebut membuat model klasifikasi harus mampu mengenali pola visual yang relevan, bukan hanya menghafal kondisi citra tertentu.

Dalam pendekatan pembelajaran mesin tradisional, citra tidak langsung digunakan sebagai input dalam bentuk piksel mentah. Citra terlebih dahulu diubah menjadi vektor fitur. Vektor fitur tersebut berisi informasi penting yang merepresentasikan karakteristik visual citra, misalnya warna, tekstur, dan tepi. Setelah fitur diperoleh, model klasifikasi dapat dilatih untuk membedakan setiap kelas buah berdasarkan pola fitur tersebut. Pada studi kasus ini, model klasifikasi yang digunakan adalah Support Vector Machine (SVM). SVM dipilih karena memiliki kemampuan yang baik dalam

menyelesaikan masalah klasifikasi, terutama pada data berdimensi tinggi.

Performa SVM sangat dipengaruhi oleh pemilihan *hyperparameter*. Pada SVM dengan kernel *radial basis function* (RBF), dua *hyperparameter* penting yang perlu ditentukan adalah parameter regularisasi C dan koefisien kernel γ. Nilai C mengontrol keseimbangan antara margin dan kesalahan klasifikasi, sedangkan γ mengatur tingkat pengaruh setiap sampel terhadap bentuk batas keputusan. Selain *hyperparameter* SVM, proses pemilihan fitur juga berpengaruh terhadap performa model. Tidak semua fitur hasil ekstraksi memiliki kontribusi yang sama terhadap klasifikasi. Sebagian fitur dapat bersifat informatif, sedangkan fitur lainnya dapat bersifat lemah, redundan, atau kurang relevan. Oleh karena itu, studi kasus ini menggunakan *low variance feature selection* (LVFS) untuk menghapus fitur dengan varians rendah. Nilai ambang pada LVFS juga diperlakukan sebagai *hyperparameter* yang perlu dioptimasi.

4.3.1 Dataset Citra Buah-Buahan

Studi kasus ini menggunakan tiga dataset citra buah-buahan. Penggunaan tiga dataset bertujuan untuk menguji kemampuan metode pada kondisi data yang berbeda. Setiap dataset memiliki karakteristik citra, jumlah kelas, jumlah sampel, dan tingkat kesulitan yang berbeda. Dataset pertama adalah dataset citra Ubaya-IFDS3000 (Siswantoro et al., 2020b). Dataset ini berisi 3000 citra buah-buahan Indonesia yang terbagi ke dalam 15 kelas dengan 200 citra pada setiap kelas. Kelas buah dalam dataset ini mencakup kedondong, alpukat, buah naga, duku, durian, jambu biji, manggis, jeruk Pacitan, kesemek, nanas, salak, sawo, jeruk siam, sirsak, dan belimbing. Citra diambil dalam ruang warna RGB (*Red*, *Green*, *Blue*) menggunakan kamera Canon EOS Kiss X6i. Ukuran citra pada dataset ini adalah 2592×1456 piksel dengan resolusi 72 dpi dan disimpan dalam format JPEG. Dataset ini juga menggunakan lima warna latar belakang, yaitu pink, putih, biru muda, hijau muda, dan kuning muda. Jumlah dan posisi objek pada setiap citra dibuat bervariasi. Selain itu, citra diambil pada dua tingkat pencahayaan, yaitu 160 lumen dan 1050 lumen. Sudut

kamera juga dibuat bervariasi, yaitu 0° dan 45°. Variasi latar, pencahayaan, sudut kamera, jumlah objek, dan bayangan membuat dataset ini lebih realistis untuk menguji model klasifikasi. Contoh citra buah-buahan dalam dataset Ubaya-IFDS3000 dapat dilihat pada Gambar 4.1.

Gambar 4.1 Contoh citra buah-buahan dalam dataset Ubaya-IFDS3000.

Dataset kedua adalah dataset citra Ubaya-IFDS5000 (Siswantoro, 2024). Dataset ini berisi 5000 citra dari 25 jenis buah Indonesia dengan 200 citra pada setiap kelas. Jenis buah yang digunakan meliputi apel ana, belimbing wuluh, blewah, mentimun, jambu air hijau, jambu mete, kenitu, semangka lonjong, apel manalagi, matoa, melon, murbei, buah siwalan, pepaya, markisa, delima, jeruk bali, rambutan, stroberi, srikaya, timun krai, timun suri, tomat, semangka, dan jambu air. Citra pada Ubaya-IFDS5000 diambil menggunakan kamera Canon EOS 80D dalam ruang warna RGB. Ukuran citra pada dataset ini adalah 2976×1984 piksel dengan resolusi 72 dpi dan disimpan dalam format JPEG. Dataset ini lebih kompleks daripada Ubaya-IFDS3000 karena jumlah kelasnya lebih banyak. Selain itu, beberapa kelas memiliki kemiripan visual, baik dari sisi warna, bentuk, maupun tekstur. Kondisi ini membuat proses klasifikasi menjadi lebih sulit. Contoh citra buah-buahan dalam dataset Ubaya-IFDS5000 dapat dilihat pada Gambar 4.2.

Gambar 4.2. Contoh citra buah-buahan dalam dataset Ubaya-IFDS5000.

Dataset ketiga adalah dataset citra *Supermarket Produce* (Rocha et al., 2010). Dataset ini berisi 2633 citra buah dan sayuran yang terbagi ke dalam 15 kelas. Jumlah citra pada setiap kelas dibuat bervariasi, yaitu antara 75 sampai 264 citra. Kelas yang digunakan meliputi *agata potato*, *asterix potato*, *cashew*, *diamond peach*, *fuji apple*, *granny smith apple*, *honeydew melon*, *kiwi*, *nectarine*, *onion*, *orange*, *plum*, *spanish pear*, *taiti lime*, dan *watermelon*. Citra diambil menggunakan kamera Canon PowerShot P1 dengan ukuran 1024×768 piksel dalam ruang warna RGB. Dataset *Supermarket Produce* memiliki karakteristik yang berbeda dari dua dataset Ubaya. Dataset ini berisi citra dengan latar belakang putih atau bening, pencahayaan yang bervariasi, jumlah objek yang berbeda, pose yang beragam, dan sebagian objek yang tertutup plastik. Beberapa citra juga mengandung refleksi cahaya, bayangan, atau objek yang tertutup sebagian. Kondisi tersebut membuat dataset ini relevan untuk menguji kemampuan model pada citra yang lebih mendekati kondisi

nyata. Contoh citra dalam dataset *Supermarket Produce* dapat dilihat pada Gambar 4.3.

Gambar 4.3. Contoh citra dalam dataset *Supermarket Produce*.

4.3.2 Ekstraksi Fitur

Ekstraksi fitur merupakan tahap untuk mengubah citra menjadi representasi numerik. Representasi ini diperlukan karena model seperti SVM tidak memproses citra secara langsung sebagai gambar, tetapi sebagai vektor fitur. Oleh karena itu, kualitas fitur sangat menentukan kemampuan model dalam membedakan satu kelas buah dengan kelas lainnya. Pada studi kasus ini, fitur yang digunakan berasal dari MPEG-7 *visual descriptor* (Manjunath et al., 2002). Fitur MPEG-7 dipilih karena mampu merepresentasikan karakteristik warna dan tekstur citra. Lima jenis fitur yang digunakan adalah *color layout descriptor* (CLD), *color structure descriptor* (CSD), *scalable color descriptor* (SCD), *edge histogram descriptor* (EHD), dan *homogeneous texture descriptor* (HTD). Kelima fitur tersebut diekstraksi langsung dari seluruh piksel pada citra. Dengan demikian, proses klasifikasi tidak memerlukan segmentasi objek terlebih dahulu.

Penggunaan fitur tanpa segmentasi menjadi salah satu keunggulan penting. Pada banyak kasus klasifikasi citra buah, segmentasi digunakan untuk memisahkan objek buah dari latar belakang. Namun, segmentasi dapat menjadi sulit ketika citra

memiliki bayangan, refleksi cahaya, warna latar yang beragam, atau objek yang saling berdekatan. Dengan menggunakan fitur MPEG-7 secara langsung dari seluruh citra, proses klasifikasi menjadi lebih sederhana dan lebih mudah diterapkan.

4.3.2.1 Color Layout Descriptor

CLD merupakan fitur yang digunakan untuk menggambarkan distribusi warna pada ruang spasial citra. Fitur ini tidak hanya melihat warna apa saja yang muncul dalam citra, tetapi juga memperhatikan bagaimana warna tersebut tersebar pada bagian-bagian tertentu dari citra. Dengan demikian, CLD dapat menangkap informasi tata letak warna secara global. Informasi ini penting karena beberapa objek dapat memiliki warna yang sama, tetapi memiliki susunan atau persebaran warna yang berbeda. CLD diekstraksi dalam ruang warna YCbCr (*Luma*, *Chroma Blue*, *Chroma Red*), yaitu ruang warna yang memisahkan informasi kecerahan dan informasi warna. Komponen Y merepresentasikan tingkat kecerahan citra. Komponen Cb merepresentasikan perbedaan warna terhadap kanal biru. Komponen Cr merepresentasikan perbedaan warna terhadap kanal merah. Pemisahan ini membuat informasi intensitas dan warna dapat dianalisis secara lebih terstruktur.

Proses ekstraksi CLD dimulai dengan membagi citra menjadi 64 blok dengan ukuran yang sama. Pembagian ini biasanya membentuk susunan 8×8 blok. Setiap blok mewakili satu wilayah spasial pada citra. Setelah citra dibagi menjadi beberapa blok, nilai rata-rata intensitas piksel dihitung pada setiap blok untuk masing-masing kanal Y, Cb, dan Cr. Hasil dari proses ini adalah tiga citra kecil berukuran 8×8, masing-masing untuk kanal Y, Cb, dan Cr. Citra kecil ini merepresentasikan ringkasan distribusi warna dari citra asli. Tahap berikutnya adalah mentransformasi citra kecil tersebut ke domain frekuensi menggunakan *Discrete Cosine Transform* (DCT). Transformasi ini bertujuan untuk mengubah informasi warna dari domain spasial menjadi koefisien frekuensi. Koefisien frekuensi rendah biasanya menyimpan informasi utama tentang distribusi warna secara global, sedangkan koefisien frekuensi tinggi lebih

banyak merepresentasikan perubahan warna yang detail. Karena CLD menekankan informasi tata letak warna secara umum, koefisien frekuensi rendah menjadi bagian yang lebih penting.

Setelah DCT dilakukan, koefisien hasil transformasi dipilih menggunakan *zigzag scanning* dari frekuensi rendah menuju frekuensi yang lebih tinggi. Koefisien yang telah dipilih kemudian melalui proses kuantisasi. Kuantisasi bertujuan untuk menyederhanakan nilai koefisien agar representasi fitur menjadi lebih kompak dan efisien. Proses ini juga membantu mengurangi ukuran data fitur yang akan digunakan oleh model pembelajaran mesin. Hasil akhir dari proses ekstraksi tersebut adalah vektor fitur CLD sepanjang 120 elemen. Vektor ini terdiri atas 64 elemen dari kanal Y, 28 elemen dari kanal Cb, dan 28 elemen dari kanal Cr.

4.3.2.2 Color Structure Descriptor

CSD merupakan fitur yang menggambarkan susunan warna pada suatu citra, baik dari sisi persebaran warna secara lokal pada area tertentu maupun distribusi warna secara keseluruhan. Berbeda dari histogram warna biasa yang hanya menghitung jumlah kemunculan warna secara global, CSD juga mempertimbangkan posisi dan keberadaan warna pada wilayah lokal citra. Sehingga fitur ini lebih sensitif dalam menangkap karakteristik visual tertentu yang mungkin tidak terlihat jika hanya menggunakan histogram warna konvensional. CSD diekstraksi dalam ruang warna HMMD (*Hue, Max, Min, Diff*). Ruang warna ini merepresentasikan informasi warna berdasarkan *hue* atau corak warna, nilai maksimum komponen warna, nilai minimum komponen warna, serta selisih antara nilai maksimum dan minimum tersebut.

Proses ekstraksi CSD dilakukan dengan memindai seluruh area citra menggunakan elemen penstruktur berukuran 8×8 piksel. Elemen ini bergerak melintasi citra untuk mengamati kemunculan warna pada setiap area lokal. Hasil dari proses tersebut adalah histogram dengan 256 bin. Setiap bin merepresentasikan kelompok warna tertentu. Nilai pada setiap bin diperbarui selama proses

pemindaian dengan cara menghitung berapa kali warna tertentu muncul di dalam elemen penstruktur 8×8. Dengan pendekatan ini, CSD tidak hanya menyimpan informasi tentang seberapa sering suatu warna muncul dalam citra, tetapi juga menangkap bagaimana warna tersebut tersebar pada bagian-bagian lokal citra. Oleh karena itu, fitur ini dapat memberikan representasi warna yang lebih kaya untuk mendukung proses klasifikasi citra.

4.3.2.3 Scalable Color Descriptor

SCD merupakan fitur yang digunakan untuk merepresentasikan distribusi warna dalam citra. Fitur ini bekerja berdasarkan prinsip histogram warna, yaitu menghitung frekuensi kemunculan warna tertentu pada suatu citra. Berbeda dari representasi warna yang hanya menyimpan informasi warna secara langsung, SCD dirancang agar informasi warna dapat disimpan dalam bentuk yang lebih ringkas, efisien, dan tetap mempertahankan karakteristik utama citra. SCD diekstraksi dalam ruang warna HSV (*Hue*, *Saturation*, *Value*). Komponen H merepresentasikan jenis atau corak warna, misalnya merah, hijau, atau biru. Komponen S merepresentasikan tingkat kejenuhan warna, yaitu seberapa kuat atau murni warna tersebut. Komponen V merepresentasikan tingkat kecerahan warna. Ruang warna HSV sering digunakan dalam analisis citra karena lebih dekat dengan cara manusia membedakan warna dibandingkan ruang warna RGB.

Proses ekstraksi SCD dimulai dengan membentuk histogram warna dari citra dalam ruang warna HSV. Histogram tersebut memiliki 256 bin. Setiap bin merepresentasikan kelompok warna tertentu. Untuk memperoleh 256 bin tersebut, ruang warna HSV dikuantisasi menjadi beberapa level. Kanal H dikuantisasi menjadi 16 level, kanal S menjadi 4 level, dan kanal V menjadi 4 level. Dengan cara ini, warna-warna dalam citra dapat direpresentasikan dalam bentuk histogram yang lebih terstruktur. Setelah histogram terbentuk, histogram tersebut dinormalisasi. Normalisasi dilakukan agar nilai histogram tidak terlalu dipengaruhi oleh ukuran citra atau jumlah piksel. Melalui normalisasi, nilai pada histogram lebih

merepresentasikan proporsi kemunculan warna, bukan sekadar jumlah absolut piksel. Tahap ini penting karena citra dengan ukuran berbeda tetap dapat dibandingkan menggunakan representasi fitur yang konsisten.

Histogram yang telah dinormalisasi kemudian dipetakan ke dalam representasi bilangan 4-bit. Pemetaan ini bertujuan untuk membuat representasi fitur menjadi lebih ringkas. Selain itu, proses ini juga memberikan bobot yang lebih sesuai pada nilai histogram kecil yang memiliki peluang kemunculan lebih tinggi. Dengan demikian, informasi warna yang penting tetap dapat dipertahankan meskipun ukuran representasi dikurangi.

Tahap berikutnya adalah transformasi menggunakan Haar Transform. Haar Transform merupakan teknik transformasi yang dapat mengubah histogram warna menjadi representasi koefisien yang lebih kompak. Transformasi ini membantu mengelompokkan informasi warna ke dalam bentuk yang lebih efisien, sehingga fitur dapat digunakan secara bertingkat sesuai kebutuhan. Karena sifat inilah fitur ini disebut scalable color. Artinya, representasi warna dapat digunakan dalam tingkat detail yang berbeda, mulai dari representasi yang lebih ringkas sampai representasi yang lebih lengkap.

4.3.2.4 Edge Histogram Descriptor

EHD merupakan fitur tekstur yang digunakan untuk menggambarkan distribusi tepi pada citra. Tepi merupakan bagian penting dalam citra karena menunjukkan adanya perubahan intensitas piksel yang cukup besar. Perubahan tersebut biasanya muncul pada batas objek, perubahan permukaan, lipatan, bayangan, atau perbedaan warna yang tajam. Dalam citra buah-buahan, informasi tepi dapat membantu mengenali bentuk, kontur, tekstur permukaan, dan pola visual lokal dari suatu objek. EHD tidak hanya menghitung jumlah tepi pada citra secara keseluruhan, tetapi juga memperhatikan lokasi kemunculan tepi. Dengan demikian, fitur ini mampu menggambarkan bagaimana tepi tersebar pada bagian-bagian tertentu dalam citra.

Informasi ini penting karena dua citra dapat memiliki jumlah tepi yang hampir sama, tetapi distribusi tepinya berbeda. Misalnya, suatu buah dapat memiliki tepi kuat pada bagian kulit karena teksturnya kasar, sedangkan buah lain memiliki tepi yang lebih halus dan hanya muncul pada batas objek.

Proses ekstraksi EHD dimulai dengan membagi citra menjadi 4×4 blok besar yang tidak saling tumpang tindih. Dengan pembagian ini, citra memiliki 16 blok lokal. Setiap blok merepresentasikan satu wilayah spasial pada citra. Pembagian blok ini bertujuan agar informasi tepi tidak hanya dihitung secara global, tetapi juga dapat diketahui posisinya pada area tertentu. Pendekatan ini membuat EHD lebih informatif daripada fitur yang hanya menghitung tepi pada seluruh citra tanpa memperhatikan lokasi. Pada setiap blok, informasi tepi dihitung menggunakan beberapa detektor tepi. Detektor ini digunakan untuk mengenali arah dominan dari perubahan intensitas piksel. EHD mengelompokkan tepi ke dalam lima jenis, yaitu tepi vertikal, tepi horizontal, tepi diagonal 45°, tepi diagonal 135°, dan tepi isotropic. Tepi vertikal menunjukkan perubahan intensitas yang membentuk pola tegak. Tepi horizontal menunjukkan perubahan intensitas yang membentuk pola mendatar. Tepi diagonal 45° dan 135° menunjukkan perubahan intensitas yang mengikuti arah miring. Sementara itu, tepi isotropic menunjukkan pola perubahan intensitas yang tidak memiliki arah dominan tertentu atau menyebar secara relatif merata.

Setiap blok menghasilkan lima nilai histogram. Lima nilai tersebut merepresentasikan frekuensi atau kekuatan kemunculan lima jenis tepi pada blok yang bersangkutan. Karena citra dibagi menjadi 16 blok dan setiap blok menghasilkan lima bin, total fitur EHD adalah 80 bin. Vektor fitur sepanjang 80 elemen ini kemudian digunakan sebagai representasi distribusi tepi pada citra.

4.3.2.5 Homogeneous Texture Descriptor

HTD merupakan fitur tekstur yang digunakan untuk menggambarkan karakteristik pola tekstur pada citra. Fitur ini

menangkap informasi tentang arah, kekasaran, dan frekuensi pola yang muncul pada permukaan objek. Dalam citra buah-buahan, informasi tekstur penting karena beberapa jenis buah memiliki warna yang mirip, tetapi memiliki permukaan yang berbeda. Perbedaan tersebut dapat berupa kulit yang halus, kasar, berbintik, berserat, berduri, atau memiliki pola visual tertentu.

Tekstur pada citra tidak hanya dapat diamati dari nilai intensitas piksel secara langsung, tetapi juga dari pola perubahan intensitas pada domain frekuensi. Oleh karena itu, HTD bekerja dengan menganalisis citra pada ruang frekuensi dua dimensi. Pada tahap awal, ruang frekuensi dua dimensi dibagi menjadi 30 kanal. Pembagian ini dilakukan berdasarkan dua aspek, yaitu skala dan arah. Skala digunakan untuk menangkap pola tekstur dengan tingkat kekasaran yang berbeda. Arah digunakan untuk menangkap orientasi pola tekstur, misalnya pola yang cenderung horizontal, vertikal, atau miring.

Pembagian ruang frekuensi tersebut dilakukan dengan lima segmen octave pada arah radial dan enam segmen pada arah sudut. Segmen radial merepresentasikan tingkat frekuensi atau tingkat kekasaran tekstur. Frekuensi rendah biasanya berkaitan dengan pola yang lebih halus atau perubahan intensitas yang lambat. Frekuensi tinggi biasanya berkaitan dengan pola yang lebih kasar atau perubahan intensitas yang cepat. Sementara itu, segmen sudut digunakan untuk membedakan orientasi tekstur. Setiap segmen sudut memiliki interval 30 derajat. Kombinasi lima segmen radial dan enam segmen sudut menghasilkan 30 kanal frekuensi.

Setelah ruang frekuensi dibagi menjadi 30 kanal, proses berikutnya menggunakan *Gabor-filtered Fourier Transform*. *Fourier Transform* digunakan untuk mengubah informasi citra dari domain spasial ke domain frekuensi. Dengan transformasi ini, pola tekstur dapat dianalisis berdasarkan kandungan frekuensinya. Gabor filter kemudian digunakan untuk menangkap pola tekstur pada kanal frekuensi tertentu. Filter ini efektif untuk menganalisis tekstur karena

mampu memperhatikan informasi frekuensi dan orientasi secara bersamaan.

Pada setiap kanal, energi tekstur dihitung dari hasil filter. Energi ini menunjukkan seberapa kuat pola tekstur tertentu muncul pada citra. Jika suatu kanal memiliki nilai energi tinggi, berarti pola tekstur dengan arah dan skala yang sesuai dengan kanal tersebut muncul cukup kuat pada citra. Jika nilai energinya rendah, berarti pola tekstur tersebut tidak dominan. Dengan cara ini, HTD dapat menggambarkan karakteristik tekstur berdasarkan distribusi energi pada berbagai arah dan skala.

Dari setiap kanal, dua nilai dihitung, yaitu rata-rata energi dan deviasi energi. Rata-rata energi menunjukkan tingkat kekuatan umum pola tekstur pada kanal tertentu. Deviasi energi menunjukkan tingkat variasi atau penyebaran energi pada kanal tersebut. Karena terdapat 30 kanal dan setiap kanal menghasilkan dua nilai, maka diperoleh 60 bin fitur. Enam puluh bin ini merepresentasikan pola tekstur utama berdasarkan arah dan skala pada domain frekuensi.

Selain 60 bin tersebut, HTD juga menambahkan dua bin tambahan. Dua bin ini berasal dari rata-rata intensitas piksel dan standar deviasi intensitas piksel pada citra. Rata-rata intensitas memberikan informasi umum tentang tingkat kecerahan citra. Standar deviasi intensitas memberikan informasi tentang variasi tingkat kecerahan. Kedua informasi ini melengkapi representasi tekstur karena pola permukaan objek juga dapat dipengaruhi oleh sebaran intensitas piksel. Dengan demikian, total fitur HTD adalah 62 bin. Enam puluh bin berasal dari rata-rata dan deviasi energi pada 30 kanal frekuensi. Dua bin lainnya berasal dari rata-rata dan standar deviasi intensitas piksel.

4.3.2.6 Fusi Fitur

Selain menggunakan fitur secara individual, studi kasus ini juga menggunakan fusi fitur. Fusi fitur dilakukan dengan menggabungkan beberapa fitur MPEG-7 ke dalam satu vektor fitur. Strategi ini digunakan karena setiap fitur menangkap karakteristik

visual yang berbeda. Fitur warna membantu membedakan buah berdasarkan distribusi warna. Fitur tekstur dan tepi membantu membedakan buah yang memiliki permukaan, pola, atau struktur visual yang berbeda. Dengan menggabungkan beberapa fitur, representasi citra menjadi lebih kaya. Pada studi kasus ini fusi fitur dilakukan dengan menggabungkan beberapa kombinasi fitur warna dan tekstur dari MPEG-7. Pemilihan kombinasi fitur tersebut mengacu pada fitur-fitur yang telah digunakan dalam penelitian klasifikasi buah pada studi sebelumnya (Siswantoro et al., 2020a). Seluruh kombinasi fusi fitur yang digunakan dalam studi kasus ini disajikan pada Tabel 4.2

Tabel 4.2. Kombinasi fusi fitur yang digunakan dalam studi kasus.

i	Fusi Fitur (F_i)	i	Fusi Fitur (F_i)
1	CSD	13	CSD+CLD+HTD
2	SCD	14	CSD+CLD+EHD
3	CSD+SCD	15	CSD+HTD+EHD
4	CSD+CLD	16	SCD+CLD+HTD
5	CSD+HTD	17	SCD+CLD+EHD
6	CSD+EHD	18	SCD+HTD+EHD
7	SCD+CLD	19	CSD+SCD+CLD+HTD
8	SCD+HTD	20	CSD+SCD+CLD+EHD
9	SCD+EHD	21	CSD+SCD+HTD+EHD
10	CSD+SCD+CLD	22	CSD+CLD+HTD+EHD
11	CSD+SCD+HTD	23	SCD+CLD+HTD+EHD
12	CSD+SCD+EHD	24	CSD+SCD+CLD+HTD+EHD

4.3.3 Pemilihan Fitur

Tahap pemilihan fitur diperlukan karena tidak semua fitur hasil ekstraksi memiliki kontribusi yang sama terhadap klasifikasi. Sebagian fitur dapat mengandung informasi penting, sedangkan sebagian lainnya dapat bersifat kurang relevan. Jika seluruh fitur digunakan tanpa seleksi, model dapat menjadi lebih kompleks dan berisiko menangkap pola yang tidak penting. Metode pemilihan fitur yang digunakan adalah *low variance feature selection*. Prinsip dasar metode ini sederhana, yaitu fitur yang memiliki varians rendah dianggap kurang informatif karena nilainya hampir sama pada banyak sampel di kelas yang berbeda-beda. Jika suatu fitur memiliki nilai yang relatif konstan, fitur tersebut tidak banyak membantu model dalam membedakan kelas. Sebaliknya, fitur dengan varians lebih tinggi dianggap lebih berpotensi memberikan informasi pembeda antar kelas.

Low variance feature selection menggunakan nilai ambang T untuk menyeleksi fitur yang digunakan. Jika varians suatu fitur lebih kecil daripada T, fitur tersebut dihapus. Jika variansnya lebih besar atau sama dengan T, fitur tersebut dipertahankan. Dengan cara ini, jumlah fitur dapat dikurangi tanpa mengubah bentuk dasar representasi data. Pemilihan nilai T menjadi aspek penting. Jika nilai T terlalu kecil, terlalu banyak fitur akan tetap dipertahankan, termasuk fitur yang kurang informatif. Jika nilai T terlalu besar, fitur yang sebenarnya bermanfaat dapat ikut terhapus. Oleh karena itu, nilai T tidak ditentukan secara manual. Nilai tersebut dioptimasi juga menggunakan GWO bersama dengan *hyperparameter* SVM. Pada studi kasus ini, nilai T dicari pada rentang $[0, 10]$.

4.3.5 Support Vector Machine

SVM pada awalnya dikembangkan sebagai model *supervised learning* untuk menyelesaikan masalah klasifikasi biner. Misalkan data latih pada klasifikasi biner terdiri atas vektor fitur input $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ dan label target $y_1, y_2, \ldots, y_N$,. Setiap vektor fitur dinyatakan sebagai $\mathbf{x}_i \in \mathbb{R}^m$, sedangkan label kelas dinyatakan

sebagai $y_i \in \{-1,1\}$, untuk $i = 1,2,..,N$. Dalam hal ini, N menyatakan jumlah data latih, sedangkan m menyatakan jumlah fitur pada setiap sampel. Tujuan utama SVM adalah mencari sebuah *hyperplane* yang dapat memisahkan data dari dua kelas yang berbeda. *Hyperplane* tersebut dapat dinyatakan seperti pada persamaan (4.7),

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b \tag{4.7}$$

dengan $\mathbf{w} \in \mathbb{R}^m$ dan $b \in \mathbb{R}$ merupakan parameter SVM, sedangkan ϕ menyatakan fungsi pemetaan dari ruang fitur asal ke ruang fitur baru. Fungsi $y(\mathbf{x})$ digunakan untuk menentukan kelas dari suatu vektor fitur input $\mathbf{x}$ yang belum diketahui labelnya. Proses klasifikasi dilakukan dengan melihat tanda dari $y(\mathbf{x})$. Jika nilai $y(\mathbf{x})$ bernilai positif, maka data diklasifikasikan ke kelas +1. Jika nilai $y(\mathbf{x})$ bernilai negatif, maka data diklasifikasikan ke kelas -1.

Nilai $\mathbf{w}$ dan b ditentukan dengan cara memaksimalkan margin. Margin adalah jarak antara *hyperplane* $\mathbf{w}^T \phi(\mathbf{x}) + b = 0$ dan vektor fitur terdekat pada data latih. Vektor fitur yang berada paling dekat dengan *hyperplane* disebut *support vector*. Dengan memaksimalkan margin, SVM berupaya membentuk batas keputusan yang tidak hanya memisahkan data latih, tetapi juga memiliki kemampuan generalisasi yang lebih baik pada data baru. Permasalahan untuk mencari nilai optimum $\mathbf{w}$ dan b dapat dirumuskan sebagai masalah optimasi seperti pada persamaan (4.8) dan persamaan (4.9).

$$\min_{\mathbf{w},b,\xi_1,\xi_2,\ldots\xi_N} \frac{1}{2}\|\mathbf{w}\| + C\sum_{i=1}^{N} \xi_i \tag{4.8}$$

dengan kendala:

$$y_i(\mathbf{w}^T \phi(\mathbf{x}_i) + b\,) \geq 1 - \xi_i, \;\; \xi_i \geq 0 \tag{4.9}$$

Pada persamaan tersebut, $\xi_i, i = 1,2,\ldots,N$ menyatakan *slack variable*. Variabel ini digunakan untuk memberikan toleransi terhadap data yang berada di dalam margin atau data yang salah

diklasifikasikan. Parameter $C > 0$ merupakan parameter regularisasi yang mengatur keseimbangan antara besarnya margin dan jumlah kesalahan klasifikasi. Nilai C yang besar membuat model lebih ketat dalam menghindari kesalahan pada data latih. Sebaliknya, nilai C yang kecil membuat model lebih toleran terhadap kesalahan, tetapi dapat menghasilkan batas keputusan yang kurang tajam.

Masalah optimasi SVM tersebut dapat dinyatakan kembali dalam bentuk dual problem seperti pada Persamaan (4.10) dan Persamaan (4.11).

$$\max_{\alpha_1,\alpha_2,\dots\alpha_N} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{4.10}$$

dengan kendala:

$$0 \le \alpha_i \le C, \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{4.11}$$

Pada persamaan tersebut, $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ menyatakan fungsi kernel, sedangkan $\alpha_1, \alpha_2, \dots \alpha_N$ menyatakan koefisien dual. Fungsi kernel memungkinkan SVM menghitung kemiripan antara dua sampel tanpa harus menghitung pemetaan ϕ secara eksplisit. Dengan cara ini, SVM dapat menangani data yang tidak dapat dipisahkan secara linier pada ruang fitur asal.

Fungsi kernel yang digunakan dalam studi ini adalah *radial basis function* (RBF). Kernel RBF dapat dinyatakan seperti pada persamaan (4.12),

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|) \tag{4.12}$$

dengan γ menyatakan koefisien kernel. Nilai γ menentukan seberapa besar pengaruh suatu sampel latih terhadap batas keputusan. Nilai γ yang besar membuat pengaruh suatu sampel menjadi lebih lokal, sehingga batas keputusan dapat menjadi lebih kompleks. Sebaliknya, nilai γ yang kecil membuat pengaruh sampel menjadi lebih luas,

sehingga batas keputusan menjadi lebih halus. Akurasi SVM sangat dipengaruhi oleh nilai hyperparameter C dan γ yang ditentukan oleh pengguna. Oleh karena itu, kedua nilai tersebut perlu dioptimasi agar SVM dapat menghasilkan performa klasifikasi terbaik.

SVM juga dapat diperluas untuk menyelesaikan masalah klasifikasi multikelas. Perluasan ini dilakukan dengan menggabungkan beberapa model SVM biner. Dua pendekatan yang umum digunakan adalah *one versus* one dan *one versus rest*. Pada studi ini, pendekatan yang digunakan adalah *one versus one*. Pendekatan ini membangun model SVM biner untuk setiap pasangan kelas. Jika jumlah kelas adalah k, maka jumlah model SVM biner yang dibentuk adalah $k(k+1)/2$. Pada saat proses klasifikasi, satu data uji akan dimasukkan ke seluruh model SVM biner yang telah dibentuk. Setiap model SVM biner hanya membandingkan dua kelas dan memberikan satu suara kepada kelas yang dianggap paling sesuai. Misalnya, jika sebuah model SVM biner dilatih untuk membedakan kelas A dan kelas B, maka model tersebut hanya akan memilih apakah data uji masuk ke kelas A atau ke kelas B. Hasil keputusan dari seluruh model kemudian dikumpulkan. Kelas akhir ditentukan berdasarkan mekanisme voting. Kelas yang memperoleh jumlah suara terbanyak dari seluruh model akan dipilih sebagai kelas hasil prediksi.

4.3.6 Optimasi *Hyperparameter* SVM dan Pemilihan Fitur dengan GWO

Pada studi kasus ini, GWO digunakan untuk mencari kombinasi terbaik dari tiga *hyperparameter*, yaitu nilai ambang pemilihan fitur T, parameter regularisasi C, dan koefisien kernel γ. Ketiga *hyperparameter* tersebut digabungkan dalam satu vektor solusi seperti pada persamaan (4.13).

$$\vec{X} = (T, C, \gamma) \tag{4.13}$$

Dengan demikian, satu serigala merepresentasikan satu konfigurasi lengkap dari proses klasifikasi.

Proses optimasi dimulai dengan membangkitkan populasi serigala secara acak. Setiap serigala memiliki nilai T, C, dan γ dalam rentang yang telah ditentukan. Pada studi kasus ini, nilai T dicari pada rentang $[0, 10]$, nilai C dicari pada rentang $[1, 1000]$, dan nilai γ dicari pada rentang $[0{,}01, 1]$. Setelah populasi awal terbentuk, setiap serigala dievaluasi berdasarkan akurasi klasifikasi yang dihasilkan. Evaluasi satu serigala dilakukan melalui beberapa tahap. Pertama, fitur hasil ekstraksi diseleksi menggunakan nilai T. Fitur yang memiliki varians di bawah T dihapus. Kedua, fitur yang tersisa digunakan sebagai input untuk SVM. Ketiga, SVM dilatih menggunakan nilai C dan γ yang dibawa oleh serigala tersebut. Keempat, model yang telah dilatih digunakan untuk mengklasifikasikan data uji. Akurasi klasifikasi kemudian dihitung untuk mendapatkan nilai fungsi objektif seperti pada persamaan (4.3).

Tujuan optimasi adalah memaksimalkan akurasi klasifikasi. Serigala yang menghasilkan akurasi tertinggi dianggap sebagai solusi terbaik. Solusi terbaik tersebut menjadi acuan bagi proses pembaruan posisi serigala pada iterasi berikutnya. Proses ini dilakukan sampai jumlah iterasi maksimum tercapai. Pada studi kasus ini, jumlah populasi yang digunakan adalah lima serigala dan jumlah iterasi maksimum adalah 10.

4.3.7 Evaluasi

Evaluasi dilakukan untuk mengukur kemampuan model dalam mengklasifikasikan citra yang tidak digunakan dalam proses pelatihan. Evaluasi semacam ini penting karena tujuan utama pembelajaran mesin bukan hanya menghasilkan model yang baik pada data latih, tetapi juga menghasilkan model yang mampu bekerja baik pada data baru.

Setiap dataset dibagi menjadi dua bagian, yaitu data latih dan data uji. Proporsi pembagian data latih dan data uji adalah 1:1, artinya, 50% data digunakan untuk pelatihan dan 50% data digunakan untuk pengujian. Pembagian data dilakukan menggunakan *stratified random sampling without replacement*. Teknik ini digunakan agar

setiap kelas tetap memiliki proporsi yang seimbang pada data latih dan data uji. Untuk memperoleh hasil yang lebih stabil, proses pembagian data diulang sebanyak lima kali. Setiap pembagian menghasilkan pasangan data latih dan data uji yang berbeda. Model kemudian dilatih dan diuji pada setiap pasangan data tersebut. Dari setiap pengujian diperoleh nilai akurasi. Nilai akhir performa model dihitung dari rata-rata akurasi seluruh pengujian.

4.3.8 Hasil Eksperimen

Hasil eksperimen menunjukkan bahwa SVM yang dioptimasi menggunakan GWO mampu menghasilkan akurasi yang tinggi pada ketiga dataset. Hasil ini menunjukkan bahwa kombinasi optimasi *hyperparameter*, pemilihan fitur, dan fusi fitur MPEG-7 dapat meningkatkan kemampuan model dalam mengklasifikasikan citra buah-buahan.

Pada dataset Ubaya-IFDS3000, akurasi terbaik mencapai 99,21%. Nilai ini diperoleh pada penggunaan fusi fitur CSD+SCD dan CSD+SCD+CLD. Fusi fitur CSD+SCD+HTD juga menghasilkan performa yang sangat dekat, yaitu 99,16%. Hasil ini menunjukkan bahwa kombinasi fitur warna memiliki kontribusi kuat dalam membedakan 15 kelas buah pada dataset tersebut. Informasi warna yang direpresentasikan melalui CSD dan SCD terbukti sangat relevan untuk mengenali buah-buahan pada dataset Ubaya-IFDS3000.

Pada dataset Ubaya-IFDS5000, akurasi terbaik mencapai 98,29%. Nilai ini diperoleh pada fusi fitur CSD+SCD+CLD+HTD. Akurasi ini sedikit lebih rendah daripada hasil pada Ubaya-IFDS3000. Perbedaan tersebut dapat dipahami karena Ubaya-IFDS5000 memiliki jumlah kelas yang lebih banyak dan variasi visual yang lebih kompleks. Pada dataset ini, fitur warna tetap berperan penting, tetapi penambahan fitur tekstur membantu memperkaya representasi citra. Hal ini terlihat dari performa terbaik yang diperoleh ketika fitur warna dan tekstur digabungkan.

Pada dataset *Supermarket Produce*, akurasi terbaik mencapai 99,85%. Nilai ini diperoleh pada fusi fitur CSD+SCD+CLD. Fusi fitur CSD+SCD juga menghasilkan akurasi tinggi, yaitu 99,82%. Hasil ini menunjukkan bahwa metode yang digunakan tidak hanya bekerja baik pada dataset Ubaya, tetapi juga mampu bekerja sangat baik pada dataset lain dengan kondisi citra yang berbeda. Hasil eksperimen lengkap evaluasi model SVM yang dioptimasi menggunakan GWO dapat dilihat pada Tabel 4.3.

Tabel 4.3. Akurasi klasifikasi model SVM yang dioptimasi menggunakan GWO.

Fusi Fitur (F_i)	**Average akurasi (%)**		
	Ubaya-IFDS3000	**Ubaya-IFDS5000**	**Supermarket produce**
F_1	98.20	95.45	99.67
F_2	98.61	96.30	99.48
F_3	**99.21**	97.43	**99.82**
F_4	98.19	95.64	99.65
F_5	97.89	96.30	99.61
F_6	97.67	95.40	99.56
F_7	98.85	96.50	99.67
F_8	98.60	97.47	99.68
F_9	98.37	95.04	99.57
F_{10}	**99.21**	**97.71**	**99.85**
F_{11}	**99.16**	**98.18**	**99.77**
F_{12}	98.85	96.26	99.70
F_{13}	98.09	96.66	99.59
F_{14}	98.08	95.45	99.53

F_{15}	96.88	96.09	99.45
F_{16}	99.03	97.69	99.64
F_{17}	98.55	94.83	99.64
F_{18}	98.45	96.87	99.61
F_{19}	99.15	**98.29**	99.76
F_{20}	98.77	96.90	99.70
F_{21}	98.89	97.51	99.71
F_{22}	97.21	96.05	99.53
F_{23}	98.59	97.11	99.61
F_{24}	98.80	97.60	99.68

05

Aplikasi *Grey Wolf Optimizer* dalam Pemilihan Fitur

Pemilihan fitur merupakan tahap prapemrosesan yang bertujuan memilih subset fitur yang paling relevan serta mengurangi fitur yang redundan atau mengandung *noise*. Pada data modern, khususnya data berdimensi tinggi, keberadaan fitur yang berlebihan tidak hanya meningkatkan biaya komputasi, tetapi juga dapat menurunkan kemampuan generalisasi model karena model harus memproses banyak variabel yang tidak informatif (Siswantoro et al., 2020b). Kondisi ini menyebabkan proses pelatihan menjadi lebih lambat, model lebih mudah mengalami *overfitting*, dan interpretasi hasil menjadi lebih sulit dilakukan. Oleh sebab itu, pemilihan fitur berperan penting untuk menyederhanakan representasi data tanpa menghilangkan informasi utama yang dibutuhkan oleh model. Dengan subset fitur yang lebih ringkas dan lebih bermakna, kinerja model dapat menjadi lebih efisien, lebih stabil, dan lebih akurat (Siswantoro et al., 2026).

Dalam banyak literatur, pemilihan fitur diposisikan sebagai masalah optimasi kombinatorial. Jika sebuah dataset memiliki *N* fitur awal, maka secara teoritis tersedia 2^N kandidat subset fitur. Pertumbuhan ruang solusi seperti ini sangat cepat, sehingga

pendekatan pencarian ekshaustif menjadi tidak praktis, terlebih ketika data yang dihadapi berukuran besar atau berdimensi tinggi. Atas dasar itu, pendekatan metaheuristik menjadi sangat relevan. Hal ini karena metode optimasi metaheuristik tidak memeriksa semua subset secara menyeluruh, tetapi menelusuri ruang solusi melalui agen-agen pencari yang bergerak, diperbarui, dan dibandingkan secara iteratif. Dalam konteks pemilihan fitur, pendekatan ini sangat berguna karena dapat mencari kompromi antara dua target yang sering saling bertentangan, yaitu menjaga akurasi model dan mengurangi jumlah fitur (Huang et al., 2025; Liu et al., 2024).

GWO merupakan salah satu metaheuristik berbasis *swarm intelligence* yang banyak digunakan dalam pemilihan fitur karena strukturnya sederhana, parameter internalnya sedikit, dan proses pencariannya mudah diadaptasi ke berbagai domain. GWO juga memiliki pembagian peran eksplorasi dan eksploitasi yang jelas melalui mekanisme mencari mangsa, mengepung, dan menyerang. Keunggulan lain dari GWO adalah kemampuannya membangun arah pencarian melalui hirarki sosial. Dalam algoritma ini, tiga agen elit, yaitu serigala α, β, dan δ, berperan sebagai penunjuk arah bagi agen lain. Secara konseptual, hal ini sangat sesuai untuk pemilihan fitur karena subset fitur yang terbaik, kedua terbaik, dan ketiga terbaik dapat dipakai sebagai acuan untuk membentuk kandidat subset berikutnya (Shahin et al., 2023).

Namun demikian, GWO dasar tidak sepenuhnya bebas dari keterbatasan. Ketika ruang solusi sangat besar, terutama pada data berdimensi tinggi, populasi awal yang sepenuhnya acak dapat memuat terlalu banyak informasi redundan. Selain itu, pada iterasi-iterasi akhir, serigala α, β, dan δ sering kali mengerucut ke solusi yang hampir sama, sehingga banyak langkah pembaruan menjadi kurang informatif. Meski begitu, justru di situlah kekuatan GWO sebagai fondasi metode pemilihan fitur. Karena mekanismenya sederhana dan modular, GWO relatif mudah diperluas melalui berbagai strategi, seperti inisialisasi berbasis filter, pembobotan elit,

hibridisasi dengan Differential Evolution, atau penambahan Levy-flight.

Banyak penelitian terkini yang memperkaya GWO lebih tahan terhadap optimum lokal dan lebih efisien pada dimensi tinggi untuk pemilihan fitur. Selain GWO standar (Asemi and Farajzadeh, 2025; Shahin et al., 2023), beberapa varian GWO yang digunakan dalam pemilihan fitur antara lain *Multi-strategy Improved Grey Wolf Optimizer* (MIGWO) (Huang et al., 2025), *Golden Jackal–Grey Wolf Hybrid Optimization Algorithm* (GJO–GWO) (Liu et al., 2024), Hibrida *Gray Wolf Optimization* dengan *Grasshopper Optimization Algorithm* (GWO–GOA) (Purushothaman et al., 2020), dan Hibrida *Grey Wolf Optimizer* dengan *Whale Optimization Algorithm* (GWO-WOA) (Siswantoro et al., 2026). Rangkuman aplikasi GWO dan variannya dalam pemilihan fitur dapat dilihat pada Tabel 5.1.

Tabel 5.1. Rangkuman aplikasi GWO dalam pemilihan fitur.

Varian GWO	**Model Pembelajaran Mesin**	**Fungsi Transfer**	**Fungsi Objectif**
GWO (Shahin et al., 2023)	KNN	Berbentuk-S	$akurasi$
GWO (Asemi and Farajzadeh, 2025)	XGBoost	Berbentuk-S	$1 - akurasi$
MIGWO (Huang et al., 2025)	KNN	Berbentuk-S	$(1-\lambda)\frac{d}{D} + \lambda(1 - akurasi)$
GJO-GWO (Liu et al., 2024)	KNN	Ambang langsung	$(1-\lambda)\frac{d}{D} + \lambda(1 - akurasi)$
GWO-GOA (Purushothaman et al., 2020)	Fuzzy c-means	Ambang langsung	*Mean Absolute Difference*

Varian GWO	Model Pembelajaran Mesin	Fungsi Transfer	Fungsi Objectif
GWO-WOA (Siswantoro et al., 2026)	KNN	Berbentuk-S, berbentuk-V, berbentuk-V terbalik	$(1-\lambda)\frac{d}{D} + \lambda(1-akurasi)$

Secara umum, tahapan seleksi fitur dengan GWO dimulai dengan merepresentasikan setiap serigala sebagai vektor biner yang menunjukkan fitur mana yang dipilih dan mana yang dibuang. Setelah itu, populasi awal serigala dibangkitkan untuk membentuk sekumpulan kandidat subset fitur yang berbeda. Setiap subset fitur kemudian dievaluasi melalui fungsi objektif, biasanya dengan melatih dan menguji model pembelajaran mesin pada data yang telah direduksi sesuai fitur terpilih, lalu menghitung nilai akurasi atau fitness gabungan antara akurasi dan jumlah fitur. Berdasarkan nilai fitness tersebut, tiga solusi terbaik ditetapkan sebagai serigala α, β, dan δ, lalu digunakan untuk memandu pembaruan posisi serigala lain sesuai mekanisme GWO. Proses evaluasi subset fitur dan pembaruan posisi ini dilakukan secara iteratif hingga jumlah iterasi maksimum tercapai, dan solusi terbaik pada akhir proses dipilih sebagai subset fitur optimal.

5.1 Representasi Serigala dalam Pemilihan Fitur

Berbeda dari optimasi hyperparameter, di mana setiap posisi agen umumnya merepresentasikan kombinasi nilai parameter model, pada pemilihan fitur setiap serigala dalam GWO direpresentasikan sebagai kandidat subset fitur. Dengan demikian, seekor serigala tidak lagi dipandang sebagai titik pada ruang fisik, melainkan sebagai satu solusi yang menunjukkan fitur mana yang dipertahankan dan fitur

mana yang dibuang. Jika suatu dataset memiliki D fitur, maka posisi seekor serigala dapat dinyatakan sebagai vektor berdimensi D, di mana setiap komponennya berkorespondensi langsung dengan satu fitur pada dataset. Dalam konteks ini, ruang pencarian GWO berubah menjadi ruang seluruh kemungkinan subset fitur, yang secara teoritis berjumlah 2^D. Inilah yang membuat pemilihan fitur menjadi masalah optimasi kombinatorial yang kompleks, terutama ketika jumlah fitur sangat besar (Shahin et al., 2023).

Dalam implementasi pemilihan fitur, representasi solusi biasanya dituliskan sebagai vektor $x = (x_1, x_2, \dots, x_D), x_j \in \{0,1\}$, dengan $x_j = 1$ menyatakan bahwa fitur ke-j dipilih, sedangkan $x_j = 0$ menyatakan bahwa fitur tersebut dibuang. Bentuk representasi ini membuat satu serigala identik dengan satu masker seleksi fitur. Dengan kata lain, posisi serigala secara langsung merepresentasikan struktur subset fitur yang sedang diuji. Jika sebuah fitur aktif, maka fitur itu akan ikut dipakai dalam proses pembentukan model; sebaliknya, jika tidak aktif, fitur tersebut dikeluarkan dari subset. Representasi ini sederhana, tetapi sangat kuat karena memungkinkan setiap solusi dalam populasi ditafsirkan secara langsung sebagai kombinasi fitur yang berbeda.

Representasi biner tersebut memiliki implikasi metodologis yang penting. Karena setiap bit menyatakan keputusan pilih atau buang, maka perubahan posisi agen tidak dapat dipahami sekadar sebagai pergeseran numerik, tetapi harus dibaca sebagai perubahan struktur subset fitur. Artinya, ketika satu atau beberapa komponen vektor berubah, yang sesungguhnya terjadi adalah perubahan komposisi subset: ada fitur yang ditambahkan, ada yang dihapus, atau ada yang bertukar status antara aktif dan tidak aktif. Dalam kerangka ini, pergerakan serigala menjadi ekuivalen dengan proses pencarian kombinasi fitur terbaik. Oleh sebab itu, representasi solusi bukan sekadar formalitas matematis, melainkan inti dari bagaimana GWO diadaptasi untuk menyelesaikan masalah pemilihan fitur.

Jika dibandingkan dengan masalah optimasi kontinu biasa, representasi seperti ini membuat makna posisi menjadi jauh lebih

kaya. Pada optimasi kontinu, perubahan posisi biasanya ditafsirkan sebagai perpindahan menuju nilai fungsi yang lebih baik di dalam ruang numerik. Sebaliknya, pada pemilihan fitur, satu langkah perubahan posisi dapat berarti perubahan langsung pada struktur informasi yang digunakan model. Sebuah solusi tidak dinilai karena berada dekat atau jauh dari titik tertentu, melainkan karena subset fitur yang diwakilinya mampu menangkap informasi yang paling relevan. Dengan demikian, posisi serigala dalam pemilihan fitur sesungguhnya adalah bentuk representasi dari keputusan struktural terhadap data, bukan hanya nilai numerik abstrak (Asemi and Farajzadeh, 2025).

Pada tahap inisialisasi, populasi serigala umumnya dibangkitkan secara acak. Dalam pendekatan ini, setiap serigala sejak awal sudah mewakili satu subset fitur kandidat yang berbeda dari serigala lain. Strategi acak memiliki kelebihan karena mudah diterapkan dan dapat menghasilkan keragaman awal dalam populasi. Namun, pada data berdimensi tinggi, inisialisasi yang terlalu acak juga dapat menyebabkan banyak solusi awal memuat fitur-fitur yang tidak relevan atau redundant. Akibatnya, populasi awal sering kali tersebar pada wilayah solusi yang kurang menjanjikan, sehingga proses pencarian memerlukan waktu lebih lama untuk bergerak menuju subset yang benar-benar informatif.

Karena keterbatasan tersebut, beberapa varian GWO untuk pemilihan fitur memperkenalkan strategi inisialisasi berbasis informasi. Salah satu pendekatan yang cukup menonjol adalah menggunakan skor kepentingan fitur untuk mengarahkan pembentukan populasi awal. Ide dasarnya ialah bahwa serigala tidak perlu memulai pencarian dari subset yang sepenuhnya acak, melainkan dapat dibentuk dari distribusi awal yang sudah lebih kaya informasi. Dengan cara ini, populasi awal diharapkan mengandung lebih banyak fitur yang relevan dan lebih sedikit fitur yang tidak penting. Strategi seperti ini sangat bermanfaat terutama ketika jumlah fitur sangat besar, karena dapat membantu memperkecil ruang pencarian efektif sejak awal optimasi (Huang et al., 2025).

Salah satu contoh nyata dari pendekatan tersebut adalah *ReliefF-based initialization*. Pada strategi ini, setiap fitur terlebih dahulu diberi skor kepentingan, lalu fitur-fitur dengan skor yang lebih tinggi digunakan untuk membangun populasi awal. Dengan demikian, serigala sejak awal tidak lagi tersebar merata di seluruh ruang solusi, tetapi lebih banyak berada pada wilayah yang berpotensi menghasilkan subset fitur yang baik. Secara konseptual, strategi ini dapat dipandang sebagai cara untuk memberi pengetahuan awal kepada populasi tanpa menghilangkan sifat stokastik dari metaheuristik. Akibatnya, serigala tetap beragam, tetapi tidak benar-benar memulai pencarian dari kondisi yang buta sama sekali.

Dari sudut pandang pemodelan, representasi serigala dalam pemilihan fitur juga menunjukkan bahwa satu agen bukan hanya solusi tunggal, tetapi juga wadah informasi tentang hubungan antarfitur. Ketika dua serigala memiliki pola aktivasi fitur yang berbeda, perbedaan itu berarti keduanya sedang mengeksplorasi hipotesis yang berbeda mengenai fitur mana yang paling relevan bagi model. Karena itu, populasi serigala pada feature selection sesungguhnya dapat dipandang sebagai sekumpulan alternatif pandangan terhadap data. Semakin beragam pola subset yang dibentuk oleh populasi, semakin luas pula wilayah pencarian yang dapat dijelajahi. Inilah salah satu alasan mengapa keberagaman populasi menjadi isu penting dalam penerapan GWO untuk pemilihan fitur.

Representasi serigala sebagai subset fitur juga memudahkan interpretasi hasil optimasi. Setelah iterasi selesai, solusi terbaik tidak hanya berupa angka fitness atau posisi abstrak, tetapi dapat langsung diterjemahkan menjadi daftar fitur yang dipilih. Hal ini membuat GWO bukan hanya berguna untuk meningkatkan performa model, tetapi juga membantu proses analisis data, dan hasil akhirnya menunjukkan fitur-fitur mana yang dianggap paling informatif oleh mekanisme pencarian. Pada bidang-bidang seperti bioinformatika, pengenalan emosi, atau data medis berdimensi tinggi, kemampuan interpretatif seperti ini sangat berharga karena peneliti tidak hanya

membutuhkan model yang akurat, tetapi juga ingin mengetahui fitur mana yang dominan dalam proses pengambilan keputusan.

5.2 Fungsi Transfer dan Mekanisme Binarisasi

Dalam pemilihan fitur, GWO dan variannya pada dasarnya memperbarui posisi agen pencari di ruang kontinu, sedangkan solusi akhir yang dibutuhkan berbentuk biner, yaitu setiap fitur harus diputuskan dipilih atau tidak dipilih. Ketidakcocokan antara dua ruang ini membuat proses binarisasi menjadi komponen yang sangat penting. Karena itu, banyak penelitian memperkenalkan fungsi transfer sebagai jembatan antara nilai kontinu hasil pembaruan posisi dan keputusan diskrit 0 – 1. Dalam kerangka ini, fungsi transfer tidak langsung memilih fitur, tetapi terlebih dahulu mengubah posisi kontinu menjadi nilai probabilistik yang kemudian dipakai untuk menentukan status biner setiap fitur.

Secara umum, jika x adalah nilai posisi kontinu suatu agen pada satu dimensi fitur tertentu, maka fungsi transfer $TF(x)$ mengubah nilai tersebut ke sebuah nilai pada interval $(0,1)$. Nilai ini kemudian dibandingkan dengan bilangan acak $p \in (0,1)$ untuk menentukan status fitur dalam ruang biner x^b. Salah satu bentuk aturan yang banyak dipakai untuk proses binarisasi tersebut adalah seperti pada persamaan (5.1). Aturan tersebut menunjukkan bahwa semakin besar nilai fungsi transfer, semakin besar pula peluang fitur dipilih. Dengan kata lain, fungsi transfer berperan sebagai pengatur intensitas perubahan bit dari ruang kontinu ke ruang diskrit (Siswantoro et al., 2026).

$$x^b = \begin{cases} 1, & p < TF(x) \\ 0, & p \geq TF(x) \end{cases} \tag{5.1}$$

Ada beberapa jenis fungsi transfer yang umum digunakan dalam pemilihan fitur menggunakan GWO dan variannya. Pertama adalah keluarga fungsi transfer berbentuk-S (*S-shaped transfer functions*). Intuisi utamanya adalah bahwa nilai kontinu yang semakin besar akan menghasilkan probabilitas aktivasi fitur yang semakin

tinggi secara halus dan monoton. Bentuk ini banyak dipakai karena sederhana, stabil, dan mudah diinterpretasikan sebagai peluang biner. Selain itu, fungsi berbentuk-S cenderung memberikan transisi yang mulus di sekitar titik tengah, sehingga cocok ketika perubahan subset fitur diinginkan berlangsung secara bertahap, bukan terlalu agresif (Asemi and Farajzadeh, 2025; Huang et al., 2025; Shahin et al., 2023; Siswantoro et al., 2026).

Beberapa varian fungsi berbentuk-S yang umum digunakan didefinisikan pada persamaan (5.2) – (5.5). Keempat bentuk tersebut memiliki pola dasar yang sama, yaitu sigmoid, tetapi berbeda pada tingkat kecuraman kurva. Semakin besar faktor pengali di dalam eksponen, semakin tajam perubahan probabilitas di sekitar titik nol. Sebaliknya, pembagi yang lebih besar menghasilkan perubahan yang lebih landai, sehingga keputusan biner menjadi lebih konservatif.

$$S_1(x) = \frac{1}{1 + e^{-2x}} \tag{5.2}$$

$$S_2(x) = \frac{1}{1 + e^{-x}} \tag{5.3}$$

$$S_3(x) = \frac{1}{1 + \mathrm{e}^{-x/2}} \tag{5.4}$$

$$S_4(x) = \frac{1}{1 + \mathrm{e}^{-x/3}} \tag{5.5}$$

Selain keluarga fungsi transfer berbentuk-S, banyak penelitian juga menggunakan fungsi transfer berbentuk-V (*V-shaped transfer functions*). Berbeda dari keluarga sigmoid yang menafsirkan nilai keluaran sebagai peluang aktivasi fitur secara langsung, keluarga fungsi transfer berbentuk-V lebih sering dipahami sebagai peluang perubahan status bit. Artinya, semakin besar nilai mutlak posisi kontinu, semakin besar pula peluang suatu fitur membalik dari 0 ke 1 atau dari 1 ke 0. Pendekatan ini sering dipilih ketika algoritma diharapkan memiliki perilaku pencarian yang lebih dinamis dan lebih mampu keluar dari stagnasi lokal.

Empat bentuk fungsi transfer berbentuk-V yang banyak digunakan didefinisikan seperti pada persamaan (5.6) – (5.9).

Keempat fungsi tersebut menghasilkan keluaran pada interval $(0, 1)$, tetapi memiliki bentuk lengkung yang berbeda. Fungsi $\tanh(x)$ dan $\arctan(x)$, misalnya, memberi respons yang relatif cepat ketika $|x|$ membesar, sedangkan bentuk berbasis *error function* dan rasio aljabar memberi transisi yang sedikit berbeda di sekitar nol. Secara praktis, pilihan bentuk V memengaruhi seberapa agresif agen akan mengubah status fitur selama proses pencarian.

$$V_1(x) = \left|\operatorname{erf}\left(\frac{\sqrt{\pi}}{2x}\right)\right| \tag{5.6}$$

$$V_2(x) = |\tanh(x)| \tag{5.7}$$

$$V_3(x) = \left|\frac{x}{\sqrt{1+x^2}}\right| \tag{5.8}$$

$$V_4(x) = \left|\frac{2}{\pi}\arctan\left(\frac{\pi}{2}x\right)\right| \tag{5.9}$$

Perkembangan berikutnya melahirkan fungsi transfer berbentuk-V terbalik, yaitu bentuk yang membalik pola dari fungsi V biasa. Jika fungsi V standar menaikkan peluang perubahan bit ketika $|x|$ makin besar, maka fungsi transfer berbentuk-V terbalik justru memberi peluang tertinggi ketika nilai posisi kontinu dekat ke nol. Secara konseptual, pendekatan ini berangkat dari gagasan bahwa nilai yang dekat nol menunjukkan ketidakpastian atau preferensi yang lemah terhadap keputusan memilih atau membuang fitur. Karena itu, justru pada daerah inilah perubahan bit perlu lebih sering terjadi agar algoritma dapat meninjau ulang fitur-fitur yang masih ambigu. Sebaliknya, ketika $|x|$ sudah besar, algoritma dianggap memiliki keyakinan yang lebih kuat terhadap status suatu fitur, sehingga peluang pembalikan bit diturunkan.

Bentuk umum dari fungsi transfer transfer berbentuk-V terbalik didefinisikan pada persamaan (5.10) – (5.13). Keluarga fungsi transfer berbentuk-V terbalik ini dirancang untuk menjaga keragaman pencarian pada fitur-fitur yang belum jelas statusnya. Di antara beberapa bentuk tersebut, hasil eksperimen menunjukkan bahwa fungsi transfer berbentuk-V terbalik sering memberikan keseimbangan yang baik antara akurasi klasifikasi dan reduksi jumlah

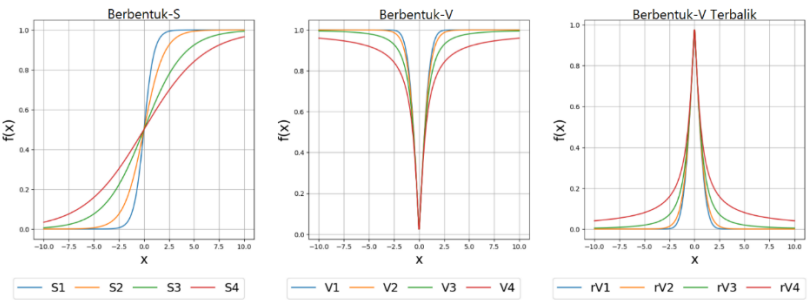
fitur, karena mampu menjaga eksplorasi pada bit-bit yang masih belum stabil sambil menahan perubahan berlebihan pada keputusan yang sudah kuat (Siswantoro et al., 2026). Kurva fungsi transfer berbentuk-S, berbetuk-V, dan berbentuk-V terbalik dapat dilihat pada Gambar 5.1.

$$rV_1(x) = 1 - \left|\operatorname{erf}\left(\frac{\sqrt{\pi}}{2x}\right)\right| \tag{5.10}$$

$$rV_2(x) = 1 - |\tanh(x)| \tag{5.11}$$

$$rV_3(x) = 1 - \left|\frac{x}{\sqrt{1+x^2}}\right| \tag{5.12}$$

$$rV_4(x) = 1 - \left|\frac{2}{\pi}\arctan\left(\frac{\pi}{2}x\right)\right| \tag{5.13}$$



Gambar 5.1. Kurva fungsi transfer berbentuk-S, berbetuk-V, dan berbentuk-V terbalik

Di samping pendekatan berbasis fungsi transfer, terdapat pula metode binarisasi yang tidak menggunakan fungsi transfer sama sekali, melainkan memakai nilai ambang langsung. Salah satu bentuk yang digunakan adalah seperti pada persamaan (5.14) (Liu et al., 2024; Purushothaman et al., 2020).

$$x^b = \begin{cases} 0, & x \le 0.5 \\ 1, & x > 0.5 \end{cases} \tag{5.14}$$

Aturan ini jauh lebih sederhana karena nilai posisi kontinu langsung dipotong oleh ambang tetap, biasanya 0.5. Pendekatan seperti ini mudah diimplementasikan dan tidak menambah komputasi berarti, tetapi kurang lentur dibanding fungsi transfer karena tidak

memberikan interpretasi probabilistik atas posisi kontinu. Meski demikian, ambang langsung tetap menarik ketika peneliti menginginkan prosedur binarisasi yang cepat, sederhana, dan mudah dijelaskan.

Secara metodologis, perbedaan utama antara fungsi transfer dan ambang langsung terletak pada tingkat fleksibilitasnya. Fungsi transfer memberi ruang bagi algoritma untuk menerjemahkan posisi kontinu ke keputusan biner secara bertahap dan stokastik, sehingga pencarian dapat lebih halus dan lebih adaptif. Sebaliknya, ambang langsung membuat keputusan secara deterministik begitu posisi melewati batas tertentu. Oleh sebab itu, fungsi berbentuk-S umumnya cocok ketika stabilitas aktivasi fitur lebih diutamakan, fungsi berbentuk-V cocok ketika perubahan bit yang lebih aktif diinginkan, dan fungsi fungsi transfer berbentuk-V terbalik cocok ketika ingin memberi perhatian lebih pada fitur-fitur yang masih ambigu. Sementara itu, ambang langsung dapat dipilih ketika kesederhanaan implementasi menjadi pertimbangan utama.

5.3 Fungsi Objektif dalam Pemilihan Fitur

Dalam pemilihan fitur berbasis GWO dan variannya, fungsi objektif (*fitness*) berperan sebagai alat ukur untuk menilai seberapa baik suatu subset fitur. Karena setiap agen merepresentasikan kandidat subset fitur, maka algoritma membutuhkan kriteria kuantitatif untuk memutuskan apakah subset tersebut layak dipertahankan, diperbaiki, atau digantikan oleh subset lain. Secara umum, fungsi objektif pada pemilihan fitur tidak hanya menilai satu aspek, tetapi harus mempertimbangkan dua sasaran yang saling terkait, yaitu meningkatkan kualitas klasifikasi dan mengurangi jumlah fitur terpilih. Hal ini membuat pemilihan fitur diperlakukan sebagai masalah optimasi multiobjektif, meskipun dalam praktiknya sering disederhanakan ke bentuk fungsi skalar tunggal agar mudah dioptimasi oleh metaheuristik (Huang et al., 2025).

Bentuk fungsi objektif yang paling sederhana adalah menggunakan akurasi klasifikasi secara langsung sebagai ukuran kualitas subset fitur. Dalam pendekatan ini, subset fitur yang diwakili oleh satu agen terlebih dahulu dipakai untuk membentuk data tereduksi, lalu sebuah model dilatih dan diuji pada data tersebut. Nilai *fitness* kemudian diambil dari performa klasifikasi yang dimaksimumkan, sehingga semakin tinggi akurasi yang diperoleh, semakin baik subset fitur tersebut (Shahin et al., 2023). Secara konseptual, bentuk sederhananya dinyatakan seperti pada persamaan (5.15). Untuk masalah minimisasi yang umum digunakan dalam optimasi, persamaan (5.15) dikonversi menjadi bentuk seperti pada persamaan (5.16) (Asemi and Farajzadeh, 2025).

$$fitness = akurasi \tag{5.15}$$

$$fitness = 1 - akurasi \tag{5.16}$$

Pendekatan berbasis akurasi murni biasanya digunakan ketika fokus utama penelitian adalah meningkatkan performa prediksi model. Dalam skema *wrapper*, kualitas subset fitur memang sangat bergantung pada bagaimana model merespons fitur-fitur yang dipilih. Karena itu, akurasi menjadi ukuran yang intuitif dan langsung. Namun, jika hanya akurasi yang dijadikan sasaran, algoritma dapat terdorong mempertahankan terlalu banyak fitur selama hal itu masih meningkatkan performa klasifikasi. Akibatnya, tujuan reduksi dimensi menjadi kurang tercapai, dan model bisa tetap mahal secara komputasi walaupun akurasinya tinggi.

Karena keterbatasan tersebut, bentuk fungsi objektif yang paling umum dalam pemilihan fitur adalah kombinasi linear antara kesalahan klasifikasi dan proporsi fitur terpilih. Fungsi objektif ini akan menyeimbangkan dua tujuan, yaitu meminimalkan error model dan meminimalkan ukuran subset (Huang et al., 2025; Liu et al., 2024; Siswantoro et al., 2026). Bentuk umum yang dapat mewakili keluarga fungsi tersebut dinyatakan pada persamaan (5.17),

$$fitness = \lambda(1 - akurasi) + (1 - \lambda)\frac{d}{D} \tag{5.17}$$

dengan d dan D masing-masing adalah banyaknya fitur terpilih dan banyaknya semua fitur serta λ adalah bilangan riil positif kurang dari 1 sebagai parameter penyeimbang. Semakin kecil nilai fungsi ini, semakin baik subset fitur yang dihasilkan, karena berarti ia memberikan kesalahan yang rendah dengan jumlah fitur yang relatif sedikit.

Penentuan nilai fitness setiap serigala pada masalah klasifikasi dimulai dengan menerjemahkan posisi biner serigala menjadi subset fitur kandidat. Subset tersebut digunakan untuk membentuk data tereduksi dari dataset X beserta label kelas y. Selanjutnya, data dibagi menjadi data latih dan data uji, lalu model pembelajaran mesin dilatih menggunakan data latih yang hanya memuat fitur-fitur terpilih. Setelah proses pelatihan selesai, performa model dihitung pada data uji dengan subset fitur yang sama untuk memperoleh nilai akurasi. Pada tahap akhir, nilai akurasi tersebut beserta banyaknya fitur yang terpilih disubstitusikan ke salah satu fungsi objektif pada persamaan (5.15) – (5.17).

Selain untuk masalah klasifikasi, fungsi objektif pada pemilihan fitur juga dapat dirancang untuk konteks *clustering*, seperti dada seleksi fitur untuk data teks (Purushothaman et al., 2020). Bentuk fungsi objektif yang digunakan pada masalah *clustering* tersebut adalah *Mean Absolute Difference* (MAD), yang mengevaluasi sebaran bobot fitur terpilih terhadap nilai rata-ratanya, seperti pada persamaan (5.18),

$$MAD = \frac{1}{Sf_i}\sum_{k=1}^{j}|m_{ik} - \overline{m_l}|, \ \overline{m_l} = \frac{1}{Sf_i}\sum_{k=1}^{j} m_{ik} \qquad (5.18)$$

dengan Sf_i adalah jumlah fitur terpilih pada dokumen i, m_{ik} adalah bobot fitur k pada dokumen i, dan j adalah pada dokumen asal. Fungsi tersebut menilai kualitas subset fitur dari seberapa konsisten dan seberapa informatif bobot fitur-fitur teks yang dipilih untuk proses *clustering*. Detail algoritma pemilihan fitung menggunakan GWO pada masalah klasifikasi dapat dilihat pada Algoritma 5.1.

Algoritma 5.1. Pemilhan fitur menggunakan GWO pada masalah klasifikasi

Input:

Dataset X dengan D fitur, label kelas y, ukuran populasi serigala N, jumlah iterasi maksimum T, model pembelajaran mesin untuk evaluasi subset fitur, parameter penyeimbang λ pada fungsi objektif, batas bawah $\overrightarrow{lb}$, dan batas atas $\overrightarrow{ub}$.

Output:

Subset fitur terbaik S_α, vektor biner terbaik $\vec{X}_\alpha$ dan nilai *fitness* terbaik.

Prosedur:

1. Bangkitkan populasi awal sebanyak N serigala. Setiap serigala direpresentasikan sebagai vektor berdimensi D, yaitu kandidat subset fitur.
2. Ubah setiap posisi serigala ke bentuk biner menggunakan persamaan (5.1).
3. Hitung nilai fitness setiap serigala menggunakan langkah-langkah berikut:
   3.1. Pilih subset fitur berdasarkan vektor biner posisi serigala,
   3.2. Bagi dataset X dan label kelas y menjadi data latih dan data uji.
   3.3. Latih model menggunakan data latih dengan subset fitur terpilih.
   3.4. Hitung akurasi model menggunakan data uji dengan subset fitur terpilih.
   3.5. Hitung *fitness* menggunakan salah satu fungsi pada persamaan (5.15) – (5.17)
4. Tentukan tiga serigala terbaik dalam populasi sebagai $\vec{X}_\alpha$, $\vec{X}_\beta$, dan $\vec{X}_\delta$.

5. Tetapkan iterasi awal $t \leftarrow 1$.
6. Hitung nilai a, $\vec{A}$, dan $\vec{C}$ menggunakan persamaan (2.5), (2.3), dan (2.4).
7. Selama $t \leq T$, lakukan langkah-langkah berikut:
   7.1. Untuk setiap serigala ke-$i = 1,2, \ldots, N$, lakukan:
      a. Perbarui posisi serigala ke-i menggunakan persamaan (2.12).
      b. Jika posisi baru berada di luar batas pencarian, lakukan penyesuaian agar tetap berada pada rentang $[\overrightarrow{lb}, \overrightarrow{ub}]$.
      c. Ubah setiap posisi serigala ke bentuk biner menggunakan persamaan (5.1).
   7.2. Hitung kembali nilai *fitness* setiap serigala menggunakan langkah 3.
   7.3. Perbarui nilai a, $\vec{A}$, dan $\vec{C}$ menggunakan persamaan (2.5), (2.3), dan (2.4).
   7.4. Perbarui $\vec{X}_\alpha$, $\vec{X}_\beta$, dan $\vec{X}_\delta$ berdasarkan tiga nilai *fitness* terbaik pada populasi.
   7.5. Set $t \leftarrow t + 1$.
8. Kembalikan Subset fitur terbaik S_α, vektor biner terbaik $\vec{X}_\alpha$ dan nilai *fitness* terbaik.

5.4 Studi Kasus Pemilihan Fitur Menggunakan Hibrida GWO-WOA Biner

Pemilihan fitur merupakan salah satu tahap penting dalam pembelajaran mesin, terutama ketika dataset memiliki jumlah fitur yang besar. Pada data berdimensi tinggi, tidak semua fitur memiliki kontribusi yang sama terhadap proses klasifikasi. Sebagian fitur dapat bersifat relevan dan membantu model mengenali pola kelas. Namun,

sebagian fitur lain dapat bersifat redundan, tidak relevan, atau mengandung noise. Keberadaan fitur yang kurang informatif dapat meningkatkan kompleksitas model, memperbesar biaya komputasi, memperlambat proses pelatihan, dan menurunkan kemampuan generalisasi model.

Permasalahan tersebut semakin penting pada era big data. Banyak bidang aplikasi, seperti kesehatan, keuangan, keamanan siber, dan bioinformatika, menghasilkan data dengan jumlah fitur yang sangat besar. Jika seluruh fitur digunakan tanpa seleksi, model pembelajaran mesin dapat bekerja lebih lambat dan lebih mudah mengalami overfitting. Oleh karena itu, reduksi dimensi menjadi tahap prapemrosesan yang penting untuk menghasilkan model yang lebih efisien dan lebih akurat.

Secara umum, reduksi dimensi dapat dilakukan melalui dua pendekatan, yaitu ekstraksi fitur dan pemilihan fitur. Ekstraksi fitur membentuk fitur baru dari kombinasi fitur asli. Sebaliknya, pemilihan fitur mempertahankan sebagian fitur asli yang dianggap paling relevan. Pemilihan fitur memiliki keunggulan dari sisi interpretasi karena fitur yang dipilih tetap berasal dari fitur awal. Dengan demikian, hasil optimasi tidak hanya menghasilkan model yang lebih efisien, tetapi juga membantu pengguna memahami fitur mana yang lebih berperan dalam proses klasifikasi.

Dalam praktiknya, pemilihan fitur dapat dipandang sebagai masalah optimasi kombinatorial. Jika sebuah dataset memiliki n fitur, maka jumlah kemungkinan subset fitur yang dapat dibentuk adalah 2^n. Jumlah kemungkinan ini meningkat sangat cepat ketika n semakin besar. Oleh karena itu, pencarian secara menyeluruh terhadap seluruh kemungkinan subset fitur menjadi tidak praktis. Kondisi ini membuat algoritma optimasi metaheuristik menjadi relevan karena mampu menelusuri ruang solusi yang besar tanpa harus memeriksa seluruh kombinasi fitur.

Salah satu pendekatan yang digunakan pada studi kasus ini adalah Binary Grey Wolf-Whale Optimization (BGWWO)

(Siswantoro et al., 2026), yaitu bentuk biner dari hibrida Grey Wolf Optimizer dan Whale Optimization Algorithm. Metode ini digunakan untuk memilih subset fitur pada tugas klasifikasi. Tujuannya adalah memperoleh subset fitur yang lebih kecil, tetapi tetap mampu menghasilkan akurasi klasifikasi yang tinggi. Dengan demikian, proses pemilihan fitur tidak hanya diarahkan untuk mengurangi jumlah fitur, tetapi juga mempertahankan kualitas prediksi model.

5.4.1 Dataset

Studi kasus ini menggunakan sepuluh dataset klasifikasi publik dari UCI (University of California, Irvine) *Machine Learning Repository*. Dataset tersebut dipilih karena memiliki karakteristik yang beragam dari sisi jumlah fitur, jumlah sampel, dan jumlah kelas. Keragaman ini digunakan untuk menguji kemampuan BGWWO dalam menyelesaikan masalah pemilihan fitur pada berbagai kondisi data. Karakteristik dataset yang digunakan disajikan pada Tabel 5.2

Berdasarkan Tabel 5.2, dataset yang digunakan mencakup data berdimensi rendah, sedang, dan tinggi. Jumlah fitur berkisar dari 10 fitur pada *Page Blocks* sampai 1203 fitur pada *Toxicity*. Jumlah sampel juga bervariasi, mulai dari 158 sampel pada *Kidney* sampai 10992 sampel pada *Digits*. Selain itu, sebagian besar dataset merupakan masalah klasifikasi biner, sedangkan *Digits*, *Page Blocks*, dan *Wine* merupakan masalah klasifikasi multikelas. Variasi ini membuat dataset tersebut sesuai untuk menguji kemampuan BGWWO dalam memilih subset fitur yang efektif pada berbagai karakteristik data.

Tabel 5.2. Karakteristik dataset yang digunakan dalam studi kasus pemilihan fitur.

No.	Nama Dataset	Jumlah Fitur	Jumlah Sampel	Jumlah Kelas
1	Breast Cancer	30	569	2
2	Darwin	624	174	2
3	Digits	16	10992	10

4	Ionosphere	34	351	2
5	Kidney	34	158	2
6	Page Blocks	10	5473	7
7	Spambase	57	4601	2
8	Tic-tac-toe	27	958	2
9	Toxicity	1203	171	2
10	Wine	13	178	3

5.4.2 Proses Pemilihan Fitur

Pada studi kasus ini, pemilihan fitur dilakukan menggunakan BGWWO. Metode ini mengadaptasi hibrida GWO-WOA ke dalam ruang solusi biner. Adaptasi ini diperlukan karena pemilihan fitur membutuhkan keputusan diskrit. Setiap fitur hanya memiliki dua kemungkinan status, yaitu dipilih atau dibuang. Oleh karena itu, setiap solusi direpresentasikan sebagai vektor biner. Jika suatu dataset memiliki n fitur, maka satu agen pencari direpresentasikan sebagai vektor dengan n elemen. Setiap elemen berhubungan dengan satu fitur. Nilai 1 menunjukkan bahwa fitur tersebut dipilih, sedangkan nilai 0 menunjukkan bahwa fitur tersebut dibuang. Sebagai contoh, jika suatu dataset memiliki 10 fitur, maka satu solusi dapat berbentuk vektor biner sepanjang 10 elemen. Vektor tersebut menunjukkan subset fitur yang akan digunakan untuk melatih model klasifikasi.

Proses pemilihan fitur dimulai dengan membangkitkan populasi awal secara acak. Setiap agen dalam populasi membawa satu kandidat subset fitur. Setelah itu, posisi agen diubah ke bentuk biner menggunakan fungsi transfer seperti pada persamaan (5.1) - (5.13) sehingga dapat digunakan untuk menentukan fitur mana yang dipertahankan. Fitur yang bernilai 1 digunakan sebagai input model klasifikasi, sedangkan fitur yang bernilai 0 tidak digunakan.

Model klasifikasi yang digunakan untuk mengevaluasi subset fitur pada studi kasus ini adalah *K-Nearest Neighbors* (KNN). Model

ini dipilih karena sederhana, mudah diimplementasikan, dan sering digunakan dalam studi pemilihan fitur berbasis *wrapper*. Pada setiap kandidat subset fitur, KNN dilatih dan diuji menggunakan fitur yang dipilih oleh agen tersebut. Hasil klasifikasi kemudian digunakan untuk menilai kualitas subset fitur menggunakan fungsi objektif pada persamaan (5.17). Kualitas subset fitur tidak hanya ditentukan oleh akurasi klasifikasi. Jumlah fitur yang dipilih juga menjadi bagian penting. Subset fitur yang baik adalah subset yang mampu menjaga akurasi tetap tinggi dengan jumlah fitur yang lebih sedikit. Oleh karena itu, solusi yang dipilih bukan sekadar solusi dengan fitur paling sedikit, melainkan solusi yang memberikan keseimbangan antara akurasi dan reduksi fitur.

Setelah seluruh agen dievaluasi, solusi terbaik digunakan sebagai acuan dalam pembaruan posisi populasi pada iterasi berikutnya menggunakan hibrida GWO-WOA seperti yang telah dijelaskan pada subbab 3.7. Proses ini dilakukan berulang sampai jumlah iterasi maksimum tercapai. Pada akhir proses, agen terbaik dipilih sebagai solusi akhir. Solusi tersebut berisi subset fitur yang dianggap paling optimal untuk dataset yang sedang diproses. Dengan mekanisme tersebut, BGWWO berfungsi sebagai metode wrapper untuk pemilihan fitur. Artinya, proses seleksi fitur melibatkan performa model klasifikasi secara langsung. Pendekatan ini biasanya membutuhkan biaya komputasi yang lebih besar daripada metode filter, tetapi dapat menghasilkan subset fitur yang lebih sesuai dengan model yang digunakan.

5.4.3 Pengaturan Eksperimen

Eksperimen dimulai dengan memuat dataset yang akan digunakan. Setelah dataset dimuat, tahap berikutnya adalah prapemrosesan data. Pada tahap ini, baris yang memiliki nilai kosong atau hilang dihapus. Fitur kategorikal kemudian diubah ke bentuk numerik agar dapat diproses oleh algoritma optimasi dan model klasifikasi. Tahap ini penting karena algoritma metaheuristik dan KNN membutuhkan representasi numerik yang konsisten.

Setelah prapemrosesan selesai, dataset dibagi menggunakan *stratified k-fold cross-validation* dengan nilai $k = 10$. Teknik ini membagi data menjadi 10 bagian dengan tetap menjaga proporsi kelas pada setiap *fold*. Dengan cara ini, evaluasi menjadi lebih seimbang dan tidak bias terhadap kelas tertentu. Teknik *stratified k-fold* juga membantu memperoleh estimasi performa yang lebih stabil karena model diuji pada beberapa pembagian data. Setelah pembagian data dilakukan, proses optimasi fitur dijalankan menggunakan BGWWO. Beberapa fungsi transfer, yaitu berbentuk-S, berbetuk-V (V_1, V_2, V_3, V_4), dan berbentuk-V terbalik (rV_1, rV_2, rV_3, rV_4), diuji untuk mengubah posisi kontinu agen menjadi keputusan biner. Kinerja setiap fungsi transfer dibandingkan berdasarkan nilai rata-rata *fitness* pada seluruh fold.

Seluruh proses pemilihan fitur dijalankan sebanyak 10 kali. Pengulangan ini dilakukan karena algoritma metaheuristik bersifat stokastik. Hasil satu kali eksekusi belum tentu mewakili performa umum algoritma. Dengan melakukan beberapa pengulangan, nilai rata-rata dapat digunakan untuk memberikan gambaran performa yang lebih stabil. Semua eskperimen menggunakan jumlah agen 10, jumlah iterasi maksimum 100, jumlah pengulangan 10, dan batas posisi setiap agen adalah $[-8, 8]$. Evaluasi dilakukan menggunakan beberapa metrik yang meliputi nilai *fitness*, akurasi klasifikasi, dan jumlah fitur terpilih.

5.4.3 Hasil Eksperimen

Kinerja BGWWO dengan berbagai fungsi transfer berdasarkan fungsi *fitness*, yang mencakup nilai rata-rata, hasil terbaik dan terburuk, serta standar deviasi dari beberapa kali percobaan disajikan pada Tabel 5.3, Tabel 5.4, dan Tabel 5.5. Hasil uji menunjukkan bahwa fungsi transfer berbentuk-V terbalik secara umum menghasilkan nilai *fitness* lebih rendah dibandingkan fungsi berbentuk-S dan berbentuk-V, yang berarti solusi yang diperoleh lebih baik. Pada dataset medis *Breast Cancer*, fungsi transfer rV_1 mencatat rata-rata *fitness* terbaik sebesar 0,0587 dengan standar deviasi 0,0056, lebih baik daripada fungsi transfer S terbaik (0,0678)

dan V terbaik (0,0852). Keunggulan fungsi transfer berbentuk-V terbalik makin terlihat pada dataset *Kidney*, di mana rV_1 (0,0102) dan rV_2 (0,0105) jauh melampaui S_1 (0,0229) dan V_1 (0,0482), dengan hasil individu serendah 0,0088. Untuk dataset *Ionosphere*, rV_2 memberikan rata-rata terbaik (0,0821) dibandingkan S_2 (0,1038) dan V_1 (0,1320), sementara rV_1 dan rV_2 sama-sama mencapai nilai fitness terbaik 0,0729. Pada dataset kecil multi-kelas *Wine*, rV_2 kembali unggul dengan rata-rata 0,0784, lebih rendah dibandingkan S_2 (0,0823) dan V_4 (0,1192), dengan *fitness* terbaik 0,0714 pada rV_1 dan rV_2.

Pada dataset berdimensi tinggi dan kompleks, fungsi transfer berbentuk-V terbalik juga menunjukkan kekuatannya. Pada dataset *Darwin*, rV_2 memperoleh rata-rata *fitness* 0,1252, lebih rendah daripada S_1 (0,1905) dan V_1 (0,2214), dengan nilai terbaik 0,1034 meskipun standar deviasi sedikit lebih tinggi (0,0194). Dataset *Toxicity* menegaskan dominasi fungsi transfer berbentuk-V, di mana rV_1 mencapai rata-rata 0,2642 dibandingkan S_1 (0,3444) dan V_4 (0,3686), dengan performa terbaik 0,2312. Pada dataset *Spambase*, rV_2 mencatat rata-rata terbaik (0,1079) dan rV_3 menghasilkan *fitness* terbaik (0,0971). Untuk dataset *Digits*, semua fungsi relatif mirip di kisaran 0,07, dengan S_1 unggul tipis (0,0693) dan rV_3 mencatat hasil terbaik 0,0674. Pada dataset *Page Blocks*, perbedaan antar fungsi kecil, dengan rata-rata terbaik pada rV_2 dan rV_4 (0,0611), sedikit lebih baik dibandingkan S_2 (0,0613) dan V_4 (0,0627). Sementara itu, pada dataset *Tic-tac-toe*, V_3 (0,1106) unggul atas rV_3 (0,1413), menunjukkan bahwa fungsi V juga dapat bekerja baik pada masalah biner tertentu. Secara keseluruhan, hasil pada Tabel 5.3, Tabel 5.4, dan Tabel 5.5 menegaskan bahwa fungsi transfer berbentuk-V terbalik memberikan kinerja fitness paling konsisten, terutama pada dataset medis dan berdimensi tinggi.

Tabel 5.3. *Fitness* dari BGWWO dengan fungsi transfer berbentuk-S

Dataset	**Metrik**	**S_1**	**S_2**	**S_3**	**S_4**
Breast Cancer	Avg	0.0678	0.0708	0.0804	0.0799
	SD	0.0084	0.0073	0.0057	0.0072
	Best	0.0558	0.0558	0.0704	0.0704
	Worst	0.0799	0.0832	0.0883	0.0899
Darwin	Avg	0.1905	0.1966	0.1986	0.2119
	SD	0.0114	0.0101	0.0106	0.0060
	Best	0.1751	0.1824	0.1877	0.2011
	Worst	0.2144	0.2131	0.2272	0.2239
Digits	Avg	**0.0693**	0.0700	0.0714	0.0724
	SD	0.0009	0.0012	0.0018	0.0036
	Best	0.0681	0.0678	0.0687	0.0678
	Worst	0.0714	0.0718	0.0745	0.0788
Ionosphere	Avg	0.1080	0.1038	0.1181	0.1226
	SD	0.0138	0.0165	0.0104	0.0172
	Best	0.0887	0.0758	0.0971	0.0865
	Worst	0.1287	0.1343	0.1336	0.1420
Kidney	Avg	0.0229	0.0328	0.0484	0.0687
	SD	0.0083	0.0088	0.0160	0.0036
	Best	0.0115	0.0230	0.0321	0.0633
	Worst	0.0432	0.0520	0.0810	0.0750
Page Blocks	Avg	0.0652	0.0613	0.0615	0.0643
	SD	0.0041	0.0006	0.0008	0.0055

	Best	0.0611	0.0611	0.0611	0.0611
	Worst	0.0708	0.0631	0.0631	0.0782
Spambase	Avg	0.1101	0.1180	0.1174	0.1211
	SD	0.0059	0.0094	0.0047	0.0054
	Best	0.1044	0.1071	0.1100	0.1110
	Worst	0.1210	0.1368	0.1260	0.1293
Tic-tac-toe	Avg	0.1270	0.1264	0.1375	0.1402
	SD	0.0109	0.0098	0.0043	0.0099
	Best	0.1111	0.1082	0.1307	0.1260
	Worst	0.1415	0.1383	0.1446	0.1568
Toxicity	Avg	0.3444	0.3540	0.3587	0.3713
	SD	0.0138	0.0116	0.0121	0.0036
	Best	0.3180	0.3367	0.3433	0.3651
	Worst	0.3677	0.3714	0.3762	0.3761
Wine	Avg	0.0828	0.0823	0.1522	0.3043
	SD	0.0062	0.0077	0.0969	0.0090
	Best	0.0738	0.0738	0.0738	0.2937
	Worst	0.0938	0.0964	0.3038	0.3268

Tabel 5.4. Fitness dari BGWWO dengan fungsi transfer berbentuk-V

Dataset	**Metrik**	**V_1**	**V_2**	**V_3**	**V_4**
Breast Cancer	Avg	0.0852	0.0822	0.0817	0.0838
	SD	0.0044	0.0045	0.0061	0.0048
	Best	0.0799	0.0738	0.0706	0.0756

	Worst	0.0951	0.0867	0.0917	0.0899
Darwin	Avg	0.2214	0.2316	0.2294	0.2240
	SD	0.0185	0.0189	0.0217	0.0107
	Best	0.1810	0.1988	0.1945	0.2059
	Worst	0.2586	0.2585	0.2658	0.2384
Digits	Avg	0.0703	0.0698	0.0703	0.0703
	SD	0.0013	0.0012	0.0016	0.0023
	Best	0.0681	0.0685	0.0681	0.0681
	Worst	0.0731	0.0722	0.0730	0.0745
Ionosphere	Avg	0.1320	0.1353	0.1361	0.1334
	SD	0.0183	0.0273	0.0181	0.0154
	Best	0.0971	0.0866	0.1001	0.1057
	Worst	0.1552	0.1629	0.1733	0.1673
Kidney	Avg	0.0482	0.0571	0.0389	0.0461
	SD	0.0128	0.0202	0.0080	0.0119
	Best	0.0289	0.0321	0.0260	0.0262
	Worst	0.0750	0.1087	0.0520	0.0662
Page Blocks	Avg	0.0678	0.0655	0.0654	0.0627
	SD	0.0068	0.0045	0.0052	0.0029
	Best	0.0611	0.0611	0.0611	0.0611
	Worst	0.0805	0.0751	0.0782	0.0711
Spambase	Avg	0.1140	0.1134	0.1156	0.1127
	SD	0.0054	0.0060	0.0028	0.0044
	Best	0.1074	0.1067	0.1098	0.1039

	Worst	0.1280	0.1270	0.1201	0.1186
Tic-tac-toe	Avg	0.1306	0.1228	**0.1106**	0.1369
	SD	0.0246	0.0183	0.0226	0.0048
	Best	0.0920	0.0948	0.0667	0.1270
	Worst	0.1660	0.1538	0.1343	0.1446
Toxicity	Avg	0.3794	0.3792	0.3690	0.3686
	SD	0.0111	0.0114	0.0090	0.0123
	Best	0.3620	0.3626	0.3567	0.3476
	Worst	0.3966	0.4015	0.3863	0.3914
Wine	Avg	0.1406	0.1832	0.1197	0.1192
	SD	0.0831	0.0919	0.0612	0.0599
	Best	0.0761	0.0840	0.0762	0.0738
	Worst	0.3115	0.3115	0.2961	0.2884

Tabel 5.5. *Fitness* dari BGWWO dengan fungsi transfer berbentuk-V terbalik

Dataset	**Metrik**	**rV$_1$**	**rV$_2$**	**rV$_3$**	**rV$_4$**
Breast Cancer	Avg	**0.0587**	0.0626	0.0624	0.0645
	SD	0.0056	0.0058	0.0043	0.0049
	Best	0.0525	0.0525	0.0525	0.0604
	Worst	0.0669	0.0699	0.0669	0.0766
Darwin	Avg	0.1378	**0.1252**	0.1543	0.1779
	SD	0.0160	0.0194	0.0090	0.0123
	Best	0.1095	0.1034	0.1421	0.1539

	Worst	0.1615	0.1588	0.1750	0.1956
Digits	Avg	0.0712	0.0706	0.0705	0.0715
	SD	0.0023	0.0018	0.0022	0.0019
	Best	0.0681	0.0678	0.0674	0.0682
	Worst	0.0760	0.0734	0.0755	0.0747
Ionosphere	Avg	0.0832	**0.0821**	0.0890	0.0925
	SD	0.0068	0.0115	0.0067	0.0063
	Best	0.0729	0.0729	0.0833	0.0836
	Worst	0.0931	0.1038	0.1042	0.1038
Kidney	Avg	**0.0102**	0.0105	0.0143	0.0393
	SD	0.0019	0.0019	0.0034	0.0108
	Best	0.0088	0.0088	0.0118	0.0289
	Worst	0.0144	0.0144	0.0230	0.0603
Page Blocks	Avg	0.0625	**0.0611**	0.0615	**0.0611**
	SD	0.0035	0.0000	0.0012	0.0000
	Best	0.0611	0.0611	0.0611	0.0611
	Worst	0.0728	0.0611	0.0651	0.0611
Spambase	Avg	0.1080	**0.1079**	0.1113	0.1130
	SD	0.0059	0.0051	0.0098	0.0047
	Best	0.1011	0.1010	0.0971	0.1055
	Worst	0.1184	0.1183	0.1306	0.1214
Tic-tac-toe	Avg	0.1635	0.1570	0.1413	0.1446
	SD	0.0201	0.0259	0.0113	0.0166
	Best	0.1307	0.1157	0.1241	0.1223

	Worst	0.2035	0.2157	0.1556	0.1687
Toxicity	Avg	**0.2642**	0.2714	0.2928	0.3232
	SD	0.0168	0.0115	0.0211	0.0067
	Best	0.2312	0.2463	0.2417	0.3135
	Worst	0.2863	0.2856	0.3140	0.3342
Wine	Avg	0.0821	**0.0784**	0.0799	0.2888
	SD	0.0095	0.0069	0.0059	0.0035
	Best	0.0714	0.0714	0.0738	0.2860
	Worst	0.0981	0.0934	0.0885	0.2937

Tabel 5.6, Tabel 5.7, dan

Tabel 5.8 menyajikan perbandingan detail akurasi klasifikasi dan jumlah rata-rata fitur terpilih oleh BGWWO ketika menggunakan fungsi transfer berbentuk S, V, dan V terbalik pada sepuluh dataset. Evaluasi difokuskan pada keseimbangan antara akurasi prediksi dan reduksi dimensi. Pada dataset *Breast Cancer*, rV_1 mencapai akurasi rata-rata 94,36% dengan rata-rata hanya 2,4 fitur dari total 30 (reduksi 92%), lebih baik daripada S_1 (93,87% dengan rata-rata 3,8 fitur) dan V_1 (93,24% dengan rata-rata 7,3 fitur). Pada dataset *Kidney*, rata-rata akurasi rV_1 bahkan mendekati sempurna (99,81%) dengan rata-rata 2,9 fitur (reduksi 91%), mengungguli S_1 (99% dengan rata-rata 4,7 fitur) dan V_3 (98,55% dengan rata-rata 9,6 fitur). Hasil serupa terlihat pada dataset *Ionosphere*, di mana rV_2 memperoleh akurasi tertinggi 91,99% dengan rata-rata 3,4 fitur dari 34 (reduksi 90%), melampaui S_2 (90,59% dengan rata-rata 6,5 fitur) dan V_1 (89,03% dengan rata-rata 11,3 fitur). Dataset kecil seperti *Wine* juga memperlihatkan dominasi fungsi transfer berbentuk-V terbalik, yaitu rV_3 mencatat akurasi 95,14% menggunakan rata-rata 4,7 fitur, lebih baik dibandingkan S_2 (94,47% dengan rata-rata 4,3 fitur) dan V_3 (91,66% dengan rata-rata 5,8 fitur).

Pada dataset berdimensi tinggi, keunggulan fungsi transfer berbentuk-V terbalik semakin menonjol. Di dataset *Darwin* (624 fitur), rV_2 mencapai akurasi rata-rata 87,73% dengan rata-rata 92 fitur (reduksi 85%), jauh lebih efisien dibandingkan S_1 (83,45% dengan rata-rata 259,4 fitur) dan V_1 (82,22% dengan rata-rata 382,5 fitur). Dataset *Toxicity* juga menegaskan kekuatan ini, dengan rV_1 mencapai 72,11% dengan rata-rata 158,4 fitur dari 1203 (reduksi 86,8%), lebih baik daripada S_1 (66,52% dengan rata-rata 517,4 fitur) dan V_3 (64,65% dengan rata-rata 611,1 fitur). Pada dataset *Digits*, akurasi seluruh fungsi tinggi, tetapi rV_4 unggul tipis (97,96% dengan rata-rata 8,5 fitur dari 16). Untuk dataset *Page Blocks*, semua fungsi serupa dengan akurasi sekitar 95%, meskipun S_4 sedikit lebih baik (95,52%). Pada dataset *Spambase*, rV_4 memberikan akurasi 91,30% dengan rata-rata 19,8 fitur, sedikit di bawah V_1 (91,95% dengan rata-rata 23,7 fitur). Sementara pada *Tic-tac-toe*, V_3 unggul dengan akurasi

95,16% tetapi dengan rata-rata 18,1 fitur, sedangkan rV_3 dan rV_4 lebih efisien (akurasi 88,7% dengan rata-rata 11 fitur). Secara keseluruhan, hasil pada Tabel 5.6, Tabel 5.7, dan

Tabel 5.8 menunjukkan bahwa keluarga fungsi transfer berbetuk-V terbalik konsisten unggul dalam menjaga keseimbangan antara akurasi dan reduksi fitur, baik pada dataset kecil, medis, maupun berdimensi tinggi.

Tabel 5.6. Akurasi dan jumlah fitur terpilih dari BGWWO dengan fungsi transfer berbentuk-S

Dataset	**Metrik**	**S_1**	**S_2**	**S_3**	**S_4**
Breast Cancer	Avg akurasi	0.9387	0.9376	0.9339	0.9361
	SD akurasi	0.0077	0.0082	0.0054	0.0058
	Avg jumlah fitur	3.8000	4.4000	6.3000	6.7000
Darwin	Avg akurasi	0.8345	0.8282	0.8284	0.8146
	SD akurasi	0.0135	0.0118	0.0114	0.0050
	Avg jumlah fitur	259.4000	261.9000	275.7000	281.4000
Digits	Avg akurasi	0.9779	0.9778	0.9762	0.9779
	SD akurasi	0.0021	0.0014	0.0029	0.0020
	Avg jumlah fitur	7.9000	8.0000	8.0000	8.4000

Ionosphere	Avg akurasi	0.9032	0.9059	0.8936	0.8902
	SD akurasi	0.0131	0.0137	0.0117	0.0147
	Avg jumlah fitur	7.1000	6.5000	7.6000	8.1000
Kidney	Avg akurasi	0.9900	0.9874	0.9767	0.9576
	SD akurasi	0.0076	0.0075	0.0158	0.0038
	Avg jumlah fitur	4.7000	7.3000	9.3000	10.4000
Page Blocks	Avg akurasi	0.9542	0.9541	0.9539	**0.9552**
	SD akurasi	0.0018	0.0007	0.0009	0.0014
	Avg jumlah fitur	2.4000	**2.0000**	**2.0000**	2.4000
Spambase	Avg akurasi	0.9143	0.9120	0.9133	0.9130
	SD akurasi	0.0085	0.0106	0.0056	0.0047
	Avg jumlah fitur	18.8000	22.1000	22.4000	24.4000

Tic-tac-toe	Avg akurasi	0.9116	0.9245	0.9090	0.9072
	SD akurasi	0.0190	0.0137	0.0043	0.0097
	Avg jumlah fitur	12.8000	15.8000	15.0000	15.3000
Toxicity	Avg akurasi	0.6652	0.6558	0.6511	0.6371
	SD akurasi	0.0155	0.0133	0.0125	0.0043
	Avg jumlah fitur	517.4000	531.5000	537.7000	538.4000
Wine	Avg akurasi	0.9447	0.9547	0.8727	0.7079
	SD akurasi	0.0076	0.0100	0.1071	0.0067
	Avg jumlah fitur	4.3000	5.4000	4.9000	5.6000

Tabel 5.7. Akurasi dan jumlah fitur terpilih dari BGWWO dengan fungsi transfer berbentuk-V

Dataset	**Metrik**	**V_1**	**V_2**	**V_3**	**V_4**
Breast Cancer	Avg akurasi	0.9324	0.9324	0.9315	0.9313
	SD akurasi	0.0016	0.0037	0.0042	0.0037
	Avg jumlah fitur	7.3000	6.4000	6.0000	6.6000

Darwin	Avg akurasi	0.8222	0.8159	0.8140	0.8161
	SD akurasi	0.0116	0.0163	0.0151	0.0093
	Avg jumlah fitur	382.5000	411.6000	387.0000	364.9000
Digits	Avg akurasi	0.9781	0.9766	0.9774	0.9775
	SD akurasi	0.0023	0.0033	0.0033	0.0032
	Avg jumlah fitur	8.1000	**7.8000**	8.0000	8.0000
Ionosphere	Avg akurasi	0.8903	0.8882	0.8840	0.8852
	SD akurasi	0.0105	0.0176	0.0121	0.0113
	Avg jumlah fitur	11.3000	11.8000	10.8000	10.2000
Kidney	Avg akurasi	0.9778	0.9721	0.9855	0.9792
	SD akurasi	0.0111	0.0238	0.0064	0.0127
	Avg jumlah fitur	9.6000	10.9000	8.8000	9.3000
Page Blocks	Avg akurasi	0.9503	0.9551	0.9551	0.9537
	SD akurasi	0.0057	0.0026	0.0021	0.0010
	Avg jumlah fitur	2.3000	2.5000	2.5000	2.1000
Spambase	Avg akurasi	**0.9195**	0.9187	**0.9195**	0.9188
	SD akurasi	0.0066	0.0030	0.0040	0.0030
	Avg jumlah fitur	23.7000	22.9000	24.6000	22.6000
Tic-tac-toe	Avg akurasi	0.9380	0.9418	**0.9516**	0.9170
	SD akurasi	0.0211	0.0171	0.0245	0.0040
	Avg jumlah fitur	20.2000	19.0000	18.1000	16.8000

Toxicity	Avg akurasi	0.6411	0.6424	0.6465	0.6436
	SD akurasi	0.0088	0.0083	0.0071	0.0126
	Avg jumlah fitur	678.7000	689.1000	611.1000	575.5000
Wine	Avg akurasi	0.8908	0.8529	0.9166	0.9154
	SD akurasi	0.0933	0.1023	0.0694	0.0703
	Avg jumlah fitur	5.5000	6.6000	5.8000	5.6000

Tabel 5.8. Akurasi dan jumlah fitur terpilih dari BGWWO dengan fungsi transfer berbentuk -V terbalik

Dataset	**Metric**	**rV_1**	**rV_2**	**rV_3**	**rV_4**
Breast Cancer	Avg akurasi	**0.9436**	0.9397	0.9403	0.9383
	SD akurasi	0.0054	0.0058	0.0034	0.0034
	Avg jumlah fitur	**2.4000**	2.5000	2.6000	2.7000
Darwin	Avg akurasi	0.8618	**0.8773**	0.8488	0.8285
	SD akurasi	0.0180	0.0214	0.0105	0.0139
	Avg jumlah fitur	**84.0000**	92.0000	113.8000	146.7000
Digits	Avg akurasi	0.9785	0.9778	0.9794	**0.9796**
	SD akurasi	0.0038	0.0031	0.0032	0.0020
	Avg jumlah fitur	8.3000	8.1000	8.3000	8.5000
Ionosphere	Avg akurasi	0.9190	**0.9199**	0.9135	0.9116
	SD akurasi	0.0078	0.0123	0.0071	0.0070
	Avg jumlah fitur	3.5000	**3.4000**	3.8000	4.4000
Kidney	Avg akurasi	**0.9981**	0.9975	0.9969	0.9773
	SD akurasi	0.0029	0.0031	0.0042	0.0096
	Avg jumlah fitur	2.9000	**2.8000**	3.9000	6.4000
Page Blocks	Avg akurasi	0.9528	0.9543	0.9539	0.9543
	SD akurasi	0.0039	0.0000	0.0013	0.0000

	Avg jumlah fitur	**2.0000**	**2.0000**	**2.0000**	**2.0000**
Spambase	Avg akurasi	0.9112	0.9092	0.9097	0.9130
	SD akurasi	0.0083	0.0087	0.0112	0.0063
	Avg jumlah fitur	16.0000	**14.9000**	17.1000	19.8000
Tic-tac-toe	Avg akurasi	0.8591	0.8671	0.8866	0.8875
	SD akurasi	0.0305	0.0373	0.0246	0.0299
	Avg jumlah fitur	**9.9000**	10.1000	10.6000	11.7000
Toxicity	Avg akurasi	**0.7211**	0.7142	0.6934	0.6664
	SD akurasi	0.0197	0.0139	0.0234	0.0086
	Avg jumlah fitur	**158.4000**	170.5000	202.7000	276.6000
Wine	Avg akurasi	0.9396	0.9496	**0.9514**	0.7073
	SD akurasi	0.0138	0.0136	0.0109	0.0017
	Avg jumlah fitur	3.6000	4.3000	4.7000	**3.3000**

06

Aplikasi *Grey Wolf Optimizer* dalam Pelatihan Model Pembelajaran Mesin

Pelatihan model pembelajaran mesin pada dasarnya merupakan proses mencari nilai parameter internal model yang paling sesuai agar keluaran model mendekati target yang diharapkan. Pada model seperti *Artificial Neural Network* (ANN), parameter internal tersebut umumnya berupa bobot dan bias yang menghubungkan neuron-neuron di dalam jaringan. Kualitas parameter ini sangat menentukan kemampuan model dalam melakukan memprediksi output. Oleh sebab itu, pelatihan model dapat dipandang sebagai masalah optimasi, yaitu mencari kombinasi parameter yang meminimalkan kesalahan prediksi atau meningkatkan kualitas keluaran model (Hemeida et al., 2020).

Dalam berbagai penelitian, GWO tidak hanya digunakan untuk optimasi *hyperparameter* dan pemilihan fitur, tetapi juga dimanfaatkan secara langsung untuk melatih model dengan mengoptimasi parameter. Dalam konteks ini, yang dioptimasi oleh GWO bukan lagi learning rate, jumlah layer, ukuran kernel, atau subset fitur, melainkan bobot, bias, atau parameter inti lain yang membentuk fungsi model. Dengan cara tersebut, proses pelatihan berubah menjadi pencarian solusi terbaik di ruang parameter model.

Setiap kandidat solusi dinilai berdasarkan *error* atau performa model, lalu diperbaiki secara iteratif menggunakan mekanisme sosial dan strategi berburu serigala abu-abu.

Penerapan GWO untuk melatih model pembelajaran mesin telah dilaporkan pada berbagai jenis tugas baik klasifikasi maupun prediksi. Beberapa diantaranya adalah penggunaan GWO sebagai algoritma pelatihan pada *Multi-Layer Perceptron* (MLP) atau ANN pada beberapa dataset publik, yaitu untuk mengoptimasi bobot dan bias jaringan secara langsung agar diperoleh rata-rata MSE yang lebih kecil serta performa klasifikasi dan aproksimasi fungsi yang lebih baik (Hemeida et al., 2020; Mirjalili, 2015b). Pada penelitian lain, GWO diterapkan untuk melatih *Elman Neural Network* (ENN) yang merupakan salah satu bentuk *Recurent Neural Network* (RNN) untuk tugas klasifikasi serta prediksi pada data runtun waktu (Rabhi et al., 2017). Selain itu GWO juga digunakan untuk mengoptimasi bobot ANN dalam prediksi kecepatan angin (Cinar and Natarajan, 2022), klasifikasi data sonar (Mosavi et al., 2016). Aplikasi GWO lain adalah melatih model ANN dan dan *Multiple linear regression* (MLR) untuk meramal debit aliran sungai (Tikhamarine et al., 2020). Pada model *Extreme Learning Machine* (ELM) untuk prediksi kuat tekan beton GWO juga digunakan untuk mengoptimasi bobot dan pada antara lapisan input dan lapisan tersembunyi yang umumnya dibangkitkan secara acak (Shariati et al., 2022).

GWO sesuai untuk melatih model pembelajaran mesin karena memiliki struktur yang sederhana, jumlah parameter pengendali yang sedikit, dan mekanisme pencarian yang relatif mudah diimplementasikan. GWO bekerja dengan membagi populasi ke dalam empat kelompok, yaitu α, β, δ, dan ω, dengan tiga agen terbaik pertama berperan sebagai pemimpin pencarian. Dalam konteks pelatihan model, tiga agen elit ini dapat dipahami sebagai tiga kandidat kombinasi parameter model terbaik pada suatu iterasi, lalu digunakan untuk memandu kandidat lain menuju wilayah solusi yang lebih menjanjikan. Struktur seperti ini membuat GWO mudah

diadaptasi sebagai pelatih model tanpa memerlukan kerangka tambahan yang terlalu kompleks.

GWO menjadi semakin relevan dalam pelatihan model ANN karena mampu menjawab beberapa keterbatasan pendekatan optimasi berbasis gradien, terutama ketergantungannya pada titik awal, kecenderungannya terjebak pada optimum lokal, dan belum tentu tercapainya solusi global terbaik. Selain itu, hasil pelatihan dapat berubah ketika inisialisasi diulang dengan nilai awal yang berbeda, sementara jumlah percobaan yang diperlukan untuk mendekati solusi global juga tidak dapat ditentukan secara pasti. Dalam kondisi seperti ini, pendekatan stokastik berbasis populasi seperti GWO menawarkan alternatif yang lebih kuat karena pencarian solusi dilakukan melalui banyak kandidat parameter secara simultan, sehingga ruang solusi dapat dijelajahi lebih luas dan risiko stagnasi pada solusi lokal dapat dikurangi (Mirjalili, 2015b).

Keunggulan lain GWO terletak pada kemampuannya menjaga keseimbangan antara eksplorasi dan eksploitasi. Dalam pelatihan model, eksplorasi berarti menjelajahi berbagai kemungkinan kombinasi bobot dan bias di ruang parameter, sedangkan eksploitasi berarti memperhalus pencarian di sekitar kombinasi parameter yang sudah menghasilkan error kecil. Sejumlah penelitian menunjukkan bahwa GWO memiliki kemampuan pencarian yang baik dalam menyeimbangkan dua proses tersebut, sehingga dapat membantu model mencapai solusi yang lebih baik dan lebih stabil. Keseimbangan ini menjadi penting karena ruang parameter model umumnya luas, nonlinier, dan dapat memiliki banyak titik solusi lokal (Mirjalili, 2015b).

GWO juga menarik untuk pelatihan model karena bersifat *derivative-free*, yaitu tidak memerlukan turunan fungsi objektif untuk melakukan pencarian solusi. Dalam praktiknya, hal ini memberi keuntungan ketika model yang dilatih memiliki lanskap error yang kompleks atau sulit diturunkan secara analitis. Dengan memanfaatkan pencarian populasi yang stokastik, GWO dapat bergerak di ruang parameter hanya berdasarkan evaluasi kualitas solusi, tanpa perlu

menghitung informasi gradien. Oleh sebab itu, GWO sangat relevan pada situasi di mana pelatihan model dipandang lebih efektif jika diformulasikan sebagai masalah optimasi global (Hemeida et al., 2020; Mirjalili, 2015b; Rabhi et al., 2017). Rangkuman aplikasi GWO dalam pelatihan model pembelajaran mesin dapat dilihat pada .

Tabel 6.1. Rangkuman aplikasi GWO dalam pelatihan model pembelajaran mesin.

Tugas	**Model**	**Parameter yang dilatih**	**Fungsi objektif**
Klasifikasi dan aproksimasi fungsi (Mirjalili, 2015b)	MLP	Bobot dan bias model	MSE
Klasifikasi data sonar (Mosavi et al., 2016)	MLP	Bobot dan bias model	MSE
Prediksi deret waktu dan klasifikasi data (Rabhi et al., 2017)	ENN	Bobot dan bias model, termasuk koneksi *context layer*	MSE
Prediksi kecepatan angin (Cinar and Natarajan, 2022)	MLP	Bobot dan bias model	MSE
Klasifikasi (Hemeida et al., 2020)	MLP	Bobot dan bias model	MSE
Prediksi debit aliran Sungai (Tikhamarine et al., 2020)	MLP, MLR	Bobot dan bias model	RMSE
Prediksi kuat tekan beton (Shariati et al., 2022)	ELM	Bobot dan bias ELM antara *input layer* dan *hidden layer*	RMSE

Secara umum, tahapan pelatihan model pembelajaran mesin menggunakan GWO dimulai dengan merepresentasikan seluruh parameter model sebagai vektor solusi. Sehingga satu serigala merepresentasikan satu kandidat konfigurasi parameter model. Setelah itu, populasi awal serigala dibangkitkan secara acak pada ruang pencarian parameter yang telah ditentukan. Setiap kandidat parameter kemudian dievaluasi dengan memasukkan setiap elemen serigala ke dalam model untuk memprediksi output. Output prediksi kemudian dibandingkan dengan target output untuk menghitung nilai fungsi objektif. Berdasarkan nilai fungsi objektif tersebut, tiga solusi terbaik ditetapkan sebagai serigala α, β, dan δ, lalu digunakan untuk memandu pembaruan posisi serigala lain sesuai mekanisme GWO. Proses evaluasi dan pembaruan posisi ini dilakukan secara iteratif hingga jumlah iterasi maksimum tercapai, dan solusi terbaik pada akhir proses dipilih sebagai konfigurasi parameter model hasil pelatihan.

6.1 Representasi Serigala dalam Pelatihan Model Pembelajaran Mesin

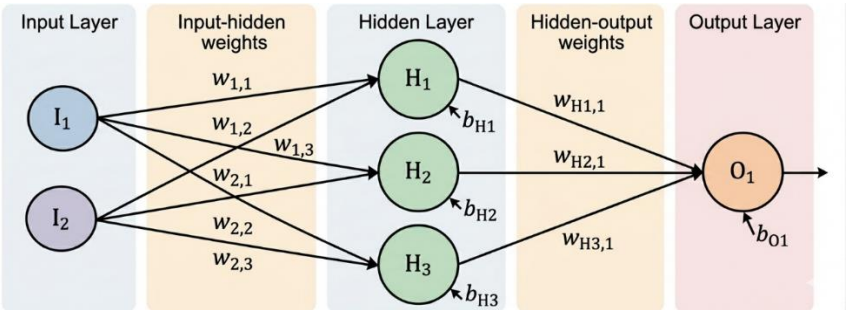
Pada pelatihan model pembelajaran mesin menggunakan GWO, setiap serigala direpresentasikan sebagai vektor parameter model. Jika suatu model memiliki D parameter yang harus dilatih, maka posisi seekor serigala dapat dituliskan sebagai vektor berdimensi D seperti pada persamaan (4.1), yang setiap komponennya menyatakan satu parameter model, misalnya bobot atau bias. Dengan demikian, serigala tidak lagi berupa agen pada ruang fisik, melainkan sebagai satu kandidat konfigurasi parameter yang dapat langsung digunakan untuk menjalankan model. Dalam kerangka ini, ruang pencarian GWO berubah menjadi ruang seluruh kemungkinan kombinasi parameter model.

Pada model MLP, representasi tersebut umumnya dibangun dengan menggabungkan seluruh bobot dan bias dari semua koneksi jaringan ke dalam satu vektor solusi. Bobot dari *input layer* ke *hidden*

layer, bobot dari *hidden layer* ke *output layer*, bias pada *hidden layer*, dan bias pada *output layer* semuanya menjadi bagian dari vektor yang dioptimasi. Dengan cara ini, satu serigala secara langsung merepresentasikan satu konfigurasi lengkap parameter MLP. Setiap perubahan posisi serigala berarti perubahan nilai bobot dan bias yang digunakan oleh jaringan untuk menghasilkan output.

Sebagai contoh, Gambar 6.1 menunjukkan arsitektur MLP sederhana yang terdiri atas dua neuron pada *input layer*, yaitu I_1 dan I_2, tiga neuron pada *hidden layer*, yaitu H_1, H_2, dan H_3, serta satu neuron pada *output layer*, yaitu O_1. Pada arsitektur ini, seluruh neuron pada *input layer* terhubung penuh ke seluruh neuron pada *hidden layer* melalui bobot $w_{1,1}$, $w_{1,2}$, $w_{1,3}$, $w_{2,1}$, $w_{2,2}$, dan $w_{2,3}$. Setiap neuron pada *hidden layer* juga memiliki bias masing-masing, yaitu b_{H1}, b_{H2}, dan b_{H3}. Selanjutnya, ketiga neuron pada *hidden layer* terhubung ke neuron output melalui bobot $w_{H1,1}$, $w_{H2,1}$, dan $w_{H3,1}$, sedangkan neuron output memiliki bias b_{O1}. Dengan demikian, seluruh parameter pada arsitektur ini dapat digabungkan ke dalam satu vektor solusi yang merepresentasikan satu serigala, seperti pada persamaan (6.1).

$$\vec{X} = [w_{1,1}, w_{1,2}, w_{1,3}, w_{2,1}, w_{2,2}, w_{2,3}, b_{H1}, b_{H2}, b_{H3}, w_{H1,1}, w_{H2,1}, w_{H3,1}, b_{O1}] \quad (6.1)$$



Gambar 6.1. Contoh arsitektu MLP lengkap dengan bobot dan bias.

Posisi serigala yang diperoleh kemudian digunakan untuk memprediksi output untuk setiap data input pada data latihan melalui proses *feedforward* pada jaringan. Pada arsitektur MLP pada Gambar 6.1, setiap data input I_1dan I_2 terlebih dahulu dipropagasikan ke

hidden layer melalui jumlahan terboboti nilai-nilai input dengan bobot $w_{1,1}$, $w_{1,2}$, $w_{1,3}$, $w_{2,1}$, $w_{2,2}$, dan $w_{2,3}$., dan bias masing-masing neuron b_{H1}, b_{H2}, dan b_{H3}. Dengan demikian, masukan bersih pada neuron H_1, H_2, dan H_3 masing dapat dinyatakan pada persamaan (6.2), (6.3), dan (6.4).

$$net_{H1} = w_{1,1}I_1 + w_{2,1}I_2 + b_{H1} \quad (6.2)$$
$$net_{H2} = w_{1,2}I_1 + w_{2,2}I_2 + b_{H2} \quad (6.3)$$
$$net_{H3} = w_{1,3}I_1 + w_{2,3}I_2 + b_{H3} \quad (6.4)$$

Nilai masukan bersih tersebut kemudian diproses menggunakan fungsi aktivasi *hidden layer* $f(\cdot)$ untuk menghasilkan output dari *hidden layer*, seperti pada persamaan (6.5).

$$H_1 = f(net_{H1}), H_2 = f(net_{H2}), H_3 = f(net_{H3}) \quad (6.5)$$

Output dari *hidden layer* selanjutnya dipropagasikan ke *output layer* melalui jumlahan terboboti nilai-nilai output *hidden layer* dengan bobot $w_{H1,1}$, $w_{H2,1}$, dan $w_{H3,1}$, serta bias output b_{O1}. Oleh karena itu, masukan bersih pada neuron output O_1 dapat dituliskan seperti pada persamaan (6.6).

$$net_{O1} = w_{H1,1}H_1 + w_{H2,1}H_2 + w_{H3,1}H_3 + b_{O1} \quad (6.6)$$

Setelah itu, output akhir jaringan diperoleh dengan menerapkan fungsi aktivasi *output layer* $g(\cdot)$ pada nilai masukan bersih tersebut, seperti pada persamaan (6.7).

$$O_1 = g(net_{O1}) \quad (6.7)$$

Dari formulasi ini terlihat bahwa seluruh output jaringan sepenuhnya ditentukan oleh nilai bobot dan bias yang tersimpan dalam vektor solusi $\vec{X}$. Karena itu, ketika GWO memperbarui posisi serigala, yang sesungguhnya diperbarui adalah parameter pada persamaan *feedforward* tersebut, sehingga prediksi output jaringan untuk setiap data input juga ikut berubah. Output yang dihasilkan ini kemudian dibandingkan dengan *target output* untuk menghitung nilai

kesalahan model, yang selanjutnya digunakan sebagai dasar dalam mengevaluasi fungsi objektif.

6.2 Fungsi Objektif dalam Pelatihan Model Pembelajaran Mesin

Dalam pelatihan model pembelajaran mesin menggunakan GWO, fungsi objektif digunakan untuk menilai seberapa baik parameter model yang direpresentasikan oleh seekor serigala. Karena setiap serigala merepresentasikan satu konfigurasi bobot dan bias, maka kualitasnya harus diukur dari performa model ketika parameter tersebut diterapkan pada data. Secara umum, fungsi objektif pada konteks ini berperan sebagai ukuran kesalahan atau ukuran kualitas keluaran model, sehingga GWO dapat membedakan mana solusi yang lebih baik, mana yang harus dipertahankan, dan mana yang perlu diperbarui pada iterasi berikutnya. Dengan demikian, fungsi objektif menjadi penghubung antara ruang parameter model dan kualitas prediksi yang dihasilkan model tersebut.

Pada beberapa penelitian dilaporkan bahwa fungsi objektif untuk pelatihan model pembelajaran mesin menggunakan GWO yang paling umum digunakan adalah *Mean Square Error* (MSE) dan *Root Mean Square Error* (RMSE). Kedua fungsi tersebut termasuk ke dalam keluarga fungsi objektif berbasis error kuadrat, sehingga sama-sama digunakan untuk mengukur selisih antara keluaran model dan target yang diharapkan. Karena itu, baik MSE maupun RMSE secara alami lebih sesuai untuk masalah regresi atau prediksi kontinu, di mana kualitas model dinilai dari seberapa dekat nilai prediksi terhadap nilai aktual.

Secara umum, MSE mengukur rata-rata kuadrat selisih antara hasil prediksi model dan target sebenarnya dan dinyatakan seperti pada persamaan (6.8),

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(O_i - P_i)^2 \tag{6.8}$$

dengan n menyatakan jumlah sampel pada data pelatihan , O_i dan P_i masing-masing adalah nilai target dan nilai prediksi model pada sampel ke-i. Karena MSE menggunakan kuadrat selisih, nilai MSE yang besar akan diberi penalti lebih kuat dibanding nilai kecil. Oleh sebab itu, minimisasi MSE berarti mendorong model agar menghasilkan keluaran yang sedekat mungkin dengan target.

Pada beberapa model ANN, bentuk MSE dituliskan dengan notasi yang lebih spesifik sesuai struktur jaringannya, tetapi maknanya tetap sama. Sebagai contoh, pada model ANN dengan lebih dari satu neuron pada *output layer*, MSE dinyatakan seperti pada persamaan (6.9),

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(O_i^j - P_i^j\right)^2 \tag{6.9}$$

dengan n menyatakan jumlah sampel pelatihan, m jumlah neuron pada *output layer*, O_i^j dan P_i^j masing-masing adalah nilai target dan nilai prediksi model pada sampel ke-i untuk neuron ke-j Walaupun notasinya berbeda dari bentuk umum MSE pada persamaan (6.8), hakikatnya tetap sama, yaitu menghitung rata-rata kuadrat selisih antara prediksi model dan target aktual. Dengan demikian, variasi notasi seperti ini tidak perlu dipandang sebagai fungsi objektif yang berbeda, melainkan sebagai penyesuaian bentuk matematis terhadap struktur model yang dilatih.

Di lain pihak, fungsi objektif RMSE pada dasarnya merupakan akar kuadrat dari MSE. Untuk output tunggal, RMSE dinyatakan seperti pada persamaan (6.10) berikut.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(O_i - P_i)^2} \tag{6.10}$$

Untuk model ANN dengan output lebih dari satu, RMSE juga dapat dipahami sebagai akar dari MSE pada seluruh neuron keluaran, seperti pada persamaan (6.11).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(O_i^j - P_i^j\right)^2} \quad (6.11)$$

Bentuk ini menunjukkan bahwa RMSE tidak berbeda secara prinsip dari MSE, melainkan hanya menambahkan operasi akar pada nilai selisih kuadrat rata-ratanya. Oleh sebab itu, baik pada output tunggal maupun output banyak, RMSE dapat dipahami sebagai bentuk turunan dari MSE yang dibuat lebih mudah untuk diinterpretasikan karena berada pada satuan yang sama dengan data aslinya. Detail algoritma pelatihan model pembelajaran mesin menggunakan GWO dapat dilihat pada Algoritma 6.1.

Algoritma 6.1. Pelatihan model pembelajaran mesin menggunakan GWO

Input:

Dataset pelatihan X dengan output target O, model pembelajaran mesin yang akan dilatih, jumlah parameter model D, fungsi objektif $f(\vec{X})$, ukuran populasi serigala N, jumlah iterasi maksimum T, batas bawah $\overrightarrow{lb}$, dan batas atas $\overrightarrow{ub}$.

Output:

Parameter model terbaik $\vec{X}_\alpha$ dan nilai *fitness* terbaik.

Prosedur:

1. Bangkitkan populasi awal sebanyak N serigala. Setiap serigala direpresentasikan sebagai vektor berdimensi D, dengan setiap elemen merepresentasikan parameter model.
2. Hitung nilai *fitness* setiap serigala menggunakan langkah-langkah berikut:
   2.1. Masukkan setiap elemen serigala $\vec{X}$ ke dalam parameter model.
   2.2. Masukkan seluruh data pelatihan ke dalam model untuk menghitung output prediksi P.

2.3. Hitung *fitness* dengan memasukkan output target dan output prediksi seluruh data pelatihan ke dalam salah satu fungsi pada persamaan (6.8) – (6.11).

3. Tentukan tiga serigala terbaik dalam populasi sebagai $\vec{X}_\alpha$, $\vec{X}_\beta$, dan $\vec{X}_\delta$.
4. Tetapkan iterasi awal $t \leftarrow 1$.
5. Hitung nilai a, $\vec{A}$, dan $\vec{C}$ menggunakan persamaan (2.5), (2.3), dan (2.4).
6. Selama $t \leq T$, lakukan langkah-langkah berikut:

   6.1. Untuk setiap serigala ke-$i = 1,2, \ldots, N$, lakukan:

   a. Perbarui posisi serigala ke-i menggunakan persamaan (2.12).

   b. Jika posisi baru berada di luar batas pencarian, lakukan penyesuaian agar tetap berada pada rentang $[\overrightarrow{lb}, \overrightarrow{ub}]$.

   6.2. Hitung kembali nilai fitness setiap serigala menggunakan langkah 2.

   6.3. Perbarui nilai a, $\vec{A}$, dan $\vec{C}$ menggunakan persamaan (2.5), (2.3), dan (2.4).

   6.4. Perbarui $\vec{X}_\alpha$, $\vec{X}_\beta$, dan $\vec{X}_\delta$ berdasarkan tiga nilai fitness terbaik pada populasi.

   6.5. Set $t \leftarrow t + 1$.

7. Kembalikan parameter terbaik $\vec{X}_\alpha$ dan nilai *fitness* terbaik.

6.3 Studi Kasus Pelatihan ANN Menggunakan GWO

Artificial Neural Network (ANN) merupakan salah satu model pembelajaran mesin yang banyak digunakan untuk klasifikasi pola, prediksi, dan aproksimasi fungsi. Salah satu bentuk ANN yang paling umum digunakan adalah *Multi-Layer Perceptron Neural*

Network (MLP NN). Model ini mampu menangani hubungan nonlinier antara input dan output, sehingga sesuai untuk berbagai masalah klasifikasi yang tidak dapat diselesaikan secara sederhana menggunakan pemisah linier.

Kemampuan utama ANN terletak pada proses pembelajaran. Dalam proses tersebut, model berusaha menyesuaikan bobot dan bias agar output yang dihasilkan semakin mendekati target yang diharapkan. Secara umum, pelatihan ANN banyak dilakukan menggunakan algoritma berbasis gradien, seperti *Back Propagation Gradient Descent*. Namun, pendekatan berbasis gradien memiliki beberapa keterbatasan. Proses konvergensinya dapat berjalan lambat, hasilnya dapat bergantung pada nilai awal, dan model dapat terjebak pada minimum lokal. Selain itu, pemilihan l*earning rate* yang tidak tepat akan membuat proses pelatihan tidak stabil.

Keterbatasan tersebut mendorong penggunaan algoritma optimasi metaheuristik sebagai alternatif untuk melatih ANN. Dalam pendekatan ini, pelatihan ANN tidak dilakukan dengan menghitung gradien, tetapi dengan mencari kombinasi bobot dan bias terbaik melalui proses pencarian berbasis populasi. Setiap kandidat solusi merepresentasikan satu konfigurasi bobot dan bias ANN. Kualitas kandidat solusi dinilai berdasarkan kesalahan prediksi yang dihasilkan oleh jaringan. Pada studi kasus ini, GWO digunakan untuk melatih MLP NN (Mosavi et al., 2016). Model yang diperoleh dari pelatihan menggunakan GWO disebut *Gray Wolf Classifier* (GWC). GWO dipilih karena memiliki kemampuan yang baik dalam menyeimbangkan proses eksplorasi dan eksploitasi. Keseimbangan ini penting karena ruang pencarian bobot dan bias ANN biasanya luas, nonlinier, dan memiliki banyak kemungkinan titik local minimum.

6.3.1 Dataset

Studi kasus ini menggunakan tiga dataset klasifikasi, yaitu *Iris*, *Lenses*, dan *Sonar*. Ketiga dataset tersebut dipilih karena memiliki ukuran, jumlah fitur, jumlah sampel, dan karakteristik

atribut yang berbeda. Dengan menggunakan dataset yang beragam, kemampuan GWO dalam melatih ANN dapat dievaluasi secara lebih menyeluruh. Karakteristik dataset yang digunakan disajikan pada Tabel 5.9.

Tabel 5.9. Karakteristik dataset yang digunakan.

No.	Dataset	Karakteristik Fitur	Jumlah Fitur	Jumlah Sampel
1	Iris	Riil	4	150
2	Lenses	Kategorikal	4	24
3	Sonar	Riil	60	208

Dataset *Iris* digunakan untuk mengklasifikasikan tiga jenis bunga *iris*, yaitu *Setosa*, *Versicolor*, dan *Virginica*. Dataset ini memiliki empat fitur, yaitu panjang sepal, lebar sepal, panjang petal, dan lebar petal. Dataset *Lenses* digunakan untuk menentukan jenis lensa kontak yang sesuai bagi pasien. Kelas pada dataset ini mencakup *hard contact lenses*, *soft contact lenses*, dan *no lenses*. Dataset *Sonar* digunakan untuk membedakan sinyal pantulan dari silinder logam dan batu. Dataset *Sonar* memiliki 60 fitur dan 208 sampel, sehingga menjadi dataset yang lebih kompleks dibandingkan *Iris* dan *Lenses*.

Pada dataset *Sonar*, data berasal dari eksperimen pengiriman sinyal *sonar* ke objek bawah laut. Objek yang digunakan adalah silinder logam sebagai target dan batu sebagai *clutter*. Dari 1200 sinyal pantulan yang diterima, dipilih 208 sinyal dengan *Signal-to-Noise Ratio* (SNR) antara 15 dB sampai 4 dB. Dari jumlah tersebut, 111 sinyal berasal dari silinder logam dan 97 sinyal berasal dari batu. Karena pola sinyal pantulan dari silinder logam dan batu memiliki kemiripan yang tinggi, proses klasifikasi pada dataset ini menjadi lebih sulit. Untuk mengurangi kompleksitas pada dataset Sonar, dimensi data direduksi terlebih dahulu menggunakan *Principal Component Analysis* (PCA). Jumlah fitur awal sebanyak 60 dikurangi

menjadi 9 fitur. Jumlah 9 fitur tersebut diperoleh secara eksperimental. Dengan reduksi dimensi ini, proses pelatihan ANN menjadi lebih sederhana dan risiko redundansi fitur dapat dikurangi.

6.3.2 Arsitektur ANN

Arsitektur ANN yang digunakan dalam studi kasus ini adalah MLP NN dengan satu hidden layer. Secara umum, arsitektur MLP terdiri atas tiga bagian utama, yaitu *input layer*, *hidden layer*, dan *output layer*. *Input layer* menerima fitur dari dataset. *Hidden layer* memproses kombinasi fitur melalui bobot, bias, dan fungsi aktivasi. *Output layer* menghasilkan keluaran akhir yang digunakan untuk menentukan kelas. MLP NN termasuk jenis *feed-forward neural network* karena aliran informasi bergerak satu arah, yaitu dari *input layer* menuju *hidden layer*, kemudian menuju *output layer*. Tidak terdapat koneksi balik dari layer berikutnya ke layer sebelumnya.

Jumlah *node* pada *input layer* mengikuti jumlah fitur yang digunakan. Jika dataset memiliki n fitur, maka jumlah *node* pada *input layer* adalah n. Jumlah *node* pada *output layer* mengikuti jumlah kelas. Jika dataset memiliki m kelas, maka jumlah *node* pada *output layer* adalah m. Penentuan jumlah *node* pada *hidden layer* tidak memiliki aturan baku yang berlaku umum untuk semua kasus klasifikasi. Oleh karena itu, studi kasus ini mengikuti pendekatan yang digunakan oleh Mirjalili dan Lewis (2013) untuk menentukan jumlah *node* pada *hidden layer* dengan mempertimbangkan struktur MLP NN. Jumlah *node* pada *hidden layer* dihitung menggunakan persamaan (6.12),

$$h = 2n + 1 \tag{6.12}$$

dengan n menunjukkan jumlah *node* input, sedangkan h menunjukkan jumlah *node* yang digunakan pada *hidden layer*. Berdasarkan aturan tersebut, arsitektur MLP NN untuk dataset *Iris* adalah 4-9-3. Angka 4 menunjukkan jumlah *node input*, angka 9 menunjukkan jumlah *node* pada *hidden layer*, dan angka 3 menunjukkan jumlah *node output*. Dataset *Lenses* juga menggunakan

arsitektur 4-9-3, sedangkan dataset *Sonar* menggunakan arsitektur 9-19-2 setelah proses reduksi dimensi menggunakan PCA.

Pada studi kasus ini, fungsi aktivasi yang digunakan pada *hidden layer* dan *output layer* adalah fungsi tanh. Fungsi ini mengubah nilai input menjadi nilai output dalam rentang tertentu. Dengan fungsi aktivasi nonlinier, MLP NN dapat membentuk hubungan nonlinier antara fitur input dan kelas output. Hal ini penting karena masalah klasifikasi seperti Sonar tidak memiliki pola pemisahan yang sederhana.

Komponen paling penting dalam MLP NN adalah bobot dan bias. Bobot menentukan kekuatan hubungan antar neuron, sedangkan bias membantu menggeser nilai aktivasi neuron. Output akhir jaringan sangat dipengaruhi oleh kombinasi bobot dan bias tersebut. Oleh karena itu, proses pelatihan MLP NN pada dasarnya merupakan proses mencari nilai bobot dan bias terbaik agar jaringan mampu menghasilkan output yang sesuai dengan target.

6.3.3 Pelatihan ANN

Pelatihan ANN menggunakan GWO dilakukan dengan cara mengoptimasi bobot dan bias jaringan secara langsung. Dalam pendekatan ini, GWO tidak digunakan untuk mencari struktur jaringan atau mengatur parameter algoritma pelatihan berbasis gradien. GWO digunakan untuk mencari kombinasi bobot dan bias yang menghasilkan error paling kecil. Agar GWO dapat digunakan untuk melatih ANN, seluruh bobot dan bias jaringan harus direpresentasikan sebagai satu vektor solusi. Dengan demikian, satu serigala merepresentasikan satu konfigurasi lengkap bobot dan bias MLP NN. Jika posisi serigala berubah, maka nilai bobot dan bias yang digunakan oleh MLP NN juga berubah. Setiap perubahan tersebut menghasilkan performa jaringan yang berbeda.

Representasi vektor dipilih karena arsitektur MLP yang digunakan tidak terlalu kompleks. Dalam representasi ini, semua bobot dari *input layer* ke *hidden layer*, semua bobot dari *hidden layer* ke *output layer*, bias pada *hidden layer*, dan bias pada *output layer*

digabungkan secara berurutan ke dalam satu vektor. Vektor inilah yang menjadi posisi serigala dalam ruang pencarian.

Setiap serigala dievaluasi dengan cara memasukkan nilai bobot dan bias yang dibawanya ke dalam MLP NN. Setelah itu, MLP digunakan untuk menghasilkan output berdasarkan data input. Output prediksi kemudian dibandingkan dengan output target. Selisih antara output prediksi dan target digunakan untuk menghitung nilai error. Semakin kecil nilai error, semakin baik kualitas serigala tersebut sebagai kandidat solusi.

Selama proses iterasi, GWO memperbarui posisi setiap serigala berdasarkan solusi terbaik yang ditemukan. Tiga kandidat terbaik digunakan untuk memandu pencarian solusi berikutnya. Proses ini dilakukan sampai kriteria berhenti tercapai. Pada akhir proses, solusi terbaik dipilih sebagai konfigurasi bobot dan bias akhir untuk MLP NN. Model MLP NN dengan bobot dan bias hasil optimasi GWO kemudian digunakan sebagai model untuk klasifikasi.

6.3.4 Pengaturan Eksperimen

Eksperimen dilakukan untuk menguji kemampuan GWO dalam melatih MLP NN pada tiga dataset klasifikasi. Setiap jaringan diuji sebanyak 10 kali. Pengulangan ini dilakukan karena GWO bersifat stokastik, sehingga hasil satu kali eksekusi belum tentu mewakili performa umum algoritma. Jumlah iterasi maksimum ditetapkan sebesar 250 iterasi. Jumlah serigala yang digunakan dalam GWO adalah 12. Nilai posisi serigala dibatasi pada rentang -5 sampai 5. Rentang ini digunakan sebagai batas nilai bobot dan bias yang dapat dicari oleh GWO.

Evaluasi dilakukan menggunakan dua ukuran utama, yaitu MSE dan akurasi klasifikasi. MSE digunakan untuk mengukur kesalahan output jaringan terhadap target. Nilai MSE yang lebih kecil menunjukkan bahwa output jaringan semakin dekat dengan target. Selain itu, standar deviasi MSE digunakan untuk melihat kestabilan hasil pelatihan dalam beberapa kali pengujian. Akurasi klasifikasi

digunakan untuk melihat persentase data yang berhasil diklasifikasikan dengan benar.

6.3.5 Hasil Eksperimen

Hasil eksperimen menunjukkan bahwa GWO mampu melatih ANN dengan performa yang baik pada ketiga dataset. Ringkasan hasil eksperimen disajikan pada Tabel 6.2. Secara umum, nilai akurasi yang diperoleh menunjukkan bahwa bobot dan bias hasil pencarian GWO mampu menghasilkan model klasifikasi yang efektif.

Tabel 6.2. Akurasi klasifikasi MLP NN yang dilatih menggunakan GWO.

No.	Dataset	Arsitektur MLP NN	MSE (Avg)	MSE (SD)	Akurasi (%)
1	Iris	4-9-3	0.0449	0.0269	98.6667
2	Lenses	4-9-3	0.0677	0.0489	98.4328
3	Sonar	9-19-2	0.0794	0.0112	95.7692

Pada dataset Iris, GWO menghasilkan MSE sebesar 0,0449 ± 0,0269 dan akurasi klasifikasi sebesar 98,6667%. Hasil ini menunjukkan bahwa GWO mampu menemukan bobot dan bias yang sesuai untuk membedakan tiga kelas bunga Iris. Nilai error yang rendah juga menunjukkan bahwa output jaringan cukup dekat dengan target yang diharapkan.

Pada dataset Lenses, GWO menghasilkan MSE sebesar 0,0677 ± 0,0489 dan akurasi klasifikasi sebesar 98,4328%. Dataset ini memiliki jumlah sampel yang sangat kecil, sehingga proses pelatihan lebih rentan terhadap ketidakstabilan. Namun, hasil yang diperoleh menunjukkan bahwa GWO tetap mampu menghasilkan model dengan akurasi tinggi pada data terbatas.

Pada dataset Sonar, GWO menghasilkan MSE sebesar 0,0794 ± 0,0112 dan akurasi klasifikasi sebesar 95,76922%. Hasil ini penting karena dataset Sonar memiliki pola kelas yang sulit

dibedakan. Sinyal pantulan dari batu dan silinder logam memiliki kemiripan yang tinggi, sehingga model memerlukan proses pelatihan yang kuat. Setelah fitur direduksi menggunakan PCA, GWO mampu melatih ANN dengan akurasi yang tetap tinggi. Nilai standar deviasi pada dataset Sonar juga relatif kecil. Hal ini menunjukkan bahwa hasil pelatihan GWO cukup stabil pada beberapa kali pengujian. Stabilitas ini penting karena proses pelatihan berbasis metaheuristik memiliki unsur stokastik. Jika standar deviasi kecil, maka performa model tidak terlalu berubah ketika proses pelatihan diulang.

Daftar Pustaka

Abdulhameed, S., Rashid, T.A., 2022. Child drawing development optimization algorithm based on child's cognitive development. Arab. J. Sci. Eng. 47, 1337–1351.

Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M., Gandomi, A.H., 2021. The arithmetic optimization algorithm. Comput. Methods Appl. Mech. Eng. 376, 113609.

Ahmed, A.N., Van Lam, T., Hung, N.D., Van Thieu, N., Kisi, O., El-Shafie, A., 2021. A comprehensive comparison of recent developed meta-heuristic algorithms for streamflow time series forecasting problem. Appl. Soft Comput. 105, 107282. https://doi.org/https://doi.org/10.1016/j.asoc.2021.107282

Ahmed, Q.I., Attar, H., Amer, A., Deif, M.A., Solyman, A.A.A., 2023. Development of a hybrid support vector machine with grey wolf optimization algorithm for detection of the solar power plants anomalies. Systems 11, 237.

Asemi, H., Farajzadeh, N., 2025. Improving EEG signal-based emotion recognition using a hybrid GWO-XGBoost feature selection method. Biomed. Signal Process. Control 99, 106795. https://doi.org/https://doi.org/10.1016/j.bspc.2024.106795

Azizi, M., Aickelin, U., A. Khorshidi, H., Baghalzadeh Shishehgarkhaneh, M., 2023. Energy valley optimizer: a novel metaheuristic algorithm for global and engineering optimization. Sci. Rep. 13, 226.

Bergstra, J., Bengio, Y., 2012. Random search for hyper-parameter optimization. J. Mach. Learn. Res. 13, 281–305.

Berrar, D., Dubitzky, W., 2013. Overfitting BT - Encyclopedia of Systems Biology, in: Dubitzky, W., Wolkenhauer, O., Cho, K.-

H., Yokota, H. (Eds.), . Springer New York, New York, NY, pp. 1617–1619. https://doi.org/10.1007/978-1-4419-9863-7_601

Bishop, C.M., 2010. Supervised Learning, in: Sammut, C., Webb, G.I. (Eds.), Encyclopedia of Machine Learning. Springer, Boston, MA, pp. 928–931. https://doi.org/10.1007/978-0-387-30164-8_785

Bishop, C.M., 2006. Pattern recognition and machine learning. springer.

Blum, C., Roli, A., 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput. Surv. 35, 268–308.

Bouaouda, A., Sayouti, Y., 2022. Hybrid Meta-Heuristic Algorithms for Optimal Sizing of Hybrid Renewable Energy System: A Review of the State-of-the-Art. Arch. Comput. Methods Eng. 29, 4049–4083. https://doi.org/10.1007/s11831-022-09730-x

Chapelle, O., Schölkopf, B., Zien, A. (Eds.), 2006. Semi-Supervised Learning. MIT Press, Cambridge, MA.

Chen, J., Dai, Z., 2024. Metaheuristic algorithms for groundwater model parameter inversion: Advances and prospects. Deep Resour. Eng. 1, 100009. https://doi.org/https://doi.org/10.1016/j.deepre.2024.100009

Chen, T., Kornblith, S., Norouzi, M., Hinton, G., 2020. A Simple Framework for Contrastive Learning of Visual Representations, in: Proceedings of the 37th International Conference on Machine Learning, Proceedings of Machine Learning Research. PMLR, pp. 1597–1607.

Chong, J.Y., Hooi, G.L., Goh, Q.Y., Lai, V., Huang, Y.F., Koo, C.H., El-Shafie, A., Ahmed, A.N., 2024. Adapting reservoir operations for optimal water management under varying climate and demand scenarios using metaheuristic algorithms. Ain Shams Eng. J. 15, 102835. https://doi.org/https://doi.org/10.1016/j.asej.2024.102835

Chopra, N., Mohsin Ansari, M., 2022. Golden jackal optimization: A novel nature-inspired optimizer for engineering applications. Expert Syst. Appl. 198, 116924. https://doi.org/https://doi.org/10.1016/j.eswa.2022.116924

Cinar, A.C., Natarajan, N., 2022. An artificial neural network optimized by grey wolf optimizer for prediction of hourly wind speed in Tamil Nadu, India. Intell. Syst. with Appl. 16, 200138. https://doi.org/https://doi.org/10.1016/j.iswa.2022.200138

Devlin, J., Chang, M.-W., Lee, K., Toutanova, K., 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Association for Computational Linguistics, pp. 4171–4186. https://doi.org/10.18653/v1/N19-1423

Djebedjian, B., Abdel-Gawad, H.A.A., Ezzeldin, R.M., 2021. Global performance of metaheuristic optimization tools for water distribution networks. Ain Shams Eng. J. 12, 223–239. https://doi.org/https://doi.org/10.1016/j.asej.2020.07.012

Dorigo, M., Birattari, M., Stutzle, T., 2006. Ant colony optimization. IEEE Comput. Intell. Mag. 1, 28–39.

Elshaer, R., Awad, H., 2020. A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. Comput. Ind. Eng. 140, 106242.

Eskandar, H., Sadollah, A., Bahreininejad, A., Hamdi, M., 2012. Water cycle algorithm–A novel metaheuristic optimization method for solving constrained engineering optimization problems. Comput. Struct. 110, 151–166.

Faramarzi, A., Heidarinejad, M., Mirjalili, S., Gandomi, A.H., 2020. Marine Predators Algorithm: A nature-inspired metaheuristic. Expert Syst. Appl. 152, 113377.

Fogel, L.J., 1999. Intelligence through simulated evolution: Forty

years of evolutionary programming. John Wiley & Sons, Inc.

Geem, Z.W., Kim, J.H., Loganathan, G.V., 2001. A new heuristic optimization algorithm: harmony search. Simulation 76, 60–68.

Ghaemifard, S., Ghannadiasl, A., 2024. Usages of metaheuristic algorithms in investigating civil infrastructure optimization models; a review. AI Civ. Eng. 3, 17. https://doi.org/10.1007/s43503-024-00036-4

Giordani, P., 2018. Principal Component Analysis BT - Encyclopedia of Social Network Analysis and Mining, in: Alhajj, R., Rokne, J. (Eds.), . Springer New York, New York, NY, pp. 1831–1844. https://doi.org/10.1007/978-1-4939-7131-2_154

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press. https://doi.org/10.1109/ACCESS.2020.2988550

Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative Adversarial Nets, in: Advances in Neural Information Processing Systems 27 (NeurIPS 2014). pp. 2672–2680.

Gui, J., Chen, T., Zhang, J., Cao, Q., Sun, Z., Luo, H., Tao, D., 2024. A Survey on Self-supervised Learning: Algorithms, Applications, and Future Trends.

Gupta, S., Deep, K., 2019. A novel Random Walk Grey Wolf Optimizer. Swarm Evol. Comput. 44, 101–112. https://doi.org/https://doi.org/10.1016/j.swevo.2018.01.001

Gusain, C., Mohan Tripathi, M., Nangia, U., 2023. Study of Meta-heuristic Optimization Methodologies for Design of Hybrid Renewable Energy Systems. Therm. Sci. Eng. Prog. 39, 101711. https://doi.org/https://doi.org/10.1016/j.tsep.2023.101711

Hashim, F.A., Houssein, E.H., Hussain, K., Mabrouk, M.S., Al-Atabany, W., 2022. Honey Badger Algorithm: New

metaheuristic algorithm for solving optimization problems. Math. Comput. Simul. 192, 84–110.

Hashim, F.A., Mostafa, R.R., Hussien, A.G., Mirjalili, S., Sallam, K.M., 2023. Fick's Law Algorithm: A physical law-based algorithm for numerical optimization. Knowledge-Based Syst. 260, 110146.

Hastie, Trevor, Tibshirani, R., Friedman, J., 2009. The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd ed. Springer, New York.

Hastie, Tibshirani, Tibshirani, R., Friedman, J., 2009. The elements of statistical learning New York, 2nd ed. Springer New York, New York, N.Y.

Heidari, A.A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., Chen, H., 2019. Harris hawks optimization: Algorithm and applications. Futur. Gener. Comput. Syst. 97, 849–872.

Hemeida, A.M., Alkhalaf, S., Mady, A., Mahmoud, E.A., Hussein, M.E., Baha Eldin, A.M., 2020. Implementation of nature-inspired optimization algorithms in some data mining tasks. Ain Shams Eng. J. 11, 309–318. https://doi.org/https://doi.org/10.1016/j.asej.2019.10.003

Holland, J.H., 1992. Genetic algorithms. Sci. Am. 267, 66–73.

Houssein, E.H., Gad, A.G., Wazery, Y.M., Suganthan, P.N., 2021. Task Scheduling in Cloud Computing based on Meta-heuristics: Review, Taxonomy, Open Challenges, and Future Trends. Swarm Evol. Comput. 62, 100841. https://doi.org/https://doi.org/10.1016/j.swevo.2021.100841

Huang, J., Deng, X., Hu, L., 2025. Enhanced grey wolf optimizer with hybrid strategies for efficient feature selection in high-dimensional data. Inf. Sci. (Ny). 705, 121958. https://doi.org/https://doi.org/10.1016/j.ins.2025.121958

Jain, S., Jain, A., Jangid, M., 2025. MGWO-CNN: hyperparameter

optimization of CNN classifier for cervical cancer detection using Modified Grey Wolf Optimizer. Sci. Rep.

Kalita, K., Jangir, P., Čepová, L., Chakraborty, S., Pandya, S.B., Arpita, 2025. Many-Objective Grey Wolf Optimizer (MaOGWO) for solving real-world problems. Results Eng. 28, 106941. https://doi.org/https://doi.org/10.1016/j.rineng.2025.106941

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization, in: Proceedings of ICNN'95-International Conference on Neural Networks. IEEE, pp. 1942–1948.

Kingma, D.P., Welling, M., 2014. Auto-Encoding Variational Bayes, in: International Conference on Learning Representations (ICLR).

Kuhn, M., Johnson, K., 2013. Applied Predictive Modeling. Springer, New York.

Kumar, V., Chhabra, J.K., Kumar, D., 2017. Grey wolf algorithm-based clustering technique. J. Intell. Syst. 26, 153–168.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., Bengio, Y., 2007. An empirical evaluation of deep architectures on problems with many factors of variation, in: Proceedings of the 24th International Conference on Machine Learning. pp. 473–480.

Li, M.D., Zhao, H., Weng, X.W., Han, T., 2016. A novel nature-inspired algorithm for optimization: Virus colony search. Adv. Eng. Softw. 92, 65–88.

Liao, T.W., Egbelu, P.J., Sarker, B.R., Leu, S.S., 2011. Metaheuristics for project and construction management – A state-of-the-art review. Autom. Constr. 20, 491–505. https://doi.org/https://doi.org/10.1016/j.autcon.2010.12.006

Liu, G., Guo, Z., Liu, W., Jiang, F., Fu, E., 2024. A feature selection method based on the Golden Jackal-Grey Wolf Hybrid

Optimization Algorithm. PLoS One 19, e0295579.

Ma, J., Xia, D., Guo, H., Wang, Y., Niu, X., Liu, Z., Jiang, S., 2022. Metaheuristic-based support vector regression for landslide displacement prediction: a comparative study. Landslides 19, 2489–2511.

Makhadmeh, S.N., Al-Betar, M.A., Doush, I.A., Awadallah, M.A., Kassaymeh, S., Mirjalili, S., Zitar, R.A., 2023. Recent advances in Grey Wolf Optimizer, its versions and applications. Ieee Access 12, 22991–23028.

Manjunath, B.S., Salembier, P., Sikora, T., 2002. Introduction to MPEG-7: multimedia content description interface. John Wiley & Sons.

Mao, A., Mohri, M., Zhong, Y., 2023. Cross-entropy loss functions: Theoretical analysis and applications, in: International Conference on Machine Learning. pmlr, pp. 23803–23828.

Mech, L.D., 1999. Alpha status, dominance, and division of labor in wolf packs. Can. J. Zool. 77, 1196–1203.

Mirjalili, S., 2016. SCA: a sine cosine algorithm for solving optimization problems. Knowledge-based Syst. 96, 120–133.

Mirjalili, S., 2015a. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowledge-based Syst. 89, 228–249.

Mirjalili, S., 2015b. How effective is the Grey Wolf optimizer in training multi-layer perceptrons. Appl. Intell. 43, 150–161. https://doi.org/10.1007/s10489-014-0645-7

Mirjalili, S., Lewis, A., 2016. The whale optimization algorithm. Adv. Eng. Softw. 95, 51–67.

Mirjalili, S., Lewis, A., 2013. S-shaped versus V-shaped transfer functions for binary Particle Swarm Optimization. Swarm Evol. Comput. 9, 1–14. https://doi.org/https://doi.org/10.1016/j.swevo.2012.09.002

Mirjalili, S., Mirjalili, S.M., Lewis, A., 2014. Grey wolf optimizer. Adv. Eng. Softw. 69, 46–61.

Mirjalili, S.Z., Mirjalili, S., Saremi, S., Faris, H., Aljarah, I., 2018. Grasshopper optimization algorithm for multi-objective optimization problems. Appl. Intell. 48, 805–820. https://doi.org/10.1007/s10489-017-1019-8

Mishra, M., Gunturi, V.R., Miranda, T.F.D.S., 2019. Slope stability analysis using recent metaheuristic techniques: A comprehensive survey. SN Appl. Sci. 1, 1674.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D., 2015. Human-level control through deep reinforcement learning. Nature 518, 529–533. https://doi.org/10.1038/nature14236

Mohakud, R., Dash, R., 2022a. Skin cancer image segmentation utilizing a novel EN-GWO based hyper-parameter optimized FCEDN. J. King Saud Univ. - Comput. Inf. Sci. 34, 9889–9904. https://doi.org/https://doi.org/10.1016/j.jksuci.2021.12.018

Mohakud, R., Dash, R., 2022b. Designing a grey wolf optimization based hyper-parameter optimized convolutional neural network classifier for skin cancer detection. J. King Saud Univ. Inf. Sci. 34, 6280–6291.

Mosavi, M.R., Khishe, M., Ghamgosar, A., 2016. Classification of sonar data set using neural network trained by gray wolf optimization. Neural Netw. World 26, 393.

Moscato, P., 1989. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Caltech Concurr. Comput. program, C3P Rep. 826, 37.

Muro, C., Escobedo, R., Spector, L., Coppinger, R.P., 2011. Wolf-pack (Canis lupus) hunting strategies emerge from simple rules

in computational simulations. Behav. Processes 88, 192–197.

Murphy, K.P., 2012. Machine learning: a probabilistic perspective. MIT press.

Nassef, A.M., Abdelkareem, M.A., Maghrabie, H.M., Baroutaji, A., 2023. Review of Metaheuristic Optimization Algorithms for Power Systems Problems. Sustainability. https://doi.org/10.3390/su15129434

Nielsen, P., Dahanayaka, M., Perera, H.N., Thibbotuwawa, A., Kilic, D.K., 2024. A systematic review of vehicle routing problems and models in multi-echelon distribution networks. Supply Chain Anal. 7, 100072.

Niu, B., Wang, H., 2012. Bacterial colony optimization. Discret. Dyn. Nat. Soc. 2012, 698057.

Obadina, Ololade O, Thaha, Mohamed A, Althoefer, Kaspar, Shaheed, Mohammad H, 2022. Dynamic characterization of a master–slave robotic manipulator using a hybrid grey wolf–whale optimization algorithm. J. Vib. Control 28, 1992–2003. https://doi.org/10.1177/10775463211003402

Ong, H.C., Milano, J., Silitonga, A.S., Hassan, M.H., Shamsuddin, A.H., Wang, C.-T., Indra Mahlia, T.M., Siswantoro, J., Kusumo, F., Sutrisno, J., 2019. Biodiesel production from Calophyllum inophyllum-Ceiba pentandra oil mixture: Optimization and characterization. J. Clean. Prod. 219, 183–198. https://doi.org/10.1016/j.jclepro.2019.02.048

Paleyes, A., Urma, R.-G., Lawrence, N.D., 2022. Challenges in Deploying Machine Learning: A Survey of Case Studies. ACM Comput. Surv. 55. https://doi.org/10.1145/3533378

Purushothaman, R., Rajagopalan, S.P., Dhandapani, G., 2020. Hybridizing Gray Wolf Optimization (GWO) with Grasshopper Optimization Algorithm (GOA) for text feature selection and clustering. Appl. Soft Comput. 96, 106651. https://doi.org/https://doi.org/10.1016/j.asoc.2020.106651

Rabhi, B., Dhahri, H., Alimi, A.M., Alturki, F.A., 2017. Grey Wolf Optimizer for Training Elman Neural Network BT - Proceedings of the 16th International Conference on Hybrid Intelligent Systems (HIS 2016), in: Abraham, A., Haqiq, A., Alimi, A.M., Mezzour, G., Rokbani, N., Muda, A.K. (Eds.), . Springer International Publishing, Cham, pp. 380–390.

Rao, R.V., Patel, V., 2012. An elitist teaching-learning-based optimization algorithm for solving complex constrained optimization problems. Int. J. Ind. Eng. Comput. 3, 535–560.

Rechenberg, I., 1984. The evolution strategy. a mathematical model of darwinian evolution, in: Synergetics—From Microscopic to Macroscopic Order: Proceedings of the International Symposium on Synergetics at Berlin, July 4–8, 1983. Springer, pp. 122–132.

Rezk, H., Olabi, A.G., Mahmoud, M., Wilberforce, T., Sayed, E.T., 2024. Metaheuristics and multi-criteria decision-making for renewable energy systems: Review, progress, bibliometric analysis, and contribution to the sustainable development pillars. Ain Shams Eng. J. 15, 102883. https://doi.org/https://doi.org/10.1016/j.asej.2024.102883

Robnik-Šikonja, M., Kononenko, I., 2003. Theoretical and Empirical Analysis of ReliefF and RReliefF. Mach. Learn. 53, 23–69. https://doi.org/10.1023/A:1025667309714

Rocha, A., Hauagge, D.C., Wainer, J., Goldenstein, S., 2010. Automatic fruit and vegetable classification from images. Comput. Electron. Agric. 70, 96–104. https://doi.org/http://dx.doi.org/10.1016/j.compag.2009.09.002

Saeys, Y., Inza, I., Larrañaga, P., 2007. A review of feature selection techniques in bioinformatics. Bioinformatics 23, 2507–2517. https://doi.org/10.1093/bioinformatics/btm344

Samuel, A.L., 1959. Some studies in machine learning using the game of checkers. IBM J. Res. Dev. 3, 210–229.

Santra, S., Hsieh, J.-W., Lin, C.-F., 2021. Gradient descent effects on differential neural architecture search: A survey. IEEE Access 9, 89602–89618.

Schlegel, M., Sattler, K.-U., 2023. Management of machine learning lifecycle artifacts: A survey. ACM SIGMOD Rec. 51, 18–35.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal Policy Optimization Algorithms. arXiv Prepr. arXiv1707.06347.

Sebayang, A.H., Kusumo, F., Milano, J., Shamsuddin, A.H., Silitonga, A.S., Ideris, F., Siswantoro, J., Veza, I., Mofijur, M., Reen Chia, S., 2023. Optimization of biodiesel production from rice bran oil by ultrasound and infrared radiation using ANN-GWO. Fuel 346, 128404. https://doi.org/https://doi.org/10.1016/j.fuel.2023.128404

Shahin, I., Alomari, O.A., Nassif, A.B., Afyouni, I., Hashem, I.A., Elnagar, A., 2023. An efficient feature selection method for arabic and english speech emotion recognition using Grey Wolf Optimizer. Appl. Acoust. 205, 109279. https://doi.org/https://doi.org/10.1016/j.apacoust.2023.109279

Shariati, M., Mafipour, M.S., Ghahremani, B., Azarhomayun, F., Ahmadi, M., Trung, N.T., Shariati, A., 2022. A novel hybrid extreme learning machine–grey wolf optimizer (ELM-GWO) model to predict compressive strength of concrete with partial replacements for cement. Eng. Comput. 38, 757–779. https://doi.org/10.1007/s00366-020-01081-0

Silitonga, A.S., Shamsuddin, A.H., Mahlia, T.M.I., Milano, J., Kusumo, F., Siswantoro, J., Dharma, S., Sebayang, A.H., Masjuki, H.H., Ong, H.C., 2020. Biodiesel synthesis from Ceiba pentandra oil by microwave irradiation-assisted transesterification: ELM modeling and optimization. Renew. Energy 146, 1278–1291. https://doi.org/10.1016/j.renene.2019.07.065

Siswantoro, J., 2024. Enhanced Support Vector Machine Based on Grey Wolf Optimizer for Fruits Image Classification using MPEG-7 Color and Texture Features Fusion. Int. J. Intell. Eng. Syst. 17, 1–11.

Siswantoro, J., Arwoko, H., Siswantoro, M.Z.F.N., 2020a. Fruits Classification from Image using MPEG-7 Visual Descriptors and Extreme Learning Machine, in: 2020 3rd International Seminar on Research of Information Technology and Intelligent Systems, ISRITI 2020. https://doi.org/10.1109/ISRITI51436.2020.9315523

Siswantoro, J., Arwoko, H., Widiasri, M., 2020b. Indonesian fruits classification from image using MPEG-7 descriptors and ensemble of simple classifiers. J. Food Process Eng. 43, 1–13. https://doi.org/10.1111/jfpe.13414

Siswantoro, J., Nehemiah Wijaya, J., Riandaru Prasetyo, V., Arwoko, H., 2026. A Hybrid Binary Grey Wolf and Whale Optimization Method for Feature Selection in Classification Tasks. Expert Syst. Appl. 132277. https://doi.org/https://doi.org/10.1016/j.eswa.2026.132277

Siswantoro, M.Z.F.N., Rochimah, S., Sunaryono, D., 2024. Software Defect Detection Using Optimized Support Vector Machine Based on Grey Wolf Optimizer with Random Walk, in: 2024 4th International Conference of Science and Information Technology in Smart Administration (ICSINTESA). IEEE, pp. 390–395.

Siswantoro, M.Z.F.N., Yuhana, U.L., 2023. Software Defect Prediction Based on Optimized Machine Learning Models: A Comparative Study. Teknika 12, 166–172.

Snoek, J., Larochelle, H., Adams, R.P., 2012. Practical bayesian optimization of machine learning algorithms. Adv. Neural Inf. Process. Syst. 25.

Sokolova, M., Lapalme, G., 2009. A Systematic Analysis of

Performance Measures for Classification Tasks. Inf. Process. \& Manag. 45, 427–437. https://doi.org/10.1016/j.ipm.2009.03.002

Storn, R., 1996. On the usage of differential evolution for function optimization, in: Proceedings of North American Fuzzy Information Processing. Ieee, pp. 519–523.

Sunaryono, D., Sarno, R., Sabilla, S.I., Putri, R.A., Amri, T.C., Mirda, I., Siswantoro, J., 2025. Augmented Data Low Confidence (ADLC): A Confidence-Driven Data Augmentation Framework With Ensemble Optimization for Enhanced Machine Learning Performance. IEEE Access 13, 201439–201459. https://doi.org/10.1109/ACCESS.2025.3636729

Sunaryono, D., Sarno, R., Siswantoro, J., Budi Raharjo, A., Izza Sabilla, S., Indarto Susilo, R., Rekha, K., 2022. ENHANCED SALP SWARM ALGORITHM BASED ON CONVOLUTIONAL NEURAL NETWORK OPTIMIZATION FOR AUTOMATIC EPILEPSY DETECTION. J. Theor. Appl. Inf. Technol. 15, 19.

Sunaryono, D., Siswantoro, J., Raharjo, A.B., Ridho, R., Sarno, R., Sabilla, S.I., Susilo, R.I., 2023. Optimized One-Dimension Convolutional Neural Network for Seizure Classification from EEG Signal based on Whale Optimization Algorithm. Int. J. Intell. Eng. Syst. 16, 310–322. https://doi.org/10.22266/ijies2023.0630.25

Sutton, R.S., Barto, A.G., 2018. Reinforcement Learning: An Introduction, 2nd ed. MIT Press, Cambridge, MA.

Tikhamarine, Y., Souag-Gamane, D., Najah Ahmed, A., Kisi, O., El-Shafie, A., 2020. Improving artificial intelligence models accuracy for monthly streamflow forecasting using grey Wolf optimization (GWO) algorithm. J. Hydrol. 582, 124435. https://doi.org/https://doi.org/10.1016/j.jhydrol.2019.124435

Tom M. Mitchell, 1997. Machine Learning. McGraw-Hill, Redmond, WA.

Van Laarhoven, P.J.M., Aarts, E.H.L., 1987. Simulated annealing, in: Simulated Annealing: Theory and Applications. Springer, pp. 7–15.

Van Thieu, N., Mirjalili, S., 2023. MEALPY: An open-source library for latest meta-heuristic algorithms in Python. J. Syst. Archit. 139, 102871. https://doi.org/https://doi.org/10.1016/j.sysarc.2023.102871

Wang, T., Meng, H., Zhang, F., Qin, R., 2024. Fault Detection of Wheelset Bearings through Vibration-Sound Fusion Data Based on Grey Wolf Optimizer and Support Vector Machine. Technologies 12, 144.

Watkins, C.J.C.H., Dayan, P., 1992. Q-learning. Mach. Learn. 8, 279–292. https://doi.org/10.1007/BF00992698

Wijaya, J.N., Siswantoro, J., 2025. Systematic Literature Review of Hybrid Metaheuristic Algorithms for Feature Selection in Classification Tasks, in: 2025 International Conference on Information Management and Technology (ICIMTech). IEEE, pp. 103–108.

Wirth, R., Hipp, J., 2000. CRISP-DM: Towards a Standard Process Model for Data Mining, in: Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining. pp. 29–39.

Xue, B., Zhang, M., Browne, W.N., 2014. Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms. Appl. Soft Comput. 18, 261–276. https://doi.org/https://doi.org/10.1016/j.asoc.2013.09.018

Yang, X.-S., Karamanoglu, M., He, X., 2014. Flower pollination algorithm: a novel approach for multiobjective optimization. Eng. Optim. 46, 1222–1237.

Yang, Y., Chen, H., Heidari, A.A., Gandomi, A.H., 2021. Hunger games search: Visions, conception, implementation, deep

analysis, perspectives, and towards performance shifts. Expert Syst. Appl. 177, 114864.

Yu, M., Xu, J., Liang, W., Qiu, Y., Bao, S., Tang, L., 2024. Improved multi-strategy adaptive Grey Wolf Optimization for practical engineering applications and high-dimensional problem solving. Artif. Intell. Rev. 57, 277. https://doi.org/10.1007/s10462-024-10821-3

Zhang, J., Dai, Y., Shi, Q., 2024. An improved grey wolf optimization algorithm based on scale-free network topology. Heliyon 10, e35958. https://doi.org/https://doi.org/10.1016/j.heliyon.2024.e35958

Zhang, Y., Wang, S., Ji, G., Phillips, P., 2014. Fruit classification using computer vision and feedforward neural network. J. Food Eng. 143, 167–177.

Zhao, W., Wang, L., Zhang, Z., 2020a. Artificial ecosystem-based optimization: a novel nature-inspired meta-heuristic algorithm. Neural Comput. Appl. 32, 9383–9425.

Zhao, W., Zhang, Z., Wang, L., 2020b. Manta ray foraging optimization: An effective bio-inspired optimizer for engineering applications. Eng. Appl. Artif. Intell. 87, 103300.

Zhu, X., 2010. Semi-Supervised Learning BT - Encyclopedia of Machine Learning, in: Sammut, C., Webb, G.I. (Eds.), . Springer US, Boston, MA, pp. 892–897. https://doi.org/10.1007/978-0-387-30164-8_749

Biografi Penulis

Prof. Dr. Joko Siswantoro, S.Si., M.Si., HCIA-AI, CADS adalah seorang akademisi, peneliti, dan pakar di bidang Kecerdasan Artificial, khususnya Pembelajaran Mesin (*Machine Learning*). Lahir di Lamongan pada 9 Desember 1974, beliau saat ini mengemban amanah sebagai Profesor dalam bidang Pembelajaran Mesin di Universitas Surabaya (UBAYA), institusi tempat beliau mengabdi sejak tahun 1997. Selain mengajar, beliau juga pernah menjabat sebagai Kepala Laboratorium *Intelligent System* di Program Studi Teknik Informatika UBAYA (2017-2021) dan Ketua Program Studi Teknik Informatika UBAYA (2020-2027). Latar belakang akademik beliau merupakan perpaduan kuat antara ilmu dasar dan komputasi terapan. Beliau meraih gelar Sarjana Sains (S.Si.) bidang Matematika dari Universitas Airlangga pada tahun 1997, dilanjutkan dengan gelar Magister Sains (M.Si.) bidang Matematika dari Institut Teknologi Bandung (ITB) pada tahun 2002.

Guna memperdalam kepakarannya di dunia teknologi, beliau menyelesaikan pendidikan *Doctor of Philosophy (Ph.D.)* dalam bidang *Computer Science* di *Center for Artificial Intelligence Technology*, Universiti Kebangsaan Malaysia (UKM) pada tahun 2016. Sebagai seorang profesional, Prof. Joko Siswantoro juga memegang berbagai sertifikasi keahlian ternama, antara lain *Certified Associate Data Scientist* (CADS), *NVIDIA Instructor for Fundamentals of Deep Learning*, dan *Huawei Certified ICT Associate – Artificial Intelligence* (HCIA-AI). Dedikasinya di dunia riset berfokus pada ranah *Machine Learning*, *Deep Learning*, *Computer Vision*, dan *Optimization*. Portofolio akademisnya sangat impresif dengan raihan *H-Index* Scopus 14 pada tahun 2026, didukung oleh

puluhan publikasi ilmiah berskala internasional di berbagai jurnal dan prosiding bereputasi tinggi. Di samping aktif mempublikasikan artikel ilmiah, beliau juga sangat produktif dalam menulis buku teks dan monograf ilmiah. Beberapa karya buku yang telah beliau terbitkan antara lain:

1. Sistem Penglihatan Komputer: Pengukuran Isipadu Produk Secara Kaedah Monte Carlo (Penerbit Universiti Kebangsaan Malaysia, 2022)
2. Matematika Farmasi (Edisi Revisi) (MNC Publishing, 2018)
3. Kalkulus Edisi Revisi (Media Nusa Creative, 2017)
4. Matematika Farmasi (Bayumedia Publishing, 2012)
5. Metode Numerik dengan Scilab (Bayumedia Publishing, 2011)
6. Kalkulus (Bayumedia Publishing, 2006)

Atas komitmennya dalam memajukan ilmu pengetahuan dan institusi, beliau kerap dipercaya memimpin berbagai hibah tingkat nasional, termasuk menjadi Koordinator Program Kompetisi Kampus Merdeka (PK-KM) Teknik Informatika UBAYA tahun 2022-2023, serta memenangkan serangkaian hibah riset bergengsi dari kementerian.

Aplikasi Grey Wolf Optimizer dalam Pembelajaran Mesin Modern

Buku ini membahas integrasi pembelajaran mesin (machine learning) dengan algoritma optimasi metaheuristik Grey Wolf Optimizer (GWO) sebagai salah satu pendekatan yang efektif dalam menyelesaikan berbagai permasalahan optimasi yang kompleks. Pembahasan diawali dengan pengenalan konsep-konsep dasar pembelajaran mesin, meliputi jenis-jenis pembelajaran, tahapan pengembangan model, perbedaan antara parameter dan hyperparameter, serta fungsi loss yang menjadi dasar evaluasi dan peningkatan kinerja model. Prinsip kerja GWO terinspirasi dari hierarki sosial dan strategi berburu kawanan serigala abu-abu. Penjelasan mencakup landasan matematis algoritma, mekanisme eksplorasi dan eksploitasi, serta berbagai pengembangan GWO yang dirancang untuk meningkatkan kemampuan pencarian solusi optimal pada berbagai permasalahan pembelajaran mesin.

Penerbit (Anggota IKAPI & APPTI)
Direktorat Penerbitan & Publikasi Ilmiah
Universitas Surabaya
Jl. Raya Kalirungkut, Surabaya 60293
Telp. (62-31) 2981344
E-mail: ppi@unit.ubaya.ac.id
Web: ppi.ubaya.ac.id

ISBN 978-623-8038-80-0
786238 038800