HIBRIDISASI *FIREWORKS ALGORITHM* (FWA) DAN *GREY WOLF OPTIMIZER* (GWO)

SKRIPSI

oleh

CLAUDIA INGGRIT SIREGAR
155090401111029

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG
2019

HIBRIDISASI *FIREWORKS ALGORITHM* (FWA) DAN *GREY WOLF OPTIMIZER* (GWO)

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana Matematika

oleh

CLAUDIA INGGRIT SIREGAR
155090401111029

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG
2019

LEMBAR PENGESAHAN SKRIPSI

HIBRIDISASI *FIREWORKS ALGORITHM* (FWA) DAN *GREY WOLF OPTIMIZER* (GWO)

oleh :

Claudia Inggrit Siregar

155090401111029

Setelah dipertahankan di depan Majelis Penguji
pada tanggal 11 Desember 2019
dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana Matematika

Pembimbing

Zuraidah Fitriah, S.Si., M.Si.
NIP. 198706102014042002

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Ratno Bagus Edy Wibowo, S.Si., M.Si., Ph.D.
NIP. 197509082000031003

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	**: Claudia Inggrit Siregar**
NIM	**: 155090401111029**
Jurusan	**: Matematika**
Penulisan Skripsi Berjudul	**: Hibridisasi *Fireworks Algorithm* (FWA) dan *Grey Wolf Optimizer* (GWO)**

Dengan ini menyatakan :

1. **Isi dari Skripsi yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam Skripsi ini.**
2. **Apabila di kemudian hari ternyata Skripsi yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 11 Desember 2019
Yang menyatakan,

Claudia Inggrit Siregar
NIM. 155090401111029

HIBRIDISASI *FIREWORKS ALGORITHM* (FWA) DAN *GREY WOLF OPTIMIZER* (GWO)

ABSTRAK

Fireworks Algorithm (FWA) merupakan algoritma metaheuristik yang meniru ledakan kembang api. FWA memiliki kekurangan pada salah satu operatornya dan strategi seleksi yang perlu menghitung jarak antar individu sehingga membuat waktu komputasi yang semakin lama. *Grey Wolf Optimizer* (GWO) adalah algoritma metaheuristik yang terinspirasi dari hierarki kepemimpinan dan perilaku berburu serigala abu-abu. GWO unggul dalam menyelesaikan fungsi unimodal dan multimodal dengan waktu komputasi yang dibutuhkan lebih singkat. Skripsi ini membahas hibridisasi baru untuk menggabungkan dua algoritma metaheuristik, yaitu Hibridisasi *Fireworks Algorithm* dan *Grey Wolf Optimizer* (Hibrid FWA-GWO). Hibrid FWA-GWO bertujuan untuk memperbaiki kekurangan yang dimiliki oleh FWA dan mempertahankan keunggulan dari kedua algoritma. Oleh karena itu, hibrid FWA-GWO menggunakan inisialisasi dan ledakan amplitudo yang digunakan pada FWA, untuk GWO akan digunakan hierarki kepemimpinan dari serigala dan penentuan lokasi selanjutnya agar mendapatkan nilai optimum. Hibrid FWA-GWO diterapkan pada *software* MATLAB R2014b dengan menggunakan enam fungsi tes dan dianalisis berdasarkan rata-rata nilai minimum, standar deviasi dari nilai minimum, nilai terbaik, dan rata-rata waktu komputasi yang dibandingkan dengan FWA dan GWO. Hasil yang diperoleh dari 30 kali percobaan menunjukkan bahwa hibrid FWA-GWO dapat menemukan nilai yang lebih baik dari segi rata-rata nilai minimum, nilai terbaik, dan standar deviasi dari nilai minimum dibandingkan FWA dan GWO dalam menyelesaikan fungsi unimodal dan fungsi multimodal berdimensi tinggi. Namun, waktu komputasi yang dibutuhkan lebih lama dibandingkan GWO.

Kata Kunci: hibridisasi *Fireworks Algorithm* dan *Grey Wolf Optimizer*, hibridisasi, optimasi.

HYBRIDIZATION OF FIREWORKS ALGORITHM (FWA) AND GREY WOLF OPTIMIZER (GWO)

ABSTRACT

Fireworks Algorithm (FWA) is a metaheuristic algorithm that imitates the explosion algorithm. FWA has a weakness in one of the operators and the selection strategy, the distances between individuals need to be calculated. Hence, the computational time is much higher. Grey Wolf Optimizer (GWO) is a metaheuristic algorithm that mimics the leadership hierarchy and hunting mechanism of grey wolf. GWO excels in completing unimodal and multimodal functions with the shorter computational time required. The main aim of this paper is to present a new hybridization for combining two metaheuristics algorithm, namely the Hybridization of Fireworks Algorithm and Grey Wolf Optimizer (Hybrid FWA-GWO). Hybrid FWA-GWO aims to enhance the weakness of FWA and maintain the excellences of both algorithms. Therefore, hybrid FWA-GWO uses initialization explosion amplitude of FWA and for the GWO, it uses the leadership hierarchy and the next location of population. The main features taken from FWA and GWO will be combined into a hybrid FWA-GWO to get the optimal value. Hybrid FWA-GWO is applied to MATLAB R2014b using six test functions and the results will be compared with FWA and GWO. The results to be compared are average minimum value, standard deviation of the minimum value, the best value, and average computational time. The results obtained from 30 trials show that the hybrid FWA-GWO can find the best value compared to FWA and GWO in completing unimodal functions and high dimensional multimodal functions despite the longer computational time required compared to GWO.

Keywords: Hybridization of Fireworks Algorithm and Grey Wolf Optimizer, hybridization, optimization.

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Allah SWT yang telah memberikan rahmat dan petunjuk-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul **Hibridisasi *Fireworks Algorithm* (FWA) dan *Grey Wolf Optimizer* (GWO).**

Adapun maksud dan tujuan dari penyusunan skripsi ini adalah untuk memenuhi persyaratan untuk memperoleh gelar Sarjana Matematika di Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Brawijaya.

Pada saat penyusunan skripsi ini, penulis mendapat banyak pengalaman serta bantuan dari pihak-pihak yang terlibat secara langsung maupun tidak langsung. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan rasa terimakasih kepada:

1. Zuraidah Fitriah, S.Si., M.Si. selaku dosen pembimbing atas segala bimbingan, motivasi, saran, dan kesabaran yang telah diberikan selama proses pengerjaan Skripsi ini.
2. Syaiful Anam, S.Si., MT., Ph.D. selaku dosen penguji atas segala kritik dan saran yang diberikan untuk perbaikan Skripsi ini.
3. Dr. Isnani Darti, S.Si., M.Si. selaku dosen penguji sekaligus dosen penasihat akademik atas segala bimbingan, nasihat, kritik, dan saran, selama perkuliahan yang sangat bermanfaat bagi penulis.
4. Seluruh dosen Matematika yang telah memberikan ilmu pengetahuan serta staf administrasi Jurusan Matematika atas segala bantuannya.
5. Hasudungan Siregar (Papa), Mulyati (Mama), dan Albert Anugerah Putra Siregar (Abang) atas segala doa, bantuan, dan motivasi yang tak pernah habis diberikan.
6. Keluarga besar Matematika UB 2015 atas kebersamaan, perjuangan, dukungan, kerjasama, dan semangat selama ini.

Semoga Allah SWT memberikan anugerah dan berkah-Nya kepada semua pihak yang telah membantu menyelesaikan Skripsi ini.

Penulis menyadari bahwa dalam penulisan Skripsi ini masih terdapat kekurangan. Oleh karena itu, kritik yang membangun sangat

penulis harapkan sebagai masukan untuk penulisan selanjutnya, melalui email cldainggrits@gmail.com.

Harapan penulis semoga Skripsi ini dapat bermanfaat bagi para pembaca dalam menambah wawasan keilmuan atau bahan informasi lainnya.

Malang, 11 Desember 2019

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Algoritma adalah setiap prosedur komputasi yang terdefinisi dengan baik yang mengambil beberapa nilai, atau seperangkat nilai-nilai, sebagai masukan dan menghasilkan beberapa nilai, atau seperangkat nilai-nilai, sebagai output. Sebuah algoritma merupakan langkah komputasi yang mengubah input ke output. Algoritma juga sebagai alat bantu untuk memecahkan masalah (Cormen, dkk, 2001). Untuk menyelesaikan berbagai permasalahan, berbagai algoritma telah dikelompokkan ke dalam kelompok algoritma optimasi. Optimasi adalah salah satu disiplin ilmu dalam matematika yang fokus untuk mendapatkan nilai minimum atau maksimum secara sistematis dari suatu fungsi maupun pencarian nilai lainnya dalam berbagai kasus (Sugioko, 2013).

Secara umum, algoritma optimasi dibagi menjadi dua, yaitu algoritma deterministik dan algoritma stokastik. Kebanyakan algoritma konvensional atau klasik bersifat deterministik. Beberapa algoritma optimasi deterministik memiliki kelebihan dalam menyelesaikan masalah fungsi unimodal, akan tetapi kelemahan dari algoritma metode ini membutuhkan informasi gradien sehingga hanya dapat diterapkan pada fungsi yang memiliki turunan dan dalam proses pencarian dapat terjebak dalam optimal lokal. Oleh karena itu, dibutuhkan algoritma yang mampu menyelesaikan kelemahan tersebut yaitu algoritma stokastik. Untuk algoritma stokastik, secara umum terdapat dua jenis yaitu, heuristik dan metaheuristik (Yang, dkk, 2010).

Heuristik merupakan strategi algoritma pencarian solusi optimal dengan melakukan *trial-and-error* untuk pemecahan masalah yang cukup kompleks. Akan tetapi, permasalahan yang kompleks itu sendiri akan membuat waktu komputasi cukup lama dan sulit dalam menggunakan algoritma heuristik ke komputer (Yang, dkk, 2010).

Karena keterbatasan pada algoritma heuristik, muncul suatu pendekatan yang berbeda yang dinamakan algoritma metaheuristik. Metaheuristik dapat didefinisikan sebagai metode lanjut (*advanced*) berbasis heuristik untuk menyelesaikan persoalan optimasi secara efisien (Talbi, 2009). Metaheuristik sendiri merupakan sebuah

algoritma yang sebagian besar terinspirasi oleh kejadian yang ada di alam atau lebih dikenal dengan *nature inspired algorithm* (Yang, dkk, 2012).

Algoritma metaheuristik memiliki kecepatan pencarian solusi optimal yang lebih baik daripada metode tradisional. Metode ini juga memberikan hasil yang lebih baik dibandingkan heuristik karena metode ini akan selalu berusaha untuk keluar dari solusi optimal lokal (Madi, dkk, 2013). Meskipun tidak ada jaminan bahwa solusi yang ditemukan merupakan solusi yang optimal, metode metaheuristik yang dibangun dengan baik dapat memberikan solusi yang mendekati solusi optimal (Hillier dan Lieberman, 2010). Beberapa contoh algoritma metaheuristik yaitu, *Genetic Algorithm* (GA), *Particle Swarm Optimization* (PSO), dan *Ant Colony Optimization* (ACO).

Fireworks Algorithm (FWA) merupakan salah satu algoritma metaheuristik yang diperkenalkan oleh Ying Tan dan Yuanchun Zhu pada tahun 2010. Algoritma ini menggunakan teknik eksplorasi dan eksploitasi yang terinspirasi oleh peristiwa ledakan kembang api. Percikan api yang dihasilkan oleh ledakan kembang api dapat dipandang sebagai sebuah proses pencarian di sekitar titik tertentu. Posisi di mana kembang api akan meledak, dianggap sebagai suatu individu yang dinamakan kembang api. Di sekitar individu tersebut akan bermunculan beberapa individu baru yang dinamakan *sparks* (percikan). *Sparks* pada setiap ledakan, akan menjadi kembang api pada ledakan berikutnya. Ledakan akan terus dihasilkan hingga suatu individu mengenai atau cukup dekat dengan titik solusi yang dicari. Menurut Tan dan Zhu (2010), FWA memiliki konvergensi yang sangat baik dan mendapatkan hasil yang lebih baik dibandingkan dengan PSO dan *Clonal Particle Swarm Optimization* (CPSO) dalam menyelesaikan permasalahan fungsi objektif.

FWA terdiri dari empat bagian yaitu, operator ledakan, operator mutasi, aturan pemetaan, dan strategi seleksi. Pada operator mutasi menggunakan mutasi Gaussian yang menghasilkan kinerja yang baik, tetapi tidak dapat menyelesaikan masalah secara efektif yang membuat waktu komputasi lebih lama, sehingga perlu menambah mekanisme baru untuk meningkatkan performa FWA. Kekurangan lain dari FWA yaitu pada strategi seleksi yang berbasis jarak sehingga membuat waktu komputasi semakin lama.

Algoritma *Grey Wolf Optimizer* (GWO) adalah algoritma metaheuristik yang diperkenalkan oleh Mirjalili, dkk (2014). Algoritma ini terinspirasi dari hierarki kepemimpinan dan perilaku berburu sekumpulan *Canis lupus* atau serigala abu-abu di alam. Hierarki kepemimpinan serigala abu-abu dibagi menjadi empat, yaitu serigala alpha, beta, delta, dan omega yang masing-masing memiliki peranan dan tanggung jawab berbeda. Solusi terbaik dalam algoritma GWO dianalogikan sebagai posisi serigala alpha. Tahapan pencarian solusi pada algoritma GWO dibagi menjadi dua tahap, yaitu eksplorasi dan eksploitasi.

Menurut Mirjalili, dkk (2014), GWO mampu memberikan hasil yang sangat baik dibandingkan dengan PSO, *Gravitational Search Algorithm* (GSA), *Differential Evolution* (DE), *Evolutionary Programming* (EP), dan *Evolution Strategy* (ES). GWO unggul untuk percobaan pada fungsi unimodal dan multimodal, waktu komputasi yang dibutuhkan jauh lebih singkat, dan nilai minimum yang didapatkan semakin optimal apabila dilakukan penambahan jumlah agen pencarian.

Hibridisasi adalah teknik perkawinan antara dua individu yang berlainan varietas dalam satu spesies. Teknik ini dilakukan dengan memperhatikan sifat-sifat yang baik dari individu-individu yang akan dihibrid (Fitiyani, dkk, 2018). Pada algoritma, hibridisasi bertujuan untuk menggabungkan dua algoritma untuk melengkapi kekurangan dari salah satu atau kedua algoritma agar mendapatkan hasil yang lebih baik.

Skripsi ini mengkaji kembali artikel Barraza, dkk (2018) mengenai pendekatan hibridisasi baru untuk menggabungkan dua algoritma metaheuristik, yaitu hibrid FWA-GWO. Tujuan utama dari hibridisasi antara kedua algoritma tersebut adalah untuk memperbaiki kekurangan yang dimiliki FWA dan mengambil keuntungan dari kelebihan algoritma GWO. Fitur-fitur yang digunakan untuk mencapai hibridisasi pada FWA, yaitu inisialisasi, ledakan amplitudo, dan lokasi. Pada GWO, yaitu hierarki kepemimpinan, interaksi di antara populasi, dan lokasi populasi berikutnya. Selain itu, algoritma hibrid FWA-GWO akan diuji menggunakan fungsi tes dan hasilnya akan dibandingkan dengan FWA dan GWO

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan di atas maka rumusan masalah dalam skripsi ini adalah sebagai berikut.

1. Bagaimana menyelesaikan masalah optimasi menggunakan algoritma hibrid FWA-GWO?
2. Bagaimana perbandingan kinerja algoritma hibrid FWA-GWO dengan algoritma FWA dan GWO?

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah di atas maka tujuan yang ingin dicapai dalam skripsi ini adalah sebagai berikut.

1. Mengkontruksi algoritma hibrid FWA-GWO untuk menyelesaikan masalah optimasi.
2. Membandingkan kinerja algoritma hibrid FWA-GWO dengan algoritma FWA dan GWO

BAB II
DASAR TEORI

2.1 Hadamard *Product*

Hadamard *product* (atau dikenal dengan Schur *product*) merupakan operasi biner yang melibatkan dua matriks atau dua vektor dengan dimensi yang sama. Operasi ini memiliki kemiripan dengan *dot product* atau hasil kali titik pada vektor, namun memiliki perbedaan dalam perhitungan sehingga untuk operasinya digunakan simbol $\circ$. Definisi Hadamard *product* dituliskan sebagai berikut.

Definisi 2.1.1 Operasi Hadamard *Product*
Jika terdapat A dan B yang merupakan dua buah matriks berdimensi $m \times n$, maka operasi Hadamard didefinisikan sebagai $(A \circ B)_{ij} = (A)_{ij}(B)_{ij}$ untuk semua $1 \leq i \leq m, 1 \leq j \leq n$ (Million, 2007).

Syarat dari perhitungan menggunakan Hadamard *product* yaitu kedua matriks atau vektor harus memiliki dimensi yang sama. Misalkan diberikan vektor $\vec{a}$ dan $\vec{b}$ berukuran 3×1 dengan, $\vec{a} = \begin{pmatrix} a_{11} \\ a_{21} \\ a_{31} \end{pmatrix}$ dan $\vec{b} = \begin{pmatrix} b_{11} \\ b_{21} \\ b_{31} \end{pmatrix}$, maka operasi vektor dengan Hadamard *product* untuk $\vec{a} \circ \vec{b}$ dihitung seperti berikut.

$$\vec{a} \circ \vec{b} = \begin{pmatrix} a_{11} \\ a_{21} \\ a_{31} \end{pmatrix} \circ \begin{pmatrix} b_{11} \\ b_{21} \\ b_{31} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} \\ a_{21}b_{21} \\ a_{31}b_{31} \end{pmatrix}.$$

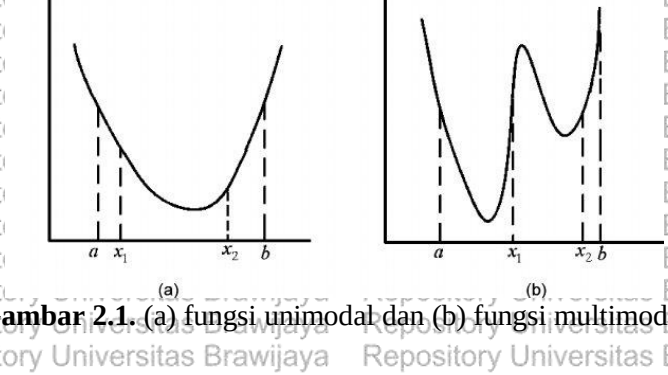
Operasi pada Hadamard *product*, baik dalam matriks maupun vektor, memiliki sifat asosiatif dengan persamaan $A \circ (B \circ C) = (A \circ B) \circ C$ dan distributif dengan persamaan $A \circ (B + C) = (A \circ B) + (A \circ C)$ (Horn dan Johnson, 2013).

2.2 Fungsi Unimodal dan Multimodal

Sebuah fungsi f adalah suatu aturan padanan yang menghubungkan himpunan objek $\vec{x} = \{x_1, x_2, \dots, x_n\}$ yang disebut daerah asal, dengan daerah hasil $f(\vec{x})$ dalam himpunan kedua.

Dalam masalah optimasi, fungsi biasanya bersifat unimodal atau multimodal. Seperti pada Gambar 2.1, fungsi unimodal yaitu

fungsi yang hanya memiliki satu puncak (titik maksimum) atau satu lembah (titik minimum) dalam suatu interval domain, sedangkan fungsi multimodal yaitu fungsi yang memiliki lebih dari satu titik maksimum atau titik minimum dalam interval domain (Rao, 2009).



Gambar 2.1. (a) fungsi unimodal dan (b) fungsi multimodal

2.3 Optimasi

Optimasi adalah proses membuat sesuatu menjadi lebih baik (Rajabioun, 2011). Dalam kehidupan sehari-hari seringkali ditemui masalah optimasi yang membutuhkan alat atau metode optimasi terbaik dengan harapan agar menghasilkan *output* yang maksimum atau minimum. Masalah optimasi dapat ditemui dalam berbagai bidang antara lain ekonomi, keuangan, transportasi, persediaan, dan sains komputasi.

Berdasarkan fungsinya optimasi dibedakan menjadi dua, yaitu *linear programming* dan *nonlinear programming*. *Linear programming* adalah salah satu metode optimasi dengan fungsi objektif dan kendala yang berbentuk linear, sedangkan *nonlinear programming* memiliki fungsi objektif atau kendala yang bersifat nonlinear (Yang, dkk, 2010).

Penyelesaian model optimasi nonlinear untuk meminimumkan fungsi objektif dengan kendala memiliki bentuk umum sebagai berikut.

$min\, f(\vec{x}), \vec{x} = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$ dengan kendala

$$h_i(\vec{x}) = 0, i = 1,2, \dots, w$$
$$g_i(\vec{x}) \leq 0, i = 1,2, \dots, r \quad (2.1)$$

dengan f adalah fungsi objektif, $h_i(\vec{x}) = 0$ adalah kendala persamaan, dan $g_i(\vec{x}) \leq 0$ adalah kendala pertidaksamaan. Berdasarkan jumlah kendala optimasi dibedakan menjadi dua, yaitu jika $w = r = 0$

disebut masalah optimasi tanpa kendala dan jika $w \neq 0$ atau $r \neq 0$ disebut masalah optimasi dengan kendala.

2.4 Titik Optimum Lokal dan Global

Pada proses pencarian nilai minimum fungsi terdapat dua jenis titik optimum, yaitu titik minimum lokal dan titik minimum global. Pengertian dari dua titik minimum tersebut dituliskan dalam Definisi 2.4.1 dan Definisi 2.4.2.

Definisi 2.4.1. Titik minimum lokal

Misalkan $f: X \rightarrow \mathbb{R}$ dengan $X \subseteq \mathbb{R}^n$, titik $\overrightarrow{x^*}$ disebut sebagai titik minimum lokal dari fungsi f jika terdapat $\delta > 0$ sedemikian sehingga memenuhi $\|\vec{x} - \overrightarrow{x^*}\| < \delta$ dan berlaku $f(\overrightarrow{x^*}) \leq f(\vec{x})$ (Sun dan Yuan, 2006).

Definisi 2.4.2. Titik minimum global

Misalkan $f: X \rightarrow \mathbb{R}$ dengan $X \subseteq \mathbb{R}^n$, $\forall \vec{x} \in X$, titik $\overrightarrow{x^*}$ disebut sebagai titik minimum global dari fungsi f jika berlaku $f(\overrightarrow{x^*}) \leq f(\vec{x})$ (Sun dan Yuan, 2006).

2.5 Eksplorasi dan Eksploitasi

Kinerja algoritma optimasi ditentukan oleh kemampuannya dalam menjelajahi (*explore*) dan mengeksploitasi (*exploit*) ruang pencarian (*search space*). Eksplorasi adalah proses pencarian solusi optimal di seluruh ruang dalam domain, sementara pencarian pada proses eksploitasi berfokus pada daerah solusi yang diperkirakan merupakan tempat titik optimal berada. Suatu algoritma optimasi dikatakan baik apabila proses eksplorasi dan eksploitasi dalam tiap iterasi berjalan secara seimbang.

2.6 *Fireworks Algorithm* (FWA)

Tan dan Zhu (2010) memperkenalkan suatu perluasan algoritma kecerdasan buatan yang terinspirasi oleh peristiwa ledakan kembang api di malam hari.

2.6.1 Inspirasi

Menyalakan kembang api merupakan kegiatan yang penting selama Festival Musim Semi di China. Puluhan ribu kembang api meledak di langit malam dan menunjukkan pola percikan yang indah. Harga dan spesifikasi menghasilkan pola percikan yang berbeda. Kembang api dengan harga lebih rendah menghasilkan lebih sedikit *sparks* dengan amplitudo lebih besar dibandingkan dengan kembang api dengan harga lebih tinggi (Tan, 2015).

Terdapat dua perilaku spesifik ledakan kembang api. Kembang api dengan harga lebih tinggi, banyak *spark* yang dihasilkan sehingga menghasilkan amplitudo yang kecil dan tampilan kembang api yang spektakuler. Untuk ledakan kembang api dengan harga yang lebih rendah, hanya sedikit *sparks* yang dihasilkan dengan amplitudo yang besar (Tan, 2015). Dua perilaku spesifik ledakan kembang api dapat dilihat pada Gambar 2.2.

2.6.2 Model matematika

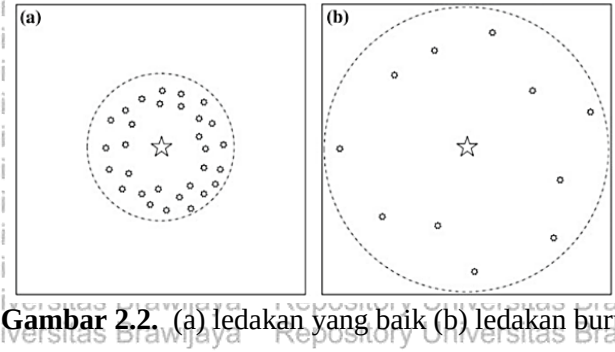
FWA terdiri dari empat bagian, yaitu operator ledakan, operator mutasi, aturan pemetaan, dan strategi seleksi.

2.6.2.1. Operator ledakan

Pada tahap inisialisasi, akan dibangkitkan n kembang api $(\vec{x}_i)$ $(i = 1,2, \dots, n,\ \vec{x} \in X)$ secara acak. Kembang api mempresentasikan titik-titik pada domain fungsi. Masing-masing kembang api akan menghasilkan *sparks* standar (m) dan amplitudo maksimal $(\hat{A})$. Operator ledakan terdiri dari jumlah *spark* (s_i), amplitudo ledakan (A_i), dan operator perpindahan (h). Operator-operator ini merupakan kunci dari FWA dan memiliki peranan yang sangat penting.

1. Jumlah *spark* (s_i)

   Masing-masing kembang api memiliki nilai *fitness* yang ditentukan dengan mensubstitusikannya ke fungsi objektif. Kembang api dengan nilai *fitness* yang lebih baik menghasilkan lebih banyak *spark* dalam amplitudo yang lebih kecil (ledakan yang baik) daripada kembang api dengan nilai *fitness* yang lebih buruk.



Gambar 2.2. (a) ledakan yang baik (b) ledakan buruk

Jumlah *spark* dapat ditentukan sebagai berikut,

$$s_i = m \ \frac{y_{max} - f(\vec{x}_i) + \varepsilon}{\sum_{i=1}^{n}\left(y_{max} - f(\vec{x}_i)\right) + \varepsilon}. \quad (2.2)$$

dengan s_i adalah jumlah *spark* untuk setiap kembang api, m adalah jumlah *spark* standar untuk setiap kembang api, $y_{max} = \max\left(f(\vec{x}_i)\right)(i = 1,2,\dots,n)$ adalah nilai maksimum dari fungsi objektif di antara n kembang api, dan ε digunakan untuk mencegah *error*.

Untuk menghindari jumlah *sparks* yang berlebihan, dibutuhkan pembatasan jumlah *sparks*.

$$\hat{S}_i = \begin{cases} round(am), & jika\ s_i < am \\ round(bm), & jika\ s_i > bm,\ a < b < 1 \\ round(s_i), & lainnya \end{cases} \quad (2.3)$$

di mana a adalah konstanta untuk mengontrol jumlah *spark* minimal pada setiap kembang api, *b* konstanta untuk mengontrol jumlah *spark* maksimal pada setiap kembang api, dan $\widehat{S}_i$ adalah batasan jumlah *sparks*.

2. Amplitudo ledakan (A_i)

Setelah menentukan jumlah *sparks*, selanjutnya akan dilakukan perhitungan untuk menentukan amplitudo ledakan kembang api. Jika amplitudo kembang api kecil dengan nilai *fitness* yang baik maka akan didapatkan optimum lokal dan global. Sebaliknya, kembang api

dengan nilai *fitness* yang buruk akan menghasilkan amplitudo yang besar.

Amplitudo setiap kembang api didefinisikan sebagai berikut,

$$A_i = \hat{A}\frac{f(\vec{x}_i) - y_{min} + \varepsilon}{\sum_{i=1}^{n}(f(\vec{x}_i) - y_{min}) + \varepsilon}. \quad (2.4)$$

di mana A_i merupakan amplitudo ledakan, $\hat{A}$ melambangkan amplitudo ledakan maksimum, dan $y_{min} = \min(f(\vec{x}_i))\,(i = 1,2,\dots,n)$ adalah nilai minimum dari fungsi objektif di antara n kembang api.

3. Operasi perpindahan (h)

Setelah menghitung amplitudo ledakan, perlu untuk menentukan perpindahan dalam amplitudo ledakan. FWA menggunakan perpindahan secara acak. Dengan cara ini, FWA menghasilkan perpindahan acak yang berbeda dalam setiap amplitudo untuk memastikan keragaman populasi.

Operasi perpindahan membuat perpindahan pada setiap dimensi kembang api, dapat didefinisikan sebagai berikut,

$$h = A_i\, rand(-1,1). \quad (2.5)$$

di mana h merupakan operasi perpindahan.

Melalui operator ledakan, setiap kembang api menghasilkan hujan *sparks* dan membantu menemukan optimum global fungsi.

2.6.2.2. Operator mutasi

Pada ledakan kembang api akan bermunculan *sparks* di persekitaran kembang api. Oleh karena itu, posisi *sparks* ($\tilde{\vec{x}}_i$) ditentukan sesuai dengan posisi kembang api.

Sebelum terjadi ledakan, posisi *sparks* setiap kembang api memiliki posisi yang sama dengan kembang api tersebut. Setelah terjadi ledakan, posisi setiap *sparks* akan berubah. Posisi setiap *sparks* setelah terjadi ledakan dapat ditentukan dengan mengganti elemen ke-z pada vektor posisi *sparks* menggunakan persamaan sebagai berikut,

$$\tilde{x}_{i,z} = \tilde{x}_{i,z} + h, \quad (2.6)$$

di mana $\tilde{x}_{i,z}$ merupakan elemen ke-z pada vektor posisi *spark* dan $i = 1,2,3,\dots,s_i$.

$$z = round\big(d.rand(0,1)\big), \tag{2.7}$$

di mana d merupakan dimensi dari fungsi objektif, dan $rand\ (0,1)$ merupakan bilangan acak (0,1), dan z merupakan suatu nilai yang dapat menentukan arah sparks setelah ledakan.

Untuk meningkatkan keragaman populasi digunakan mutasi Gaussian untuk menghasilkan *spark* sejumlah Gaussian *spark* $(\widehat{m})$ pada kembang api terbaik kedua. Untuk menentukan posisi Gaussian *spark* menggunakan persamaan sebagai berikut,

$$\tilde{x}_{i,z} = \tilde{x}_{i,z}.g, \tag{2.8}$$

di mana $Q_z = \tilde{x}_i, i = 1,2,\dots,\widehat{m}$ dan g merupakan nilai acak dari distribusi Gaussian dengan rata-rata 1 dan variansi 1.

$$g = \mathcal{N}(1,1). \tag{2.9}$$

Menurut Tan, dkk (2013) Gaussian *spark* $(\widehat{m})$ ditentukan pada saat inisalisasi parameter sejumlah $\widehat{m} = 5, m = 50$, dan $n = 5$.

2.6.2.3. Aturan pemetaan

Untuk memastikan semua individu berada di ruang yang layak diperlukan aturan pemetaan. Jika ada *sparks* berada di luar batas, *sparks* akan dipetakan kembali.

Aturan pemetaan dapat dinyatakan sebagai berikut,

$$\tilde{x}_i = X_{LB} + \tilde{x}_i.(X_{UB} - X_{LB}), \tag{2.10}$$

di mana $\tilde{x}_i$ melambangkan posisi setiap *sparks* yang berada di luar batas, X_{LB} merupakan batas bawah posisi kembang api, dan X_{UB} merupakan batas atas posisi kembang api.

2.6.2.4. Strategi seleksi

Pada awal setiap generasi ledakan, n lokasi harus dipilih untuk ledakan kembang api. Lokasi terbaik yaitu, lokasi dengan fungsi objektif terbaik akan disimpan untuk generasi ledakan selanjutnya.

Setelah itu, $n-1$ kembang api dipilih berdasarkan jarak kembang api ke kembang api lainnya dan juga jarak kembang api ke *sparks* kembang api tersebut. Jarak antara kembang api ke kembang api lainnya dan jarak kembang api ke *sparks* didefinisikan sebagai berikut,

$$R(\vec{x}_i) = \sum_{j \in N} d\left(\vec{x}_i, \vec{x}_j\right) = \sum_{j \in N} \left\| \vec{x}_i - \vec{x}_j \right\|, \quad (2.11)$$

di mana N adalah himpunan semua lokasi, baik kembang api dan *sparks*. Kemudian probabilitas pemilihan lokasi didefinisikan sebagai berikut.

$$p(\vec{x}_i) = \frac{R(\vec{x}_i)}{\sum_{j \in N} R\left(\vec{x}_j\right)}. \quad (2.12)$$

Dapat dilihat bahwa individu dengan jarak yang lebih besar akan memiliki lebih banyak kesempatan untuk pemilihan lokasi selanjutnya agar keragaman populasi dapat terjamin.

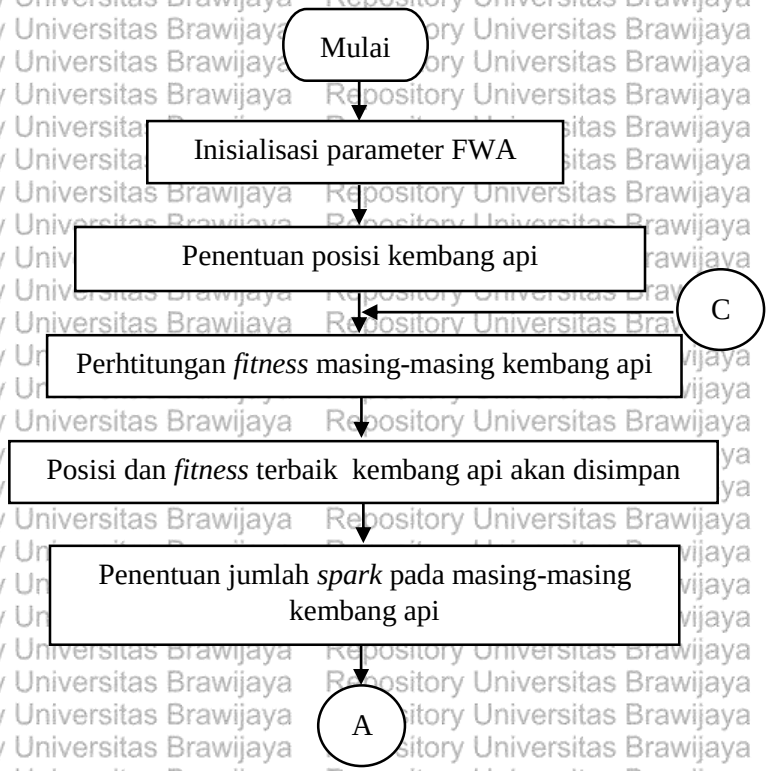
2.6.3 Algoritma

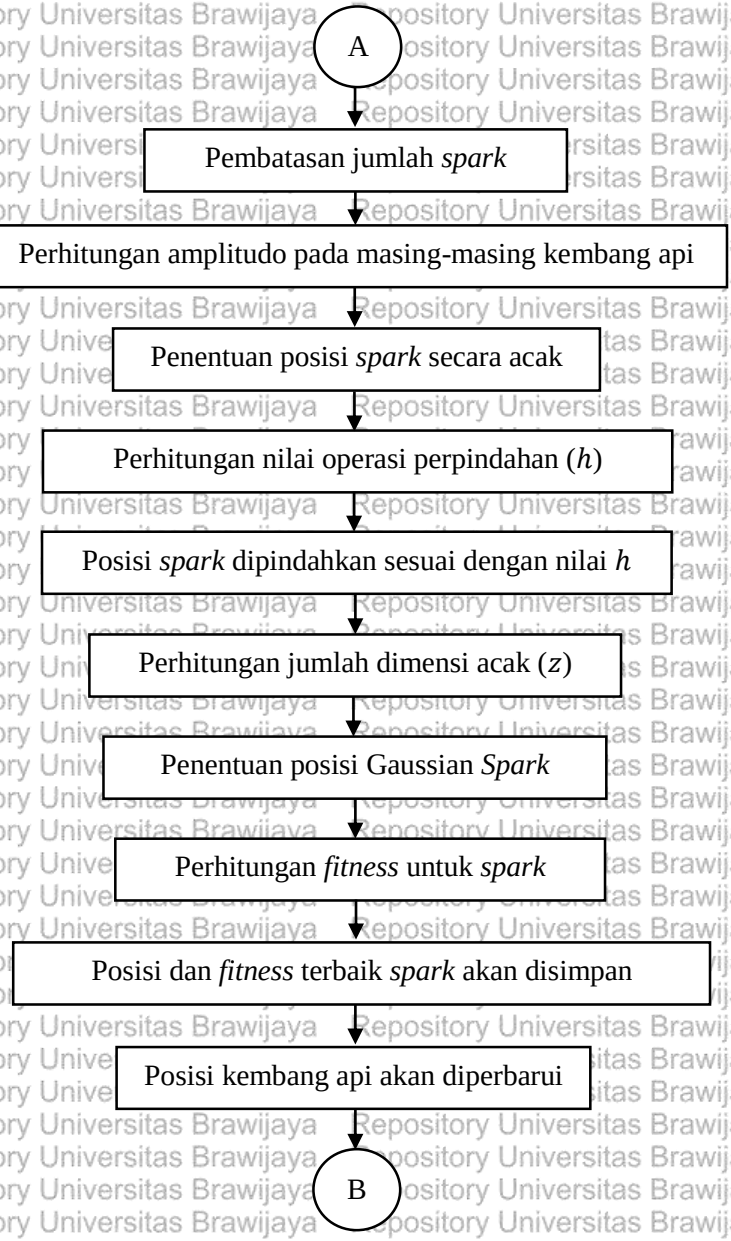
Algoritma untuk FWA (Tan, dkk, 2010) sebagai berikut.

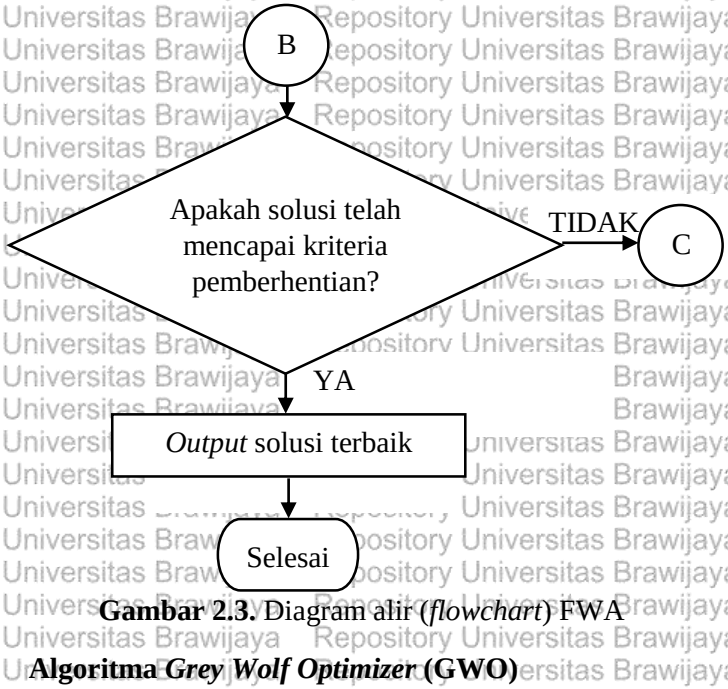
1. Inisialisasi parameter yang dibutuhkan FWA yaitu, jumlah kembang api yang ditentukan secara acak (n), jumlah *spark* standar setiap kembang api (m), jumlah Gaussian *spark* ($\hat{m}$), dan amplitudo maksimal setiap kembang api ($\hat{A}$).
2. Penentuan posisi kembang api secara acak pada setiap dimensi.
3. Perhitungan nilai *fitness* masing-masing kembang api pada posisi tersebut. Karena tujuan permasalahan adalah mencari nilai minimal, maka semakin rendah nilai *fitness* akan semakin baik.
4. Posisi dan *fitness* terbaik kembang api akan disimpan.
5. Penentuan jumlah *spark* pada masing-masing kembang api menggunakan persamaan (2.2).
6. Jumlah *spark* akan dibatasi menggunakan persamaan (2.3) agar tidak terjadi jumlah *spark* yang berlebihan.
7. Perhitungan amplitudo pada masing-masing kembang api menggunakan persamaan (2.4).
8. Perhitungan operasi perpindahan (h) menggunakan persamaan (2.5).
9. Perhitungan jumlah dimensi acak (z) menggunakan persamaan (2.7)

10. Penentuan posisi *spark* sesuai dengan posisi kembang api dengan mengganti elemen ke-*z* menggunakan persamaan (2.6).
11. Perhitungan nilai *fitness* untuk *spark*.
12. Peningkatan keragaman populasi dengan menentukan posisi Gaussian *spark* sesuai dengan posisi kembang api terbaik kedua dan mengganti nilai dimensi ke-*z* menggunakan persaman (2.8).
13. Perhitungan nilai *fitness* untuk Gaussian *spark*.
14. Posisi dan *fitness* terbaik *spark* akan disimpan.
15. Posisi kembang api akan diperbarui.
16. Jika solusi belum mencapai kriteria pemberhentian kembali ke Langkah 3
17. *Output* solusi terbaik.

Diagram alir (*flowchart)* FWA dapat dilihat pada Gambar 2.3.




A
Pembatasan jumlah *spark*
Perhitungan amplitudo pada masing-masing kembang api
Penentuan posisi *spark* secara acak
Perhitungan nilai operasi perpindahan (h)
Posisi *spark* dipindahkan sesuai dengan nilai h
Perhitungan jumlah dimensi acak (z)
Penentuan posisi Gaussian *Spark*
Perhitungan *fitness* untuk *spark*
Posisi dan *fitness* terbaik *spark* akan disimpan
Posisi kembang api akan diperbarui
B



Gambar 2.3. Diagram alir (*flowchart*) FWA

2.7 Algoritma *Grey Wolf Optimizer* (GWO)

Algoritma *Grey Wolf Optimizer* (GWO) adalah algoritma metaheuristik yang diperkenalkan oleh Seyedali Mirjalili pada tahun 2014.

2.6.1 Inspirasi

Grey Wolf (*Canis lupus*) termasuk famili *Canidae*. Serigala abu-abu biasanya merupakan puncak dari rantai makanan. Serigala abu-abu kebanyakan lebih suka hidup berkelompok. Yang menarik adalah bahwa serigala abu-abu memiliki hierarki dominan sosial yang sangat ketat. Hierarki kepemimpinan serigala abu-abu dibagi menjadi empat, yaitu serigala alpha (α), beta (β), delta (δ), dan omega (ω).

Serigala alpha bertanggung jawab dalam membuat keputusan mengenai berburu, tempat tidur, waktu untuk bangun, dan sebagainya. Serigala alpha juga disebut serigala dominan karena perintahnya yang harus diikuti oleh kelompok tersebut.

Tingkat kedua dalam hierarki serigala abu-abu adalah beta. Serigala beta bertugas untuk membantu serigala alpha dalam pengambilan keputusan, penasihat, mendisiplinkan kelompok, dan

kegiatan kelompok lainnya. Serigala beta merupakan kandidat terbaik untuk menjadi alpha jika salah satu serigala alpha meninggal.

Tingkat ketiga dalam hierarki serigala abu-abu adalah delta. Serigala delta atau *subordinate* harus tunduk kepada serigala alpha dan beta. Serigala delta memiliki empat kategori, yatu pengintai, penjaga, pemburu, dan pengasuh. Pengintai bertanggung jawab untuk mengawasi batas wilayah dan memperingatkan kelompok jika ada bahaya. Para penjaga melindungi dan menjamin seluruh kelompok. Pemburu membantu alpha dan beta saat berburu mangsa dan menyediakan makanan untuk kelompok. Pengasuh bertanggung jawab untuk merawat serigala yang lemah, sakit, dan terluka di dalam kelompok.

Tingkat terendah dalam hierarki serigala abu-abu adalah omega. Serigala omega harus selalu tunduk kepada semua serigala domain lainnya.

Selain hierarki sosial serigala, perburuan kelompok juga merupakan perilaku sosial serigala abu-abu yang menarik. Fase utama perburuan serigala abu-abu adalah melacak dan mengejar mangsa, mendekati dan mengelilingi mangsa hingga mangsa tidak bergerak, dan menyerang mangsa.

(Mirjalili, dkk, 2014)

2.6.2 Model Matematika

Serigala mengelilingi mangsa selama berburu dan perilaku mengelilingi dapat dimodelkan dengan persamaan di bawah ini.

$$\vec{D}(t) = \left|\vec{C}(t) \circ \overrightarrow{X_p}(t) - \vec{X}(t)\right|, \tag{2.13}$$

$$\vec{X}(t+1) = \overrightarrow{X_p}(t) - \vec{A}(t) \circ \vec{D}(t), \tag{2.14}$$

di mana $\overrightarrow{X_p}$ menunjukkan vektor posisi dari mangsa, $\vec{X}$ merupakan vektor posisi dari serigala, $\vec{D}$ merepresentasikan jarak yang harus ditempuh serigala untuk mencapai posisi mangsa, t adalah iterasi, $\vec{A}$ dan $\vec{C}$ menunjukkan vektor koefisien, dan $\circ$ merupakan Hadamard *Product* pada Definisi 2.1.1.

Vektor $\vec{A}$ dan $\vec{C}$ dapat dihitung sebagai berikut,

$$\vec{A}(t) = 2\vec{a}(t) \circ \overrightarrow{r_1} - \vec{a}(t), \tag{2.15}$$

$$\vec{C}(t) = 2 \circ \overrightarrow{r_2}, \tag{2.16}$$

di mana $\vec{a}(t) = \begin{bmatrix} b_1(t) \\ b_2(t) \\ \vdots \\ b_n(t) \end{bmatrix}$, $b_1(t) = b_2(t) = \cdots = b_n(t) = b(t)$,

$b(t) = 2 - \frac{2t}{t_{max}}$, t_{max} merupakan batas maksimum iterasi, dan $\overrightarrow{r_1}, \overrightarrow{r_2}$ adalah vektor random dengan nilai antara 0 sampai 1.

Perburuan biasanya dipandu oleh serigala α yang disebut pemimpin diikuti oleh β dan δ yang dapat juga berpartisipasi berburu. $\alpha, \beta,$ dan δ sebagai agen terbaik dan agen pemangsa lain dipaksa untuk memperbaharui posisinya tergantung posisi agen terbaik. Persamaan perburuan posisi seperti persamaan berikut,

$$\begin{aligned} \overrightarrow{D_\alpha}(t) &= \left|\overrightarrow{C_1}(t) \circ \overrightarrow{X_\alpha}(t) - \vec{X}(t)\right|, \\ \overrightarrow{D_\beta}(t) &= \left|\overrightarrow{C_2}(t) \circ \overrightarrow{X_\beta}(t) - \vec{X}(t)\right|, \\ \overrightarrow{D_\delta}(t) &= \left|\overrightarrow{C_3}(t) \circ \overrightarrow{X_\delta}(t) - \vec{X}(t)\right|, \end{aligned} \tag{2.17}$$

$$\begin{aligned} \overrightarrow{X_1}(t) &= \overrightarrow{X_\alpha}(t) - \overrightarrow{A_1}(t) \circ \overrightarrow{D_\alpha}(t), \\ \overrightarrow{X_2}(t) &= \overrightarrow{X_\beta}(t) - \overrightarrow{A_2}(t) \circ \overrightarrow{D_\beta}(t), \\ \overrightarrow{X_3}(t) &= \overrightarrow{X_\delta}(t) - \overrightarrow{A_3}(t) \circ \overrightarrow{D_\delta}(t), \end{aligned} \tag{2.18}$$

$$\vec{X}(t+1) = \frac{\overrightarrow{X_1}(t) + \overrightarrow{X_2}(t) + \overrightarrow{X_3}(t)}{3}. \tag{2.19}$$

Pada persamaan (2.19) dapat menemukan posisi selanjutnya dari solusi yang telah dievaluasi, yang pada dasarnya adalah rata-rata berdasarkan tiga serigala terbaik dalam kelompok.

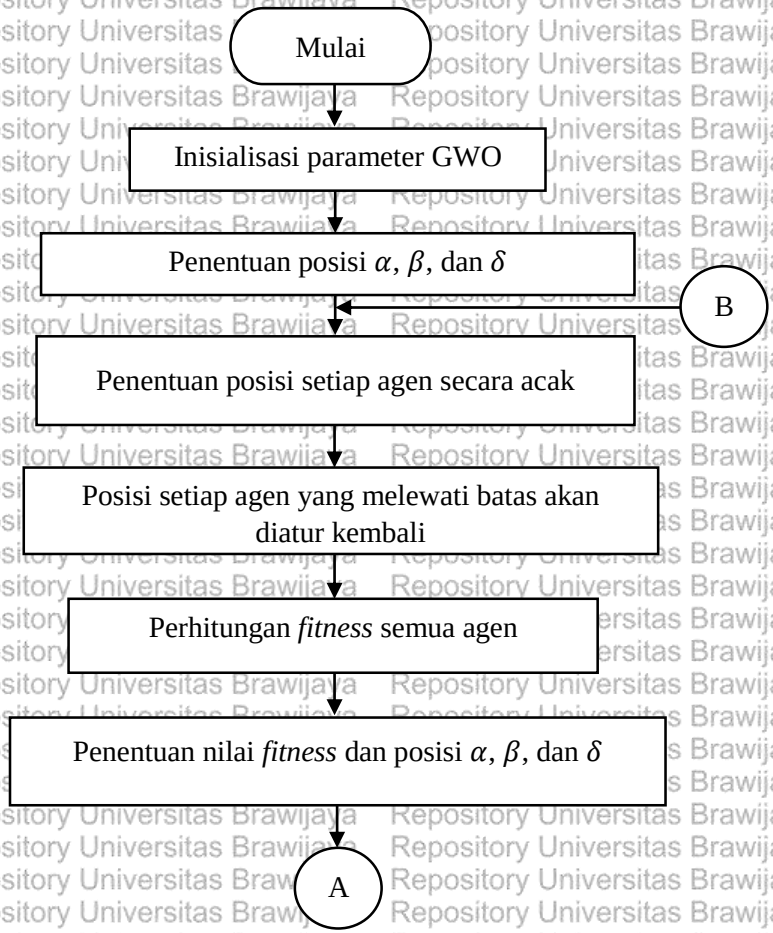
2.6.3 Algoritma

Algoritma untuk GWO (Mirjalili, dkk, 2014) sebagai berikut.

1. Inisialisasi parameter yang dibutuhkan Algoritma GWO.
2. Penentuan posisi awal α, β, dan δ.
3. Penentuan posisi setiap agen secara acak.
4. Pengaturan posisi setiap agen yang melewati batas wilayah.
5. Perhitungan nilai *fitness* semua agen berdasarkan fungsi uji.
6. Penentuan nilai *fitness* dan posisi α, β, dan δ sebagai tiga nilai *fitness* terbaik.

7. Posisi agen yang lain akan diperbarui mengikuti posisi α, β, dan δ menggunakan Persamaan (2.16), (2.17), dan (2.18).
8. Nilai *fitness* α disimpan sebagai solusi terbaik.
9. Perhitungan nilai *fitness* posisi yang baru.
10. Jika solusi belum mencapai kriteria pemberhentian kembali ke Langkah 3.
11. *Output* solusi terbaik.

Diagram alir (*flowchart)* FWA dapat dilihat pada Gambar 2.4.



Gambar 2.4. Diagram alir (*flowchart*) algoritma GWO

BAB II
DASAR TEORI

2.1 Hadamard *Product*

Hadamard *product* (atau dikenal dengan Schur *product*) merupakan operasi biner yang melibatkan dua matriks atau dua vektor dengan dimensi yang sama. Operasi ini memiliki kemiripan dengan *dot product* atau hasil kali titik pada vektor, namun memiliki perbedaan dalam perhitungan sehingga untuk operasinya digunakan simbol $\circ$. Definisi Hadamard *product* dituliskan sebagai berikut.

Definisi 2.1.1 Operasi Hadamard *Product*
Jika terdapat A dan B yang merupakan dua buah matriks berdimensi $m \times n$, maka operasi Hadamard didefinisikan sebagai $(A \circ B)_{ij} = (A)_{ij}(B)_{ij}$ untuk semua $1 \leq i \leq m, 1 \leq j \leq n$ (Million, 2007).

Syarat dari perhitungan menggunakan Hadamard *product* yaitu kedua matriks atau vektor harus memiliki dimensi yang sama. Misalkan diberikan vektor $\vec{a}$ dan $\vec{b}$ berukuran 3×1 dengan, $\vec{a} = \begin{pmatrix} a_{11} \\ a_{21} \\ a_{31} \end{pmatrix}$ dan $\vec{b} = \begin{pmatrix} b_{11} \\ b_{21} \\ b_{31} \end{pmatrix}$, maka operasi vektor dengan Hadamard *product* untuk $\vec{a} \circ \vec{b}$ dihitung seperti berikut.

$$\vec{a} \circ \vec{b} = \begin{pmatrix} a_{11} \\ a_{21} \\ a_{31} \end{pmatrix} \circ \begin{pmatrix} b_{11} \\ b_{21} \\ b_{31} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} \\ a_{21}b_{21} \\ a_{31}b_{31} \end{pmatrix}.$$

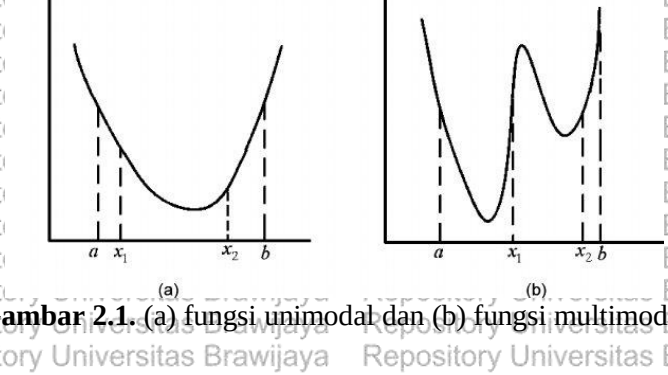
Operasi pada Hadamard *product*, baik dalam matriks maupun vektor, memiliki sifat asosiatif dengan persamaan $A \circ (B \circ C) = (A \circ B) \circ C$ dan distributif dengan persamaan $A \circ (B + C) = (A \circ B) + (A \circ C)$ (Horn dan Johnson, 2013).

2.2 Fungsi Unimodal dan Multimodal

Sebuah fungsi f adalah suatu aturan padanan yang menghubungkan himpunan objek $\vec{x} = \{x_1, x_2, \dots, x_n\}$ yang disebut daerah asal, dengan daerah hasil $f(\vec{x})$ dalam himpunan kedua.

Dalam masalah optimasi, fungsi biasanya bersifat unimodal atau multimodal. Seperti pada Gambar 2.1, fungsi unimodal yaitu

fungsi yang hanya memiliki satu puncak (titik maksimum) atau satu lembah (titik minimum) dalam suatu interval domain, sedangkan fungsi multimodal yaitu fungsi yang memiliki lebih dari satu titik maksimum atau titik minimum dalam interval domain (Rao, 2009).



Gambar 2.1. (a) fungsi unimodal dan (b) fungsi multimodal

2.3 Optimasi

Optimasi adalah proses membuat sesuatu menjadi lebih baik (Rajabioun, 2011). Dalam kehidupan sehari-hari seringkali ditemui masalah optimasi yang membutuhkan alat atau metode optimasi terbaik dengan harapan agar menghasilkan *output* yang maksimum atau minimum. Masalah optimasi dapat ditemui dalam berbagai bidang antara lain ekonomi, keuangan, transportasi, persediaan, dan sains komputasi.

Berdasarkan fungsinya optimasi dibedakan menjadi dua, yaitu *linear programming* dan *nonlinear programming*. *Linear programming* adalah salah satu metode optimasi dengan fungsi objektif dan kendala yang berbentuk linear, sedangkan *nonlinear programming* memiliki fungsi objektif atau kendala yang bersifat nonlinear (Yang, dkk, 2010).

Penyelesaian model optimasi nonlinear untuk meminimumkan fungsi objektif dengan kendala memiliki bentuk umum sebagai berikut.

$min\, f(\vec{x}), \vec{x} = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$ dengan kendala

$$h_i(\vec{x}) = 0, i = 1,2, \dots, w$$
$$g_i(\vec{x}) \leq 0, i = 1,2, \dots, r \quad (2.1)$$

dengan f adalah fungsi objektif, $h_i(\vec{x}) = 0$ adalah kendala persamaan, dan $g_i(\vec{x}) \leq 0$ adalah kendala pertidaksamaan. Berdasarkan jumlah kendala optimasi dibedakan menjadi dua, yaitu jika $w = r = 0$

disebut masalah optimasi tanpa kendala dan jika $w \neq 0$ atau $r \neq 0$ disebut masalah optimasi dengan kendala.

2.4 Titik Optimum Lokal dan Global

Pada proses pencarian nilai minimum fungsi terdapat dua jenis titik optimum, yaitu titik minimum lokal dan titik minimum global. Pengertian dari dua titik minimum tersebut dituliskan dalam Definisi 2.4.1 dan Definisi 2.4.2.

Definisi 2.4.1. Titik minimum lokal

Misalkan $f: X \rightarrow \mathbb{R}$ dengan $X \subseteq \mathbb{R}^n$, titik $\overrightarrow{x^*}$ disebut sebagai titik minimum lokal dari fungsi f jika terdapat $\delta > 0$ sedemikian sehingga memenuhi $\|\vec{x} - \overrightarrow{x^*}\| < \delta$ dan berlaku $f(\overrightarrow{x^*}) \leq f(\vec{x})$ (Sun dan Yuan, 2006).

Definisi 2.4.2. Titik minimum global

Misalkan $f: X \rightarrow \mathbb{R}$ dengan $X \subseteq \mathbb{R}^n$, $\forall \vec{x} \in X$, titik $\overrightarrow{x^*}$ disebut sebagai titik minimum global dari fungsi f jika berlaku $f(\overrightarrow{x^*}) \leq f(\vec{x})$ (Sun dan Yuan, 2006).

2.5 Eksplorasi dan Eksploitasi

Kinerja algoritma optimasi ditentukan oleh kemampuannya dalam menjelajahi (*explore*) dan mengeksploitasi (*exploit*) ruang pencarian (*search space*). Eksplorasi adalah proses pencarian solusi optimal di seluruh ruang dalam domain, sementara pencarian pada proses eksploitasi berfokus pada daerah solusi yang diperkirakan merupakan tempat titik optimal berada. Suatu algoritma optimasi dikatakan baik apabila proses eksplorasi dan eksploitasi dalam tiap iterasi berjalan secara seimbang.

2.6 *Fireworks Algorithm* (FWA)

Tan dan Zhu (2010) memperkenalkan suatu perluasan algoritma kecerdasan buatan yang terinspirasi oleh peristiwa ledakan kembang api di malam hari.

2.6.1 Inspirasi

Menyalakan kembang api merupakan kegiatan yang penting selama Festival Musim Semi di China. Puluhan ribu kembang api meledak di langit malam dan menunjukkan pola percikan yang indah. Harga dan spesifikasi menghasilkan pola percikan yang berbeda. Kembang api dengan harga lebih rendah menghasilkan lebih sedikit *sparks* dengan amplitudo lebih besar dibandingkan dengan kembang api dengan harga lebih tinggi (Tan, 2015).

Terdapat dua perilaku spesifik ledakan kembang api. Kembang api dengan harga lebih tinggi, banyak *spark* yang dihasilkan sehingga menghasilkan amplitudo yang kecil dan tampilan kembang api yang spektakuler. Untuk ledakan kembang api dengan harga yang lebih rendah, hanya sedikit *sparks* yang dihasilkan dengan amplitudo yang besar (Tan, 2015). Dua perilaku spesifik ledakan kembang api dapat dilihat pada Gambar 2.2.

2.6.2 Model matematika

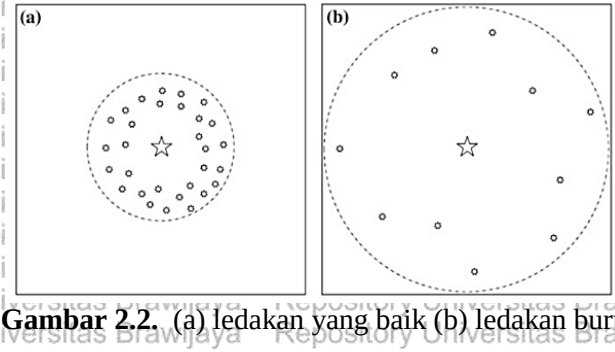
FWA terdiri dari empat bagian, yaitu operator ledakan, operator mutasi, aturan pemetaan, dan strategi seleksi.

2.6.2.1. Operator ledakan

Pada tahap inisialisasi, akan dibangkitkan n kembang api $(\vec{x}_i)$ $(i = 1,2, \dots, n,\ \vec{x} \in X)$ secara acak. Kembang api mempresentasikan titik-titik pada domain fungsi. Masing-masing kembang api akan menghasilkan *sparks* standar (m) dan amplitudo maksimal $(\hat{A})$. Operator ledakan terdiri dari jumlah *spark* (s_i), amplitudo ledakan (A_i), dan operator perpindahan (h). Operator-operator ini merupakan kunci dari FWA dan memiliki peranan yang sangat penting.

1. Jumlah *spark* (s_i)

   Masing-masing kembang api memiliki nilai *fitness* yang ditentukan dengan mensubstitusikannya ke fungsi objektif. Kembang api dengan nilai *fitness* yang lebih baik menghasilkan lebih banyak *spark* dalam amplitudo yang lebih kecil (ledakan yang baik) daripada kembang api dengan nilai *fitness* yang lebih buruk.



Gambar 2.2. (a) ledakan yang baik (b) ledakan buruk

Jumlah *spark* dapat ditentukan sebagai berikut,

$$s_i = m \ \frac{y_{max} - f(\vec{x}_i) + \varepsilon}{\sum_{i=1}^{n}\left(y_{max} - f(\vec{x}_i)\right) + \varepsilon}. \quad (2.2)$$

dengan s_i adalah jumlah *spark* untuk setiap kembang api, m adalah jumlah *spark* standar untuk setiap kembang api, $y_{max} = \max\left(f(\vec{x}_i)\right)(i = 1,2,\dots,n)$ adalah nilai maksimum dari fungsi objektif di antara n kembang api, dan ε digunakan untuk mencegah *error*.

Untuk menghindari jumlah *sparks* yang berlebihan, dibutuhkan pembatasan jumlah *sparks*.

$$\hat{S}_i = \begin{cases} round(am), & jika\ s_i < am \\ round(bm), & jika\ s_i > bm,\ a < b < 1 \\ round(s_i), & lainnya \end{cases} \quad (2.3)$$

di mana a adalah konstanta untuk mengontrol jumlah *spark* minimal pada setiap kembang api, *b* konstanta untuk mengontrol jumlah *spark* maksimal pada setiap kembang api, dan $\widehat{S}_i$ adalah batasan jumlah *sparks*.

2. Amplitudo ledakan (A_i)

Setelah menentukan jumlah *sparks*, selanjutnya akan dilakukan perhitungan untuk menentukan amplitudo ledakan kembang api. Jika amplitudo kembang api kecil dengan nilai *fitness* yang baik maka akan didapatkan optimum lokal dan global. Sebaliknya, kembang api

dengan nilai *fitness* yang buruk akan menghasilkan amplitudo yang besar.

Amplitudo setiap kembang api didefinisikan sebagai berikut,

$$A_i = \hat{A}\frac{f(\vec{x}_i) - y_{min} + \varepsilon}{\sum_{i=1}^{n}(f(\vec{x}_i) - y_{min}) + \varepsilon}. \quad (2.4)$$

di mana A_i merupakan amplitudo ledakan, $\hat{A}$ melambangkan amplitudo ledakan maksimum, dan $y_{min} = \min(f(\vec{x}_i))\,(i = 1,2,\dots,n)$ adalah nilai minimum dari fungsi objektif di antara n kembang api.

3. Operasi perpindahan (h)

Setelah menghitung amplitudo ledakan, perlu untuk menentukan perpindahan dalam amplitudo ledakan. FWA menggunakan perpindahan secara acak. Dengan cara ini, FWA menghasilkan perpindahan acak yang berbeda dalam setiap amplitudo untuk memastikan keragaman populasi.

Operasi perpindahan membuat perpindahan pada setiap dimensi kembang api, dapat didefinisikan sebagai berikut,

$$h = A_i\, rand(-1,1). \quad (2.5)$$

di mana h merupakan operasi perpindahan.

Melalui operator ledakan, setiap kembang api menghasilkan hujan *sparks* dan membantu menemukan optimum global fungsi.

2.6.2.2. Operator mutasi

Pada ledakan kembang api akan bermunculan *sparks* di persekitaran kembang api. Oleh karena itu, posisi *sparks* ($\tilde{\vec{x}}_i$) ditentukan sesuai dengan posisi kembang api.

Sebelum terjadi ledakan, posisi *sparks* setiap kembang api memiliki posisi yang sama dengan kembang api tersebut. Setelah terjadi ledakan, posisi setiap *sparks* akan berubah. Posisi setiap *sparks* setelah terjadi ledakan dapat ditentukan dengan mengganti elemen ke-z pada vektor posisi *sparks* menggunakan persamaan sebagai berikut,

$$\tilde{x}_{i,z} = \tilde{x}_{i,z} + h, \quad (2.6)$$

di mana $\tilde{x}_{i,z}$ merupakan elemen ke-z pada vektor posisi *spark* dan $i = 1,2,3,\dots,s_i$.

$$z = round\big(d.rand(0,1)\big), \tag{2.7}$$

di mana d merupakan dimensi dari fungsi objektif, dan $rand\ (0,1)$ merupakan bilangan acak (0,1), dan z merupakan suatu nilai yang dapat menentukan arah sparks setelah ledakan.

Untuk meningkatkan keragaman populasi digunakan mutasi Gaussian untuk menghasilkan *spark* sejumlah Gaussian *spark* $(\widehat{m})$ pada kembang api terbaik kedua. Untuk menentukan posisi Gaussian *spark* menggunakan persamaan sebagai berikut,

$$\tilde{x}_{i,z} = \tilde{x}_{i,z}.g, \tag{2.8}$$

di mana $Q_z = \tilde{x}_i, i = 1,2,\dots,\widehat{m}$ dan g merupakan nilai acak dari distribusi Gaussian dengan rata-rata 1 dan variansi 1.

$$g = \mathcal{N}(1,1). \tag{2.9}$$

Menurut Tan, dkk (2013) Gaussian *spark* $(\widehat{m})$ ditentukan pada saat inisalisasi parameter sejumlah $\widehat{m} = 5, m = 50$, dan $n = 5$.

2.6.2.3. Aturan pemetaan

Untuk memastikan semua individu berada di ruang yang layak diperlukan aturan pemetaan. Jika ada *sparks* berada di luar batas, *sparks* akan dipetakan kembali.

Aturan pemetaan dapat dinyatakan sebagai berikut,

$$\tilde{x}_i = X_{LB} + \tilde{x}_i.(X_{UB} - X_{LB}), \tag{2.10}$$

di mana $\tilde{x}_i$ melambangkan posisi setiap *sparks* yang berada di luar batas, X_{LB} merupakan batas bawah posisi kembang api, dan X_{UB} merupakan batas atas posisi kembang api.

2.6.2.4. Strategi seleksi

Pada awal setiap generasi ledakan, n lokasi harus dipilih untuk ledakan kembang api. Lokasi terbaik yaitu, lokasi dengan fungsi objektif terbaik akan disimpan untuk generasi ledakan selanjutnya.

Setelah itu, $n-1$ kembang api dipilih berdasarkan jarak kembang api ke kembang api lainnya dan juga jarak kembang api ke *sparks* kembang api tersebut. Jarak antara kembang api ke kembang api lainnya dan jarak kembang api ke *sparks* didefinisikan sebagai berikut,

$$R(\vec{x}_i) = \sum_{j \in N} d\left(\vec{x}_i, \vec{x}_j\right) = \sum_{j \in N} \left\| \vec{x}_i - \vec{x}_j \right\|, \quad (2.11)$$

di mana N adalah himpunan semua lokasi, baik kembang api dan *sparks*. Kemudian probabilitas pemilihan lokasi didefinisikan sebagai berikut.

$$p(\vec{x}_i) = \frac{R(\vec{x}_i)}{\sum_{j \in N} R\left(\vec{x}_j\right)}. \quad (2.12)$$

Dapat dilihat bahwa individu dengan jarak yang lebih besar akan memiliki lebih banyak kesempatan untuk pemilihan lokasi selanjutnya agar keragaman populasi dapat terjamin.

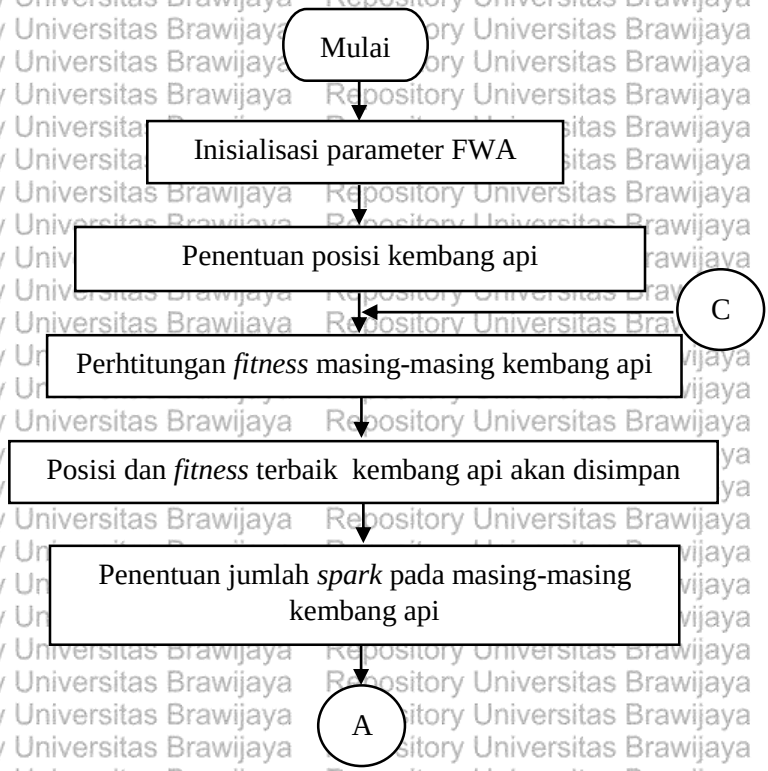
2.6.3 Algoritma

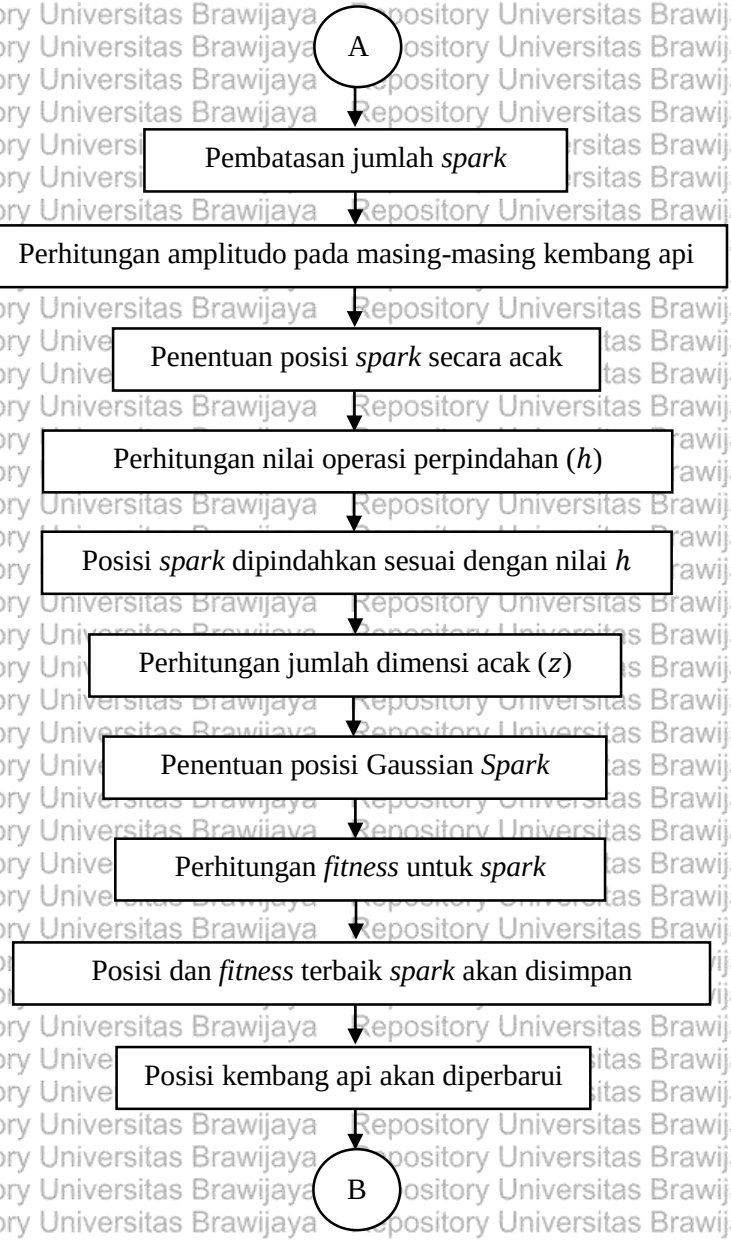
Algoritma untuk FWA (Tan, dkk, 2010) sebagai berikut.

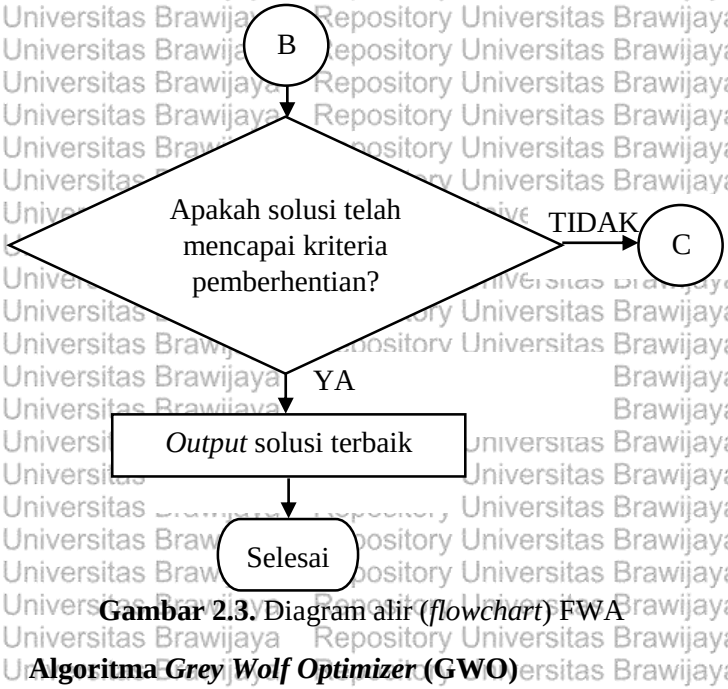
1. Inisialisasi parameter yang dibutuhkan FWA yaitu, jumlah kembang api yang ditentukan secara acak (n), jumlah *spark* standar setiap kembang api (m), jumlah Gaussian *spark* ($\hat{m}$), dan amplitudo maksimal setiap kembang api ($\hat{A}$).
2. Penentuan posisi kembang api secara acak pada setiap dimensi.
3. Perhitungan nilai *fitness* masing-masing kembang api pada posisi tersebut. Karena tujuan permasalahan adalah mencari nilai minimal, maka semakin rendah nilai *fitness* akan semakin baik.
4. Posisi dan *fitness* terbaik kembang api akan disimpan.
5. Penentuan jumlah *spark* pada masing-masing kembang api menggunakan persamaan (2.2).
6. Jumlah *spark* akan dibatasi menggunakan persamaan (2.3) agar tidak terjadi jumlah *spark* yang berlebihan.
7. Perhitungan amplitudo pada masing-masing kembang api menggunakan persamaan (2.4).
8. Perhitungan operasi perpindahan (h) menggunakan persamaan (2.5).
9. Perhitungan jumlah dimensi acak (z) menggunakan persamaan (2.7)

10. Penentuan posisi *spark* sesuai dengan posisi kembang api dengan mengganti elemen ke-*z* menggunakan persamaan (2.6).
11. Perhitungan nilai *fitness* untuk *spark*.
12. Peningkatan keragaman populasi dengan menentukan posisi Gaussian *spark* sesuai dengan posisi kembang api terbaik kedua dan mengganti nilai dimensi ke-*z* menggunakan persaman (2.8).
13. Perhitungan nilai *fitness* untuk Gaussian *spark*.
14. Posisi dan *fitness* terbaik *spark* akan disimpan.
15. Posisi kembang api akan diperbarui.
16. Jika solusi belum mencapai kriteria pemberhentian kembali ke Langkah 3
17. *Output* solusi terbaik.

Diagram alir (*flowchart)* FWA dapat dilihat pada Gambar 2.3.




A
Pembatasan jumlah *spark*
Perhitungan amplitudo pada masing-masing kembang api
Penentuan posisi *spark* secara acak
Perhitungan nilai operasi perpindahan (h)
Posisi *spark* dipindahkan sesuai dengan nilai h
Perhitungan jumlah dimensi acak (z)
Penentuan posisi Gaussian *Spark*
Perhitungan *fitness* untuk *spark*
Posisi dan *fitness* terbaik *spark* akan disimpan
Posisi kembang api akan diperbarui
B



Gambar 2.3. Diagram alir (*flowchart*) FWA

2.7 Algoritma *Grey Wolf Optimizer* (GWO)

Algoritma *Grey Wolf Optimizer* (GWO) adalah algoritma metaheuristik yang diperkenalkan oleh Seyedali Mirjalili pada tahun 2014.

2.6.1 Inspirasi

Grey Wolf (*Canis lupus*) termasuk famili *Canidae*. Serigala abu-abu biasanya merupakan puncak dari rantai makanan. Serigala abu-abu kebanyakan lebih suka hidup berkelompok. Yang menarik adalah bahwa serigala abu-abu memiliki hierarki dominan sosial yang sangat ketat. Hierarki kepemimpinan serigala abu-abu dibagi menjadi empat, yaitu serigala alpha (α), beta (β), delta (δ), dan omega (ω).

Serigala alpha bertanggung jawab dalam membuat keputusan mengenai berburu, tempat tidur, waktu untuk bangun, dan sebagainya. Serigala alpha juga disebut serigala dominan karena perintahnya yang harus diikuti oleh kelompok tersebut.

Tingkat kedua dalam hierarki serigala abu-abu adalah beta. Serigala beta bertugas untuk membantu serigala alpha dalam pengambilan keputusan, penasihat, mendisiplinkan kelompok, dan

kegiatan kelompok lainnya. Serigala beta merupakan kandidat terbaik untuk menjadi alpha jika salah satu serigala alpha meninggal.

Tingkat ketiga dalam hierarki serigala abu-abu adalah delta. Serigala delta atau *subordinate* harus tunduk kepada serigala alpha dan beta. Serigala delta memiliki empat kategori, yatu pengintai, penjaga, pemburu, dan pengasuh. Pengintai bertanggung jawab untuk mengawasi batas wilayah dan memperingatkan kelompok jika ada bahaya. Para penjaga melindungi dan menjamin seluruh kelompok. Pemburu membantu alpha dan beta saat berburu mangsa dan menyediakan makanan untuk kelompok. Pengasuh bertanggung jawab untuk merawat serigala yang lemah, sakit, dan terluka di dalam kelompok.

Tingkat terendah dalam hierarki serigala abu-abu adalah omega. Serigala omega harus selalu tunduk kepada semua serigala domain lainnya.

Selain hierarki sosial serigala, perburuan kelompok juga merupakan perilaku sosial serigala abu-abu yang menarik. Fase utama perburuan serigala abu-abu adalah melacak dan mengejar mangsa, mendekati dan mengelilingi mangsa hingga mangsa tidak bergerak, dan menyerang mangsa.

(Mirjalili, dkk, 2014)

2.6.2 Model Matematika

Serigala mengelilingi mangsa selama berburu dan perilaku mengelilingi dapat dimodelkan dengan persamaan di bawah ini.

$$\vec{D}(t) = \left|\vec{C}(t) \circ \overrightarrow{X_p}(t) - \vec{X}(t)\right|, \tag{2.13}$$

$$\vec{X}(t+1) = \overrightarrow{X_p}(t) - \vec{A}(t) \circ \vec{D}(t), \tag{2.14}$$

di mana $\overrightarrow{X_p}$ menunjukkan vektor posisi dari mangsa, $\vec{X}$ merupakan vektor posisi dari serigala, $\vec{D}$ merepresentasikan jarak yang harus ditempuh serigala untuk mencapai posisi mangsa, t adalah iterasi, $\vec{A}$ dan $\vec{C}$ menunjukkan vektor koefisien, dan $\circ$ merupakan Hadamard *Product* pada Definisi 2.1.1.

Vektor $\vec{A}$ dan $\vec{C}$ dapat dihitung sebagai berikut,

$$\vec{A}(t) = 2\vec{a}(t) \circ \overrightarrow{r_1} - \vec{a}(t), \tag{2.15}$$

$$\vec{C}(t) = 2 \circ \overrightarrow{r_2}, \tag{2.16}$$

di mana $\vec{a}(t) = \begin{bmatrix} b_1(t) \\ b_2(t) \\ \vdots \\ b_n(t) \end{bmatrix}$, $b_1(t) = b_2(t) = \cdots = b_n(t) = b(t)$,

$b(t) = 2 - \frac{2t}{t_{max}}$, t_{max} merupakan batas maksimum iterasi, dan $\overrightarrow{r_1}, \overrightarrow{r_2}$ adalah vektor random dengan nilai antara 0 sampai 1.

Perburuan biasanya dipandu oleh serigala α yang disebut pemimpin diikuti oleh β dan δ yang dapat juga berpartisipasi berburu. $\alpha, \beta,$ dan δ sebagai agen terbaik dan agen pemangsa lain dipaksa untuk memperbaharui posisinya tergantung posisi agen terbaik. Persamaan perburuan posisi seperti persamaan berikut,

$$\begin{aligned} \overrightarrow{D_\alpha}(t) &= \left|\overrightarrow{C_1}(t) \circ \overrightarrow{X_\alpha}(t) - \vec{X}(t)\right|, \\ \overrightarrow{D_\beta}(t) &= \left|\overrightarrow{C_2}(t) \circ \overrightarrow{X_\beta}(t) - \vec{X}(t)\right|, \\ \overrightarrow{D_\delta}(t) &= \left|\overrightarrow{C_3}(t) \circ \overrightarrow{X_\delta}(t) - \vec{X}(t)\right|, \end{aligned} \tag{2.17}$$

$$\begin{aligned} \overrightarrow{X_1}(t) &= \overrightarrow{X_\alpha}(t) - \overrightarrow{A_1}(t) \circ \overrightarrow{D_\alpha}(t), \\ \overrightarrow{X_2}(t) &= \overrightarrow{X_\beta}(t) - \overrightarrow{A_2}(t) \circ \overrightarrow{D_\beta}(t), \\ \overrightarrow{X_3}(t) &= \overrightarrow{X_\delta}(t) - \overrightarrow{A_3}(t) \circ \overrightarrow{D_\delta}(t), \end{aligned} \tag{2.18}$$

$$\vec{X}(t+1) = \frac{\overrightarrow{X_1}(t) + \overrightarrow{X_2}(t) + \overrightarrow{X_3}(t)}{3}. \tag{2.19}$$

Pada persamaan (2.19) dapat menemukan posisi selanjutnya dari solusi yang telah dievaluasi, yang pada dasarnya adalah rata-rata berdasarkan tiga serigala terbaik dalam kelompok.

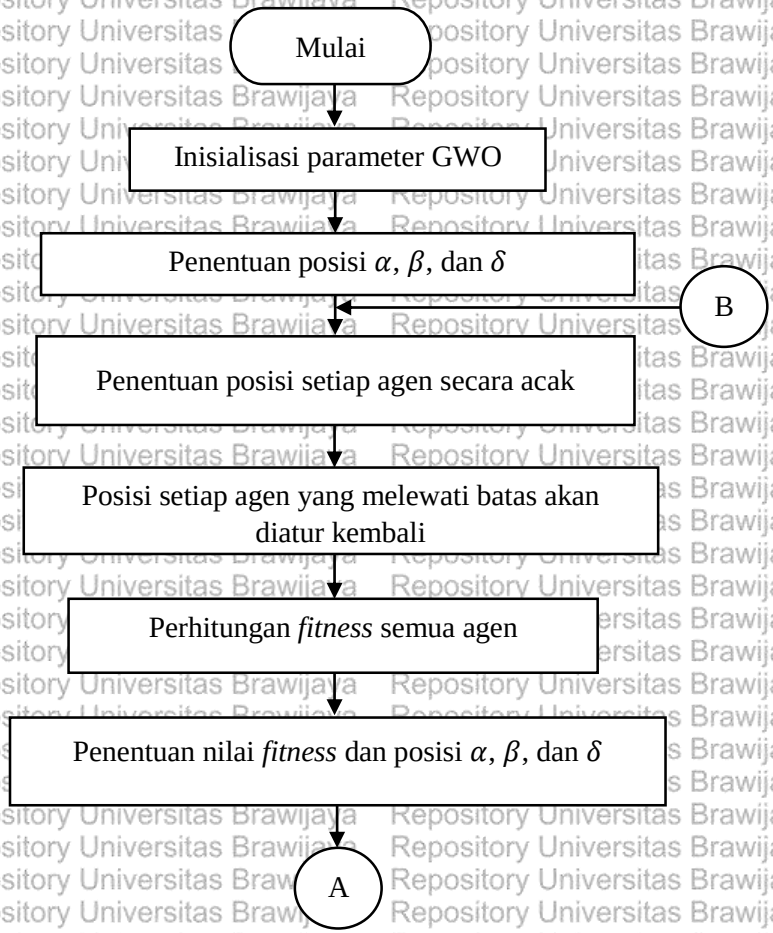
2.6.3 Algoritma

Algoritma untuk GWO (Mirjalili, dkk, 2014) sebagai berikut.

1. Inisialisasi parameter yang dibutuhkan Algoritma GWO.
2. Penentuan posisi awal α, β, dan δ.
3. Penentuan posisi setiap agen secara acak.
4. Pengaturan posisi setiap agen yang melewati batas wilayah.
5. Perhitungan nilai *fitness* semua agen berdasarkan fungsi uji.
6. Penentuan nilai *fitness* dan posisi α, β, dan δ sebagai tiga nilai *fitness* terbaik.

7. Posisi agen yang lain akan diperbarui mengikuti posisi α, β, dan δ menggunakan Persamaan (2.16), (2.17), dan (2.18).
8. Nilai *fitness* α disimpan sebagai solusi terbaik.
9. Perhitungan nilai *fitness* posisi yang baru.
10. Jika solusi belum mencapai kriteria pemberhentian kembali ke Langkah 3.
11. *Output* solusi terbaik.

Diagram alir (*flowchart)* FWA dapat dilihat pada Gambar 2.4.



Gambar 2.4. Diagram alir (*flowchart*) algoritma GWO

BAB III
METODE PENELITIAN

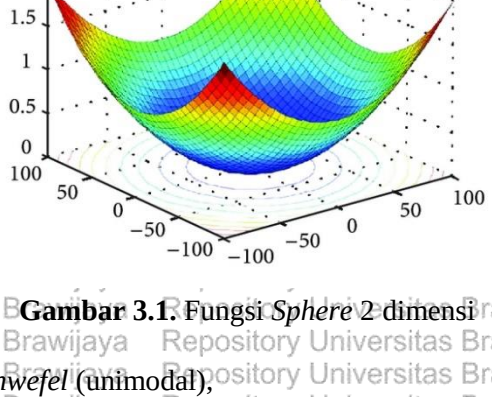
3.1 Pemilihan *Test Function*

Test function yang digunakan berupa enam fungsi uji yang dipilih dengan dua fungsi unimodal, dua fungsi multimodal, dan dua fungsi multimodal berdimensi tetap. Enam fungsi uji yang dipilih, yaitu:

1. Fungsi *Sphere* (unimodal),

$$f(\vec{x}) = \sum_{i=1}^{n} x_i^2,$$

dengan n adalah jumlah dimensi dari fungsi, domain $-100 \leq x_i \leq 100 \; \forall i \in \{1, \dots, n\}$, dan nilai minimum global $f(\vec{x}) = 0$ pada saat $\vec{x^*} = \vec{0}$. Fungsi ini akan diuji menggunakan dimensi $n = 30, 60,$ dan 90.



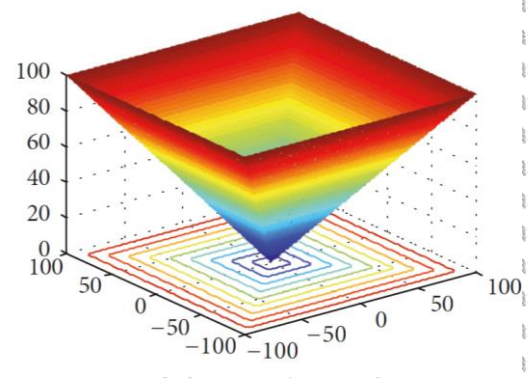
Gambar 3.1. Fungsi *Sphere* 2 dimensi

2. Fungsi *Schwefel* (unimodal),

$$f(\vec{x}) = \max_i \{|x_i|, 1 \leq i \leq n\},$$

dengan n adalah jumlah dimensi dari fungsi, domain $-100 \leq x_i \leq 100 \; \forall i \in \{1, \dots, n\}$, dan nilai minimum global $f(\vec{x}) = 0$

pada saat $\overrightarrow{x^*} = \vec{0}$. Fungsi ini akan diuji menggunakan dimensi $n = 30, 60,$ dan 90.

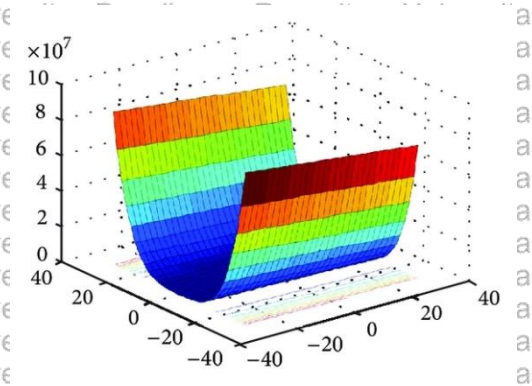


Gambar 3.2. Fungsi *Schwefel* 2 dimensi

3. fungsi *Rosenbrock* (multimodal),

$$f(\vec{x}) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2 \right],$$

dengan n adalah jumlah dimensi dari fungsi, domain $-30 \leq x_i \leq 30\ \forall i \in \{1, \ldots, n\}$, dan nilai minimum global $f(\vec{x}) = 0$ pada saat $\overrightarrow{x^*} = 1$. Fungsi ini akan diuji menggunakan dimensi $n = 30, 60,$ dan 90.

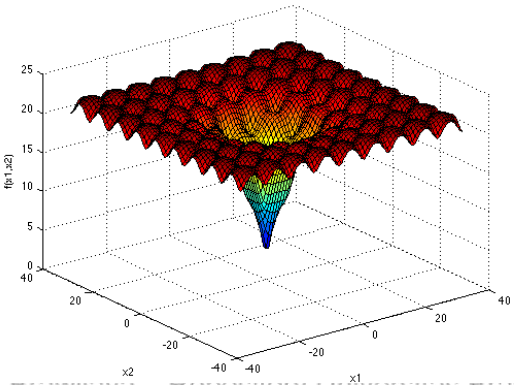


Gambar 3.3. Fungsi *Rosenbrock* 2 dimensi

4. fungsi *Ackley* (multimodal),

$$f(\vec{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i-1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i-1}^{n} cos(2\pi x_i)\right) + 20 + \exp(1),$$

dengan n adalah jumlah dimensi dari fungsi, domain $-32 \leq x_i \leq 32\ \forall i \in \{1, \ldots, n\}$, dan nilai minimum global $f(\vec{x}) = 0$ pada saat $\vec{x^*} = \vec{0}$. Fungsi ini akan diuji menggunakan dimensi $n = 30, 60,$ dan 90.

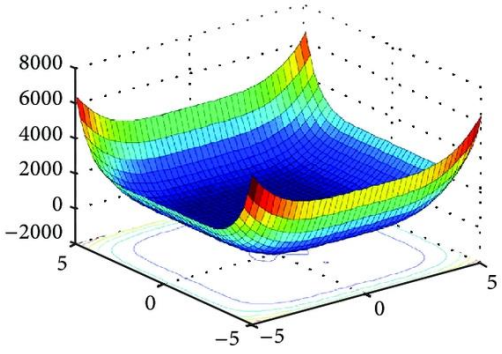


Gambar 3.4. Fungsi *Ackley* 2 dimensi

5. fungsi *Camel – Six Humps* (multimodal berdimensi tetap),

$$f(\vec{x}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4,$$

dengan domain $-5 \leq x_i \leq 5\ \forall i \in \{1,2\}$, dan nilai minimum global $f(\vec{x}) = -1.0316$ pada saat $\vec{x^*} = (-0.0898, 0.7126)$,

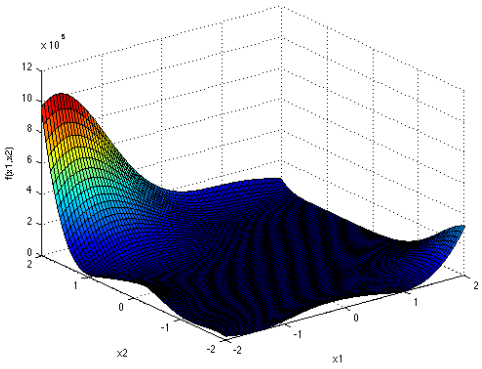


Gambar 3.5. Fungsi *Camel – Six* 2 Dimensi

6. fungsi *Goldstein Price* (multimodal berdimensi tetap)

$$f(\vec{x}) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$$

dengan domain $-2 \leq x_i \leq 2 \; \forall i \in \{1,2\}$, dan nilai minimum global $f(\vec{x}) = 3$ pada saat $\overrightarrow{x^*} = (0, -1)$.



Gambar 3.6. Fungsi *Goldstein Price* 2 Dimensi

3.2 Konstruksi Hibrid FWA-GWO

Pada tahap ini akan dibahas bagaimana mengkonstruksi hibrid FWA-GWO. Proses kontruksi hibrid FWA-GWO terbagi menjadi lima tahapan, yaitu inisialisasi, *update* posisi, menghitung nilai *fitness*, dan cek kriteria pemberhentian. Kriteria pemberhentian hibrid FWA-GWO jika iterasi telah mencapai batas maksimal. Parameter-parameter yang akan digunakan pada penelitian ini sesuai dengan parameter-parameter yang digunakan dalam penelitian Barraza, dkk (2018) yang dapat dilihat pada Tabel 3.1 dengan tiga versi yang berbeda, yaitu Versi 1, Versi 2, dan Versi 3.

Tabel 3.1. Tabel Parameter

Parameter	FWA			GWO			FWA-GWO		
	V1	V2	V3	V1	V2	V3	V1	V2	V3
Jumlah *Pack*	4	5	8	-	-	-	4	5	8
Jumlah *Wolf*	-	-	-	24	30	40	6	6	5
Iterasi maksimal	625	500	375	625	500	375	625	500	375

3.3 Pembuatan Program FWA, GWO, dan Hibrid FWA-GWO

Pembuatan program FWA, GWO dan hibrid FWA-GWO untuk penyelesaian optimasi fungsi menggunakan *software* MATLAB R2014b dan menggunakan sistem komputer dengan spesifikasi *processor* Intel(R) Core(TM) i3-7020U, CPU 2.30GHz, RAM 4GB, serta hard disk berukuran 500GB.

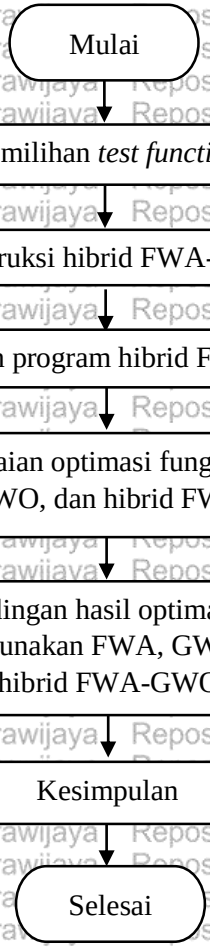
3.4 Perbandingan FWA, GWO, dan Hibrid FWA-GWO

Masing-masing algoritma akan dilakukan 30 simulasi. Hasil percobaan berupa nilai minimum dari fungsi uji yang telah diteliti dan dianalisis berdasarkan nilai terbaik (paling minimum), rata-rata nilai terbaik, standar deviasi dari nilai terbaik, dan rata-rata waktu komputasi yang akan dibandingkan dengan FWA dan algoritma GWO.

3.5 Kesimpulan

Penarikan kesimpulan metode terbaik antara FWA, GWO, dan hibrid FWA-GWO.

Diagram alir tahapan penelitian dapat dilihat pada Gambar 3.7.



Gambar 3.7. Diagram alir penelitian

3.6 Evaluasi Kinerja Algoritma

Hasil penelitian berupa perbandingan performa hibrid FWA-GWO, FWA, dan algoritma GWO. Fungsi uji diteliti dan dianalisis berdasarkan nilai terbaik (paling minimum), rata-rata nilai terbaik, standar deviasi, dan rata-rata waktu komputasi.

BAB IV
HASIL DAN PEMBAHASAN

Hibrid FWA-GWO merupakan gabungan dua algoritma metaheuristik, yaitu FWA dan GWO. Pada bab ini akan dibahas algoritma hibrid FWA-GWO, perbandingan nilai terbaik (paling minimum), rata-rata nilai terbaik, standar deviasi dari nilai minimum, dan waktu komputasi dari enam fungsi objektif dengan menggunakan tiga algoritma, yaitu FWA, GWO, dan hibrid FWA-GWO. Kriteria tersebut dapat digunakan untuk mengukur kinerja dari masing-masing algoritma.

4.1. Implementasi Algoritma

Pengaplikasian algoritma FWA-GWO untuk masalah optimasi dengan menggunakan enam fungsi uji dibuat dengan MATLAB R2014b dan menggunakan sistem komputer dengan spesifikasi *processor* Intel(R) Core(TM) i3-7020U, CPU 2.30GHz, RAM 4GB, serta hard disk berukuran 500GB. Rata-rata dan standar deviasi dari nilai minimum fungsi, waktu komputasi, dan nilai terbaik (paling minimum) algoritma FWA-GWO dibandingkan dengan algoritma FWA dan GWO.

4.1.1. Algoritma Hibrid FWA-GWO

Pada tahap inisialisasi, hibrid FWA-GWO menggunakan inisialisasi dari FWA, di mana akan dibangkitkan sejumlah *n packs*, *m wolf* untuk setiap *pack*, dan amplitudo maksimal $\hat{A}$. Posisi *pack* akan dibangkitkan secara acak, sedangkan posisi *wolf* akan dibangkitkan secara acak berdasarkan posisi setiap *pack*. Pengujian dilakukan dengan tiga versi jumlah *pack* dan *wolf* yang berbeda sesuai dengan Tabel 3.1. Algoritma akan berhenti ketika mencapai iterasi maksimal atau pada keadaan yang telah konvergen. Keadaan konvergen dalam skripsi ini merupakan keadaan ketika nilai minimum yang didapatkan tidak berubah dalam 200 iterasi berturut-turut.

Pada hibrid FWA-GWO menggunakan hierarki kepemimpinan pada GWO, sehingga setiap *pack* nya memiliki serigala $\alpha, \beta,$ dan δ. Untuk menentukan posisi selanjutnya digunakan perhitungan dari algoritma GWO dengan beberapa modifikasi, yaitu pada Persamaan

(2.16) ditambahkan ledakan amplitudo FWA pada Persamaan (2.4), persamaan tersebut dapat dituliskan sebagai berikut,

$$\vec{C}_n = \hat{A}_n \circ \vec{r}_{2n}, \tag{4.1}$$

$$\hat{A}_n = \frac{2.A_n}{\max(A_n)}, \tag{4.2}$$

$$A_n = \hat{A}\frac{y_{max} - f(\vec{x}_i) + \varepsilon}{\sum_{i=1}^{n}\left(y_{max} - f(\vec{x}_i)\right) + \varepsilon}, \tag{4.3}$$

di mana A_n merupakan ledakan amplitudo setiap *pack*, $\overrightarrow{r_{2n}}$ merupakan bilangan acak antara 0 dan 1, $\hat{A}$ merupakan amplitudo maksimal, dan $\vec{C}_n$ merupakan vektor koefisien $\vec{C}$ pada GWO yang telah dimodifikasi.

Selain itu, Persamaan (2.14) pada algoritma GWO, yaitu vektor koefisien A akan dimodifikasi menjadi vektor koefisien $\vec{E}_n$ dengan persamaan sebagai berikut,

$$\vec{E}_n = 2\vec{a} \circ \vec{r}_{1n}, \tag{4.4}$$

$$\vec{a} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}, \tag{4.5}$$

di mana $b_1 = b_2 = \cdots = b_n = b$, $b = 2 - \frac{2t}{t_{max}}$. Sehingga untuk menghitung posisi selanjutnya setiap *wolf* pada masing-masing *pack* dapat didefinisikan sebagai berikut.

$$\begin{aligned} \overrightarrow{D_{\alpha n}} &= \left|\overrightarrow{C_{1n}} \circ \overrightarrow{X_{\alpha n}} - \overrightarrow{X_n}\right|, \\ \overrightarrow{D_{\beta n}} &= \left|\overrightarrow{C_{2n}} \circ \overrightarrow{X_{\beta n}} - \overrightarrow{X_n}\right|, \\ \overrightarrow{D_{\delta n}} &= \left|\overrightarrow{C_{3n}} \circ \overrightarrow{X_{\delta n}} - \overrightarrow{X_n}\right|, \end{aligned} \tag{4.6}$$

$$\begin{aligned} \overrightarrow{X_{1n}} &= \overrightarrow{X_{\alpha n}} - \overrightarrow{E_{1n}} \circ \overrightarrow{D_{\alpha n}}, \\ \overrightarrow{X_{2n}} &= \overrightarrow{X_{\beta n}} - \overrightarrow{E_{2n}} \circ \overrightarrow{D_{\beta n}}, \\ \overrightarrow{X_{3n}} &= \overrightarrow{X_{\delta n}} - \overrightarrow{E_{3n}} \circ \overrightarrow{D_{\delta n}}. \end{aligned} \tag{4.7}$$

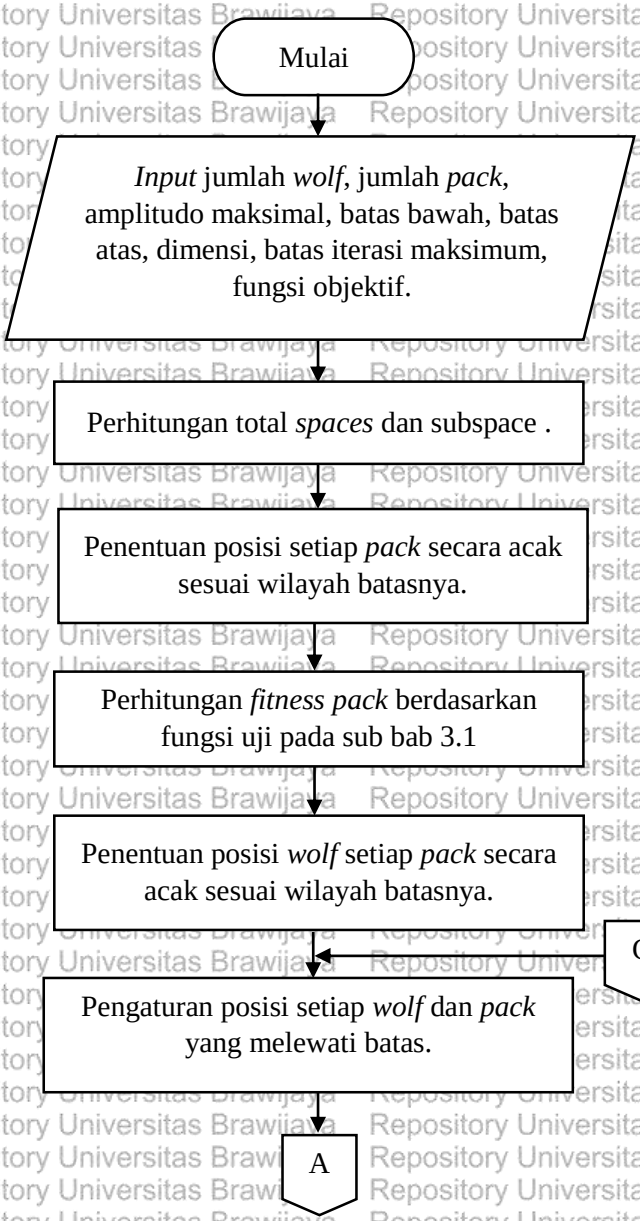
$$\vec{X}(t+1) = \frac{\overrightarrow{X_{1n}} + \overrightarrow{X_{2n}} + \overrightarrow{X_{3n}}}{3}. \tag{4.8}$$

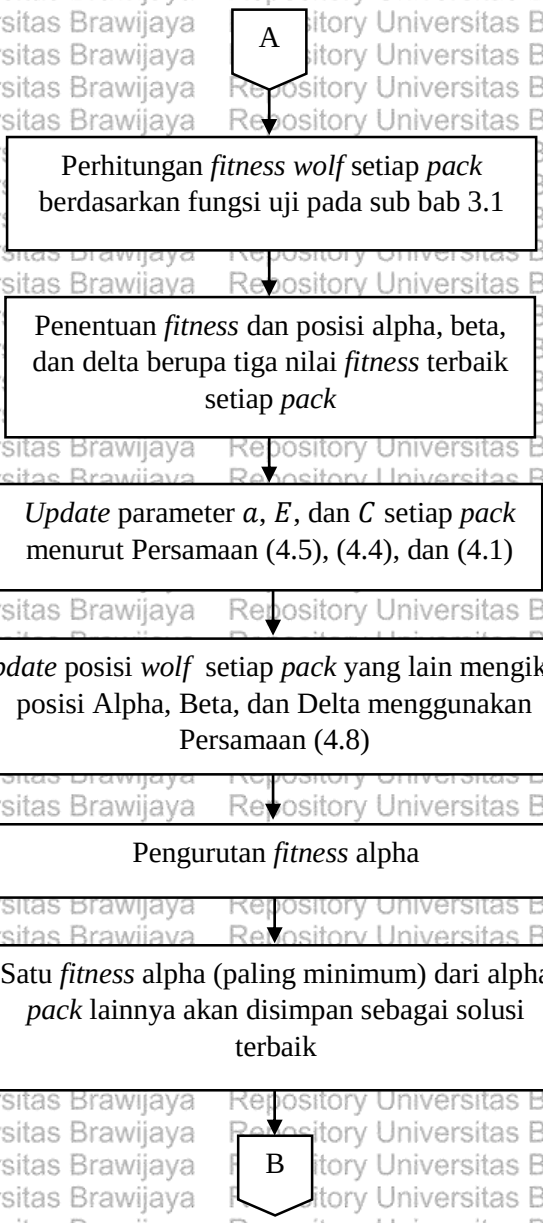
Vektor $\overrightarrow{D_{\alpha n}}$, $\overrightarrow{D_{\beta n}}$, dan $\overrightarrow{D_{\delta n}}$ pada Persamaan (4.6) menunjukkan jarak antara posisi serigala $\vec{X}$ dengan posisi terbaik milik serigala alpha $(\overrightarrow{X_{\alpha n}})$, beta $(\overrightarrow{X_{\beta n}})$, dan delta $(\overrightarrow{X_{\delta n}})$ setiap *pack*. Seluruh *wolf* akan diperbarui posisinya dengan Persamaan (4.8). Solusi dari serigala alpha setiap *pack* akan dipilih solusi yang terbaik (paling minimum) yang dianggap sebagai solusi terbaik dan dicetak pada akhir iterasi.

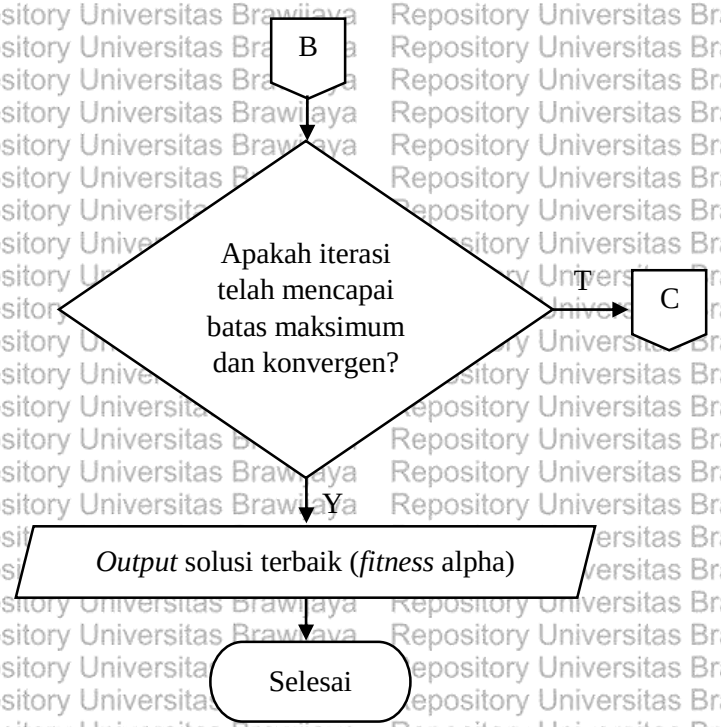
Apabila disusun dalam bentuk *pseudocode* dan *flowchart*, langkah-langkah dari algoritma hibrid FWA-GWO dapat dilihat pada Algoritma 4.1 dan Gambar 4.1.

Algoritma 4.1 *Pseudocode* Hibrid FWA-GWO

Input : $MaxIter, n, m, \hat{A}, fobj, ub, lb, dim$
Inisialisasi posisi $wolf$ X_{in} $(i = 1,2, \dots, m)$;
Hitung nilai *fitness* setiap $wolf$;
$fit_{i\alpha}$ = *wolf* terbaik pada *pack* ke-i;
$fit_{i\beta}$ = *wolf* terbaik kedua pada *pack* ke-i;
$fit_{i\delta}$ = *wolf* terbaik ketiga pada pack ke-i;
$Conv = 0$; $t = 1$;
while $t <$ $MaxIter$ && $Conv < 200$
    Hitung a berdasarkan Persamaan (4.5);
    for j ← 1 to n
        Hitung A_j berdasarkan Persamaan (4.3);
        r_1, r_2 ← random
        Hitung E_1, E_2, E_3 menggunakan Persamaan (4.4);
        Hitung C_1, C_2, C_3 menggunakan Persamaan (4.1);
        Hitung $D_{\alpha j}, D_{\beta j}, D_{\delta j}$ menggunakan Persamaan (4.6);
        Hitung $X_{\alpha j}, X_{\beta j}, X_{\delta j}$ menggunakan Persamaan (4.7);
        Perbarui posisi $wolf$ dengan Persamaan (4.8);
    end for
    $t \leftarrow t + 1$; $convergence_0 = 0$;
    $convergence_t \leftarrow fit_\alpha$;
    if $convergence_t \leftarrow convergence_{t-1}$
        $Conv \leftarrow Conv + 1$;
      else
        $Conv \leftarrow 0$;
    end if
end while
Output: fit_α;


Mulai
Input jumlah *wolf*, jumlah *pack*, amplitudo maksimal, batas bawah, batas atas, dimensi, batas iterasi maksimum, fungsi objektif.
Perhitungan total *spaces* dan subspace .
Penentuan posisi setiap *pack* secara acak sesuai wilayah batasnya.
Perhitungan *fitness pack* berdasarkan fungsi uji pada sub bab 3.1
Penentuan posisi *wolf* setiap *pack* secara acak sesuai wilayah batasnya.
C
Pengaturan posisi setiap *wolf* dan *pack* yang melewati batas.
A


A
Perhitungan *fitness wolf* setiap *pack* berdasarkan fungsi uji pada sub bab 3.1
Penentuan *fitness* dan posisi alpha, beta, dan delta berupa tiga nilai *fitness* terbaik setiap *pack*
Update parameter a, E, dan C setiap *pack* menurut Persamaan (4.5), (4.4), dan (4.1)
Update posisi *wolf* setiap *pack* yang lain mengikuti posisi Alpha, Beta, dan Delta menggunakan Persamaan (4.8)
Pengurutan *fitness* alpha
Satu *fitness* alpha (paling minimum) dari alpha *pack* lainnya akan disimpan sebagai solusi terbaik
B



Gambar 4.1. Diagram alir (*flowchart*) hibrid FWA-GWO

4.1.2. Perhitungan manual dengan algoritma hibrid FWA-GWO

Pada subbab ini, akan diselesaikan secara manual fungsi *Sphere* berdimensi dua dengan persamaan $f(x, y) = x^2 + y^2$ menggunakan algoritma hibrid FWA-GWO.

Langkah-langkah yang dilakukan untuk mengimplementasikan algoritma hibrid FWA-GWO adalah sebagai berikut.

1. Inisialisasi nilai parameter
   Parameter algoritma yang ditentukan antara lain jumlah *pack*, $n = 2$, jumlah *wolf* setiap *pack*, $m = 6$, dengan batas bawah, $lb = -10$, batas atas, $ub = 10$, amplitudo maksimal, $\hat{A} = 3$, dan maksimum iterasi, $MaxIter = 100$.
2. Perhitungan batas bawah dan batas atas setiap *pack*
   Setiap *pack* akan memiliki ruang pencarian (SS) yang berbeda dengan membagi total ruang pencarian (TS) dengan jumlah *pack*,

di mana batas atas dan batas bawah setiap *pack* juga sebagai ruang pencarian dari *wolf* tersebut.

$$TS = ub - lb = 10 - (-10) = 20,$$
$$SS = \frac{TS}{n} = \frac{20}{2} = 10,$$

sehingga akan didapatkan ruang pencarian setiap pack,

$$PackLB(1) = -10,$$
$$PackUB(1) = 0,$$
$$PackLB(2) = 0,$$
$$PackUB(2) = 10,$$

3. Inisialisasi posisi *pack*

Posisi *pack* akan dibangkitkan secara acak. Untuk $i = 1, \dots, n$.

$$posisiPack = \big(rand(n, \dim)\big) \times \big(PackUB(i) - PackLB(i)\big) + (PackLB(i)),$$
$$posisiPack = \begin{pmatrix} -0.4042 & -9.8069 \\ 7.2514 & 8.2623 \end{pmatrix}.$$

4. Inisialisasi posisi *wolf*

Selanjutnya posisi *wolf* akan dibangkitkan secara acak berdasarkan posisi setiap *pack*. Untuk $i = 1, \dots, n$.

$$pack(i).posisiwolf = \big(rand(wolf, \dim)\big) \times \big(PackUB(i) - PackLB(i)\big) + (PackLB(i)),$$
$$pack(1).posisiwolf = \begin{pmatrix} -4.6494 & -8.3458 \\ -6.9164 & -7.5933 \\ -1.4544 & -9.2484 \\ -3.6764 & -9.0246 \\ -7.2105 & -4.5312 \\ -0.4249 & -0.3511 \end{pmatrix},$$
$$pack(2).posisiwolf = \begin{pmatrix} 9.0246 & 0.2350 \\ 4.1828 & 0.5427 \\ 8.2777 & 6.8324 \\ 1.5761 & 9.7059 \\ 9.5717 & 4.8538 \\ 8.0028 & 1.4189 \end{pmatrix}.$$

5. Pengaturan batas domain
   Selanjutnya dilakukan pengecekan posisi *pack* dan *wolf* apakah posisi random yang dibangkitkan tidak keluar dari domain. Jika tidak maka posisi tetap dipakai, jika keluar domain maka posisi dibangkitkan ulang.
6. Perhitungan *fitness pack* dan *wolf*
   Fitness dihitung dengan menginputkan posisi *pack* dan *wolf* ke dalam fungsi Sphere, misalkan

$$posisiPack(1) = (x, y) = (-0.4042, -9.8069),$$

   maka fitnessnya adalah

$$f(x, y) = (-0.4042)^2 + (-9.8069)^2 = 96.3396.$$

   Fitness dari keseluruhan posisi *pack* dan *wolf* menghasilkan nilai seperti berikut.

$$fitPack = \begin{pmatrix} 96.3396 \\ 120.8498 \end{pmatrix},$$

$$Pack(1).fitWolf = \begin{pmatrix} 91.2693 \\ 105.4948 \\ \mathbf{87.6484} \\ 94.9593 \\ \mathbf{72.5231} \\ \mathbf{0.3038} \end{pmatrix},$$

$$Pack(2).fitWolf = \begin{pmatrix} \mathbf{81.4986} \\ \mathbf{17.7902} \\ 115.2019 \\ 96.6886 \\ 115.1768 \\ \mathbf{66.0581} \end{pmatrix},$$

   yang bercetak tebal merupakan nilai *fitness* terbaik (paling minimum).

7. Penentuan nilai *fitness* alpha, beta, dan delta setiap *pack*
   Nilai *fitness* dari serigala alpha, beta, dan delta setiap *pack* merupakan tiga nilai terbaik (paling minimum).

$$alphascore = \begin{pmatrix} 0.3038 \\ 17.7902 \end{pmatrix},$$

$$betascore = \begin{pmatrix} 72.5231 \\ 66.0581 \end{pmatrix},$$

$$deltascore = \begin{pmatrix} 87.6484 \\ 81.4986 \end{pmatrix},$$

$$alphapos = \begin{pmatrix} -0.4249 & -0.3511 \\ 4.1828 & 0.5427 \end{pmatrix},$$

$$betapos = \begin{pmatrix} -7.2105 & -4.5312 \\ 8.0028 & 1.4189 \end{pmatrix},$$

$$deltapos = \begin{pmatrix} -1.4544 & -9.2484 \\ 9.0246 & 0.2350 \end{pmatrix}.$$

8. Perhitungan amplitudo setiap *pack* dan parameter $\hat{A}_n$

Selanjutnya akan dilakukan perhitungan amplitudo setiap *pack* menggunakan Persamaan (4.3) dan dimislkan $\varepsilon = 0.0001$.

- Untuk *pack* 1

$$A(1) = 3 \times \frac{120.8498 - 96.3396 + 0.0001}{(120.8498 - 96.3396) + (120.8498 - 120.8498) + 0.0001},$$

$$A(1) = 3.0069.$$

- Untuk *pack* 2

$$A(2) = 3 \times \frac{120.8498 - 120.8498 + 0.0001}{(120.8498 - 96.3396) + (120.8498 - 120.8498) + 0.0001},$$

$$A(2) = 0.00001.$$

Setelah mendapatkan amplitudo setiap *pack*, akan dicari nilai $\hat{A}_n = pack(n).\hat{A}$ menggunakan Persamaan (4.2)

$$pack1.\hat{A} = \frac{2 \times 3}{3} = 2,$$

$$pack2.\hat{A} = \frac{2 \times 0.00001}{3} = 0.00001.$$

9. Perhitungan parameter a

Parameter a berkurang nilainya seiring berjalannya iterasi. Di dalam parameter ini terdapat variabel *loop* yang pada inisialisasi awal bernilai 0.

$$a = 2 - loop\left(\frac{2}{MaxIter}\right) = 2 - 0\left(\frac{2}{100}\right) = 2.$$

10. Pembaruan posisi semua *wolf* setiap *pack*

Perulangan akan dilakukan sebanyak jumlah *wolf*

- Untuk $wolf = 1$

Pada tahap ini akan dilakukan perhitungan untuk pembaruan posisi dengan mengikuti posisi dari serigala alpha.

$r1 = 0.6550,$
$r2 = 0.0260,$

$C1 = pack1.\hat{A} \times r2 = 2 \times 0.0260 = 0.0520,$
$E1 = 2 \times a \times r1 = 2 \times 2 \times 0.6550 = 2.6200,$

$\vec{D}alpha = |C1 \times alphapos(1) - posisiwolf(1)|,$
$\vec{D}alpha = \left|0.0520\begin{pmatrix}-0.4249\\-0.3511\end{pmatrix} - \begin{pmatrix}-4.6494\\-8.3458\end{pmatrix}\right|,$
$\vec{D}alpha = \begin{pmatrix}0.2197\\0.4469\end{pmatrix},$
$\vec{X}1 = alphapos(1) - E1 \times Dalpha,$
$\vec{X}1 = \begin{pmatrix}-0.4249\\-0.3511\end{pmatrix} - 2.6200\begin{pmatrix}0.2197\\0.4469\end{pmatrix},$
$\vec{X}1 = \begin{pmatrix}-1.0005\\-1.5219\end{pmatrix}.$

Selanjutnya akan dilakukan perhitungan untuk pembaruan posisi dengan mengikuti posisi dari serigala beta.

$r1 = 0.3734,$
$r2 = 0.1444,$

$C2 = pack1.\hat{A} \times r2 = 2 \times 0.1444 = 0.2888,$
$E2 = 2 \times a \times r1 = 2 \times 2 \times 0.3734 = 1.4930,$

$\vec{D}beta = |C2 \times betapos(1) - posisiwolf(1)|,$
$\vec{D}beta = \left|0.2888\begin{pmatrix}-7.2105\\-4.5312\end{pmatrix} - \begin{pmatrix}-4.6494\\-8.3458\end{pmatrix}\right|,$

$$\vec{D}beta = \begin{pmatrix} 2.5670 \\ 7.0372 \end{pmatrix},$$
$$\vec{X}2 = betapos(1) - E2 \times \vec{D}beta,$$
$$\vec{X}2 = \begin{pmatrix} -7.2105 \\ -4.5312 \end{pmatrix} - 1.4930 \begin{pmatrix} 2.5670 \\ 7.0372 \end{pmatrix},$$
$$\vec{X}2 = \begin{pmatrix} -11.0430 \\ -15.0377 \end{pmatrix}.$$

Selanjutnya akan dilakukan perhitungan untuk pembaruan posisi dengan mengikuti posisi dari serigala delta.

$$r1 = 0.0281,$$
$$r2 = 0.8398,$$
$$C3 = pack1.\hat{A} \times r2 = 2 \times 0.8398 = 1.6796,$$
$$E3 = 2 \times a \times r1 = 2 \times 2 \times 0.0281 = 0.1124,$$
$$\vec{D}delta = |C3 \times deltapos(1) - posisiwolf(1)|,$$
$$\vec{D}delta = \left| 1.6796 \begin{pmatrix} -1.4544 \\ -9.2484 \end{pmatrix} - \begin{pmatrix} -4.6494 \\ -8.3458 \end{pmatrix} \right|,$$
$$\vec{D}delta = \begin{pmatrix} 2.2066 \\ -7.1878 \end{pmatrix},$$
$$\vec{X}3 = deltapos(1) - E2 \times \vec{D}beta,$$
$$\vec{X}3 = \begin{pmatrix} -1.4544 \\ -9.2484 \end{pmatrix} - 0.1124 \begin{pmatrix} 2.2066 \\ -7.1878 \end{pmatrix},$$
$$\vec{X}3 = \begin{pmatrix} -1.7024 \\ -8.4405 \end{pmatrix}.$$

Setelah didapatkan penyesuaian posisi alpha, beta, dan delta, selanjutnya posisi akan diperbarui menggunakan Persamaan (4.8).

$$pack1.posisiwolf(1) = \frac{\vec{X}1 + \vec{X}2 + \vec{X}3}{3},$$
$$pack1.posisiwolf(1) = \frac{\begin{pmatrix} -1.0005 \\ -1.5219 \end{pmatrix} + \begin{pmatrix} -11.0430 \\ -15.0377 \end{pmatrix} + \begin{pmatrix} -1.7024 \\ -8.4405 \end{pmatrix}}{3},$$
$$pack1.posisiwolf(1) = (-4.5819 \quad -8.3334).$$

Dari perhitungan di atas, didapatkan posisi selanjutnya untuk *wolf* pertama pada *pack* pertama, perhitungan tersebut di ulang sampai sejumlah *wolf* disetiap *pack*-nya.

Selanjutnya jika posisi terbaru seluruh *wolf* setiap *pack* didapatkan, posisi *wolf* terbaru akan dihitung nilai fitnessnya dengan mengulangi langkah kelima.

11. Kriteria pemberhentian

Proses akan berhenti ketika mencapai batas iterasi maksimum (*MaxIter*) yang telah ditentukan, yaitu 100 iterasi.

4.2. Hasil dan Analisis Keluaran Program

Dengan jumlah *pack*, *wolf*, dan iterasi maksimum yang telah ditentukan sebelumnya, program hibrid FWA-GWO mengeksekusi fungsi uji dalam tiga puluh kali percobaan. Hal ini bertujuan untuk menentukan rata-rata nilai minimum dari setiap fungsi uji. Rata-rata nilai minimum diperoleh berdasarkan nilai variabel yang dimasukkan ke dalam setiap fungsi uji. Berikut ini adalah hasil uji minimasi menggunakan tiga metode dengan versi yang sama setiap metode, yaitu versi 1, versi 2, dan versi 3 dengan setiap versi memiliki jumlah *pack*, *wolf*, dan iterasi maksimum yang berbeda, dapat disajikan dalam bentuk tabel dan perbandingan ketiga metode setiap versinya.

1. Fungsi *Sphere*

Berdasarkan Tabel 4.1 dapat dilihat bahwa rata-rata nilai minimum yang dihasilkan oleh hibrid FWA-GWO dalam meminimalkan salah satu fungsi unimodal dan *differentiable*, yaitu fungsi *Sphere* lebih baik dibandingkan hasil yang diperoleh menggunakan FWA dan GWO. Semakin kecil dimensi yang digunakan, rata-rata nilai minimum yang dihasilkan oleh ketiga algoritma akan semakin baik.

Apabila dianalisis berdasarkan standar deviasinya, pengujian fungsi *Sphere* dengan ketiga algoritma menghasilkan nilai yang mendekati seragam dengan rata-rata nilai minimum untuk 30 kali perulangan. Hibrid FWA-GWO menghasilkan standar deviasi yang lebih besar daripada rata-rata nilai minimum, sehingga rata-rata nilai minimum yang diperoleh dapat digunakan sebagai representasi dari keseluruhan data. Sementara FWA menunjukkan representasi yang

buruk dalam hal penyebaran data untuk 60 dan 90 dimensi. Hal ini dikarenakan standar deviasi FWA cukup besar (lebih dari 0).

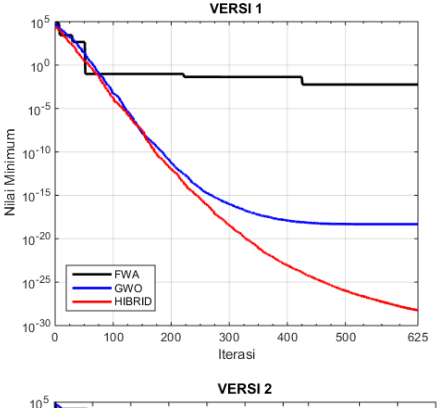
Apabila dilihat dari rata-rata waktu komputasi, GWO memiliki waktu komputasi yang lebih cepat apabila dibandingkan dengan FWA dan hibrid FWA-GWO, hal ini dikarenakan tahapan yang dilakukan untuk satu iterasi pada GWO lebih sederhana dibandingkan dengan FWA dan hibrid FWA-GWO. Apabila dilihat berdasarkan versinya, Versi 3 memiliki waktu komputasi lebih lama. Hal ini menunjukkan bahwa apabila jumlah populasi ditambah, maka waktu komputasi menjadi semakin lama.

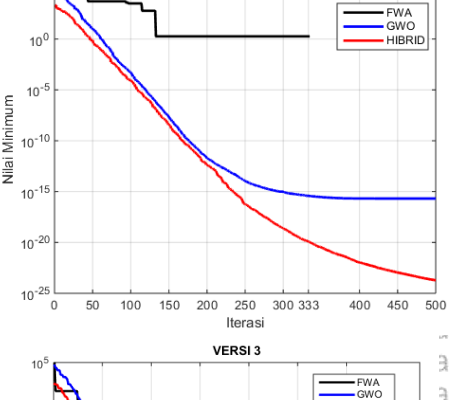
Berdasarkan Gambar 4.2 dapat dilihat bahwa grafik solusi dari ketiga algoritma berdimensi 30 menghasilkan hasil yang berbeda-beda setiap versinya. Setiap versi memiliki jumlah populasi yang berbeda yang dapat dilihat pada Tabel 3.1, semakin kecil versinya maka semakin besar juga jumlah populasinya dan semakin banyaknya perulangan atau iterasi maka semakin baik solusi yang dihasilkan. Pada Versi 1 Hibrid FWA-GWO dan GWO banyak mengalami penurunan seiring berjalannya iterasi hingga iterasi ke-625. Pada Versi 3 sebelum mencapai iterasi 100 GWO mengalami solusi yang lebih baik dibandingkan hibrid FWA-GWO, setelah iterasi 300 solusi hibrid FWA-GWO mengalami solusi yang lebih baik dibandingkan GWO sampai iterasi terakhir, yaitu iterasi 375. Secara umum solusi yang dihasilkan hibrid FWA-GWO lebih baik dibandingkan hasil yang diperoleh menggunakan FWA dan GWO meskipun belum mencapai nilai minimum global.

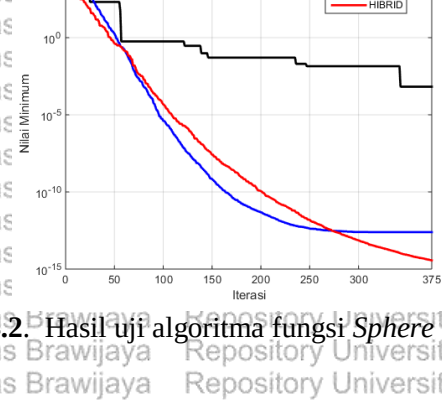
Tabel 4.1 Hasil uji algoritma pada fungsi *Sphere*

		Rata-rata Nilai Minimum	Standar Deviasi	Nilai terbaik	Rata-rata Waktu Komputasi (detik)
30 DIMENSI					
V1	FWA	2.1460E-01	6.1322E-01	1.0261E-06	5.8482
	GWO	9.4412E-19	1.9906E-18	1.7441E-20	1.7472
	HIBRID	9.7619E-31	1.5608E-30	5.2186E-33	7.3960
V2	FWA	1.1279E-01	3.5084E-01	5.0244E-08	8.5732
	GWO	6.8806E-16	1.8475E-15	4.5014E-18	1.7406
	HIBRID	4.5765E-23	8.3828E-23	1.7808E-25	7.6217
V3	FWA	5.0393E-03	1.1830E-02	1.1619E-06	10.455
	GWO	3.1229E-13	5.3758E-13	1.8519E-14	1.7276
	HIBRID	5.6896E-15	6.1219E-15	1.0940E-16	9.8085
60 DIMENSI					
V1	FWA	3.8967E+03	1.3186E+04	1.5013E-05	5.1508
	GWO	1.4136E-08	1.7557E-08	1.5686E-09	3.3504
	HIBRID	4.1323E-23	5.1921E-23	9.8753E-25	12.6660
V2	FWA	1.6112E+03	7.0997E+03	1.2657E-03	5.5257
	GWO	3.0169E-07	3.0169E-07	2.8200E-08	3.3462
	HIBRID	3.1346E-17	4.4999E-17	1.9949E-18	16.9781
V3	FWA	2.4577E-01	5.0609E-01	5.8474E-07	7.3778
	GWO	1.9975E-05	1.7084E-05	2.6257E-06	3.3948
	HIBRID	6.4214E-11	4.5363E-11	4.0227E-12	17.4383
90 DIMENSI					
V1	FWA	1.8503E+04	5.4148E+04	1.6519E-02	6.6439
	GWO	1.3624E-04	9.8397E-05	1.0535E-05	4.9702
	HIBRID	8.9850E-20	1.6995E-19	3.8849E-22	20.3171
V2	FWA	6.7996E+03	1.7511E+04	1.6519E-02	6.6439
	GWO	1.3035E-03	7.7683E-04	2.1458E-04	4.9692
	HIBRID	9.8348E-15	8.3346E-15	5.2662E-16	20.2127
V3	FWA	1.9198E+02	1.0047E+03	9.7526E-04	8.9493
	GWO	2.5320E-02	2.2703E-02	4.2108E-03	4.9423
	HIBRID	5.5294E-09	4.7631E-09	4.6151E-10	19.4070

: Nilai paling minimum







Gambar 4.2. Hasil uji algoritma fungsi *Sphere* dalam grafik

2. Fungsi *Schwefel*

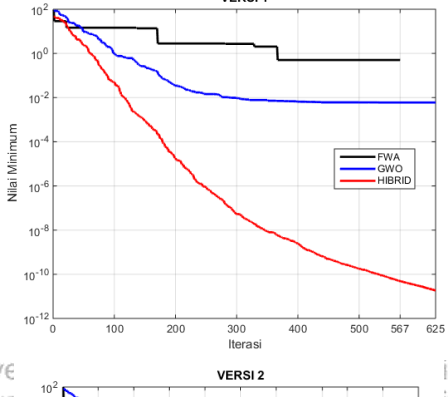
Pengujian selanjutnya yaitu untuk meminumkan fungsi unimodal dan tidak *differentiable*, yaitu fungsi *Schwefel* dengan tiga algoritma. Seperti yang dapat terlihat pada Tabel 4.2, hibrid FWA-GWO memiliki rata-rata nilai minimum yang lebih baik apabila dibandingkan dengan FWA dan GWO untuk pengujian menggunakan dimensi 30 dan 60.

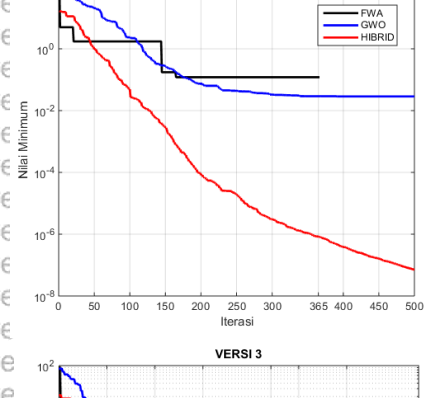
Jika dianalisis berdasarkan standar deviasi untuk pengujian menggunakan dimensi 30 dan 60 dengan ketiga algoritma menghasilkan nilai yang mendekati seragam dengan rata-rata nilai minimum untuk 30 kali perulangan. Hibrid FWA-GWO menghasilkan standar deviasi yang lebih besar daripada rata-rata nilai minimum, sehingga rata-rata nilai minimum yang diperoleh dapat digunakan sebagai representasi dari keseluruhan data. Apabila menggunakan dimensi 90 pada Versi 3 rata-rata nilai minimum dan nilai terbaik FWA lebih unggul dibandingkan GWO dan hibrid FWA-GWO. Akan tetapi, standar deviasi hibrid FWA-GWO lebih unggul dibandingkan kedua algoritma lainnya, hal ini menunjukkan bahwa meskipun belum mampu mendapatkan hasil yang optimal namun kinerjanya konstan untuk 30 kali perulangan. Apabila dilihat dari waktu komputasi algoritma yang memiliki waktu komputasi yang lebih cepat adalah algoritma GWO.

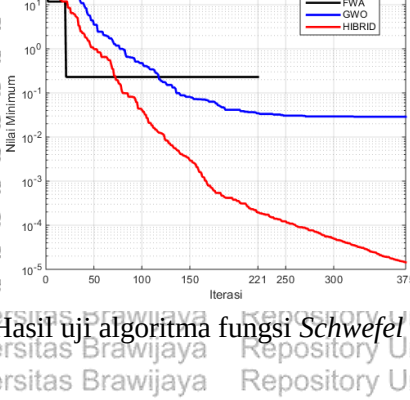
Grafik solusi fungsi *Schwefel* dari ketiga algortima dapat dilihat pada Gambar 4.3. Hibrid FWA-GWO menghasilkan solusi yang lebih baik dibandingkan kedua algoritma pembandingnya meskipun belum mencapai nilai minimum global. Sedangkan untuk FWA pada Versi 1, Versi 2, dan Versi 3 konvergen sebelum mencapai iterasi maksimal, namun belum mencapai nilai minimum global.

Tabel 4.2 Hasil uji algoritma pada fungsi *Schwefel*

		Rata-rata Nilai Minimum	Standar Deviasi	Nilai Terbaik	Rata-rata Waktu Komputasi (detik)
30 DIMENSI					
V1	FWA	7.0477E-01	1.3173	5.9575E-05	6.3414
	GWO	1.2366E-02	1.0804E-02	5.9010E-04	1.7898
	HIBRID	5.3865E-10	6.4043E-10	2.1930E-11	9.5287
V2	FWA	3.0250E-01	1.1147	2.5086E-04	7.7501
	GWO	2.4984E-02	2.1399E-02	3.8612E-03	2.1963
	HIBRID	1.8775E-07	1.4600E-07	2.5288E-08	7.2572
V3	FWA	3.4452E-01	1.0984	6.1015E-04	14.0799
	GWO	6.1400E-02	4.0480E-02	1.1024E-02	1.8058
	HIBRID	4.3326E-05	3.6987E-05	4.1176E-06	12.0081
60 DIMENSI					
V1	FWA	8.4240	11.0265	4.1557E-03	7.62440
	GWO	12.6576	12.5009	4.0092	4.1672
	HIBRID	1.0479E-04	3.2013E-04	1.2439E-07	12.9941
V2	FWA	2.7003	5.5131	1.5055E-03	9.5861
	GWO	10.0401	7.6455	1.8638	4.2082
	HIBRID	2.2149E-04	3.1540E-03	1.3353E-05	16.1411
V3	FWA	1.7020	3.8942	2.5102E-03	10.9657
	GWO	11.5613	5.5131	2.3785	5.5604
	HIBRID	7.8199E-02	7.5947E-01	1.2685E-03	13.7427
90 DIMENSI					
V1	FWA	4.3591	6.5634	2.8850E-02	9.1600
	GWO	55.1879	14.8206	22.2243	6.6914
	HIBRID	1.4172E-02	2.3359E-02	9.8631E-06	18.3267
V2	FWA	4.2424	6.4669	4.6924E-03	11.3452
	GWO	50.6526	11.4171	25.7429	6.8041
	HIBRID	1.6476E-01	4.4030E-01	2.0106E-04	19.6933
V3	FWA	0.6089	1.7747	2.2747E-04	15.5075
	GWO	53.5611	15.2916	25.7716	5.7250
	HIBRID	1.6949	9.6814E-01	8.3951E-04	20.5566







Gambar 4.3. Hasil uji algoritma fungsi *Schwefel* dalam grafik

3. Fungsi *Rosenbrock*

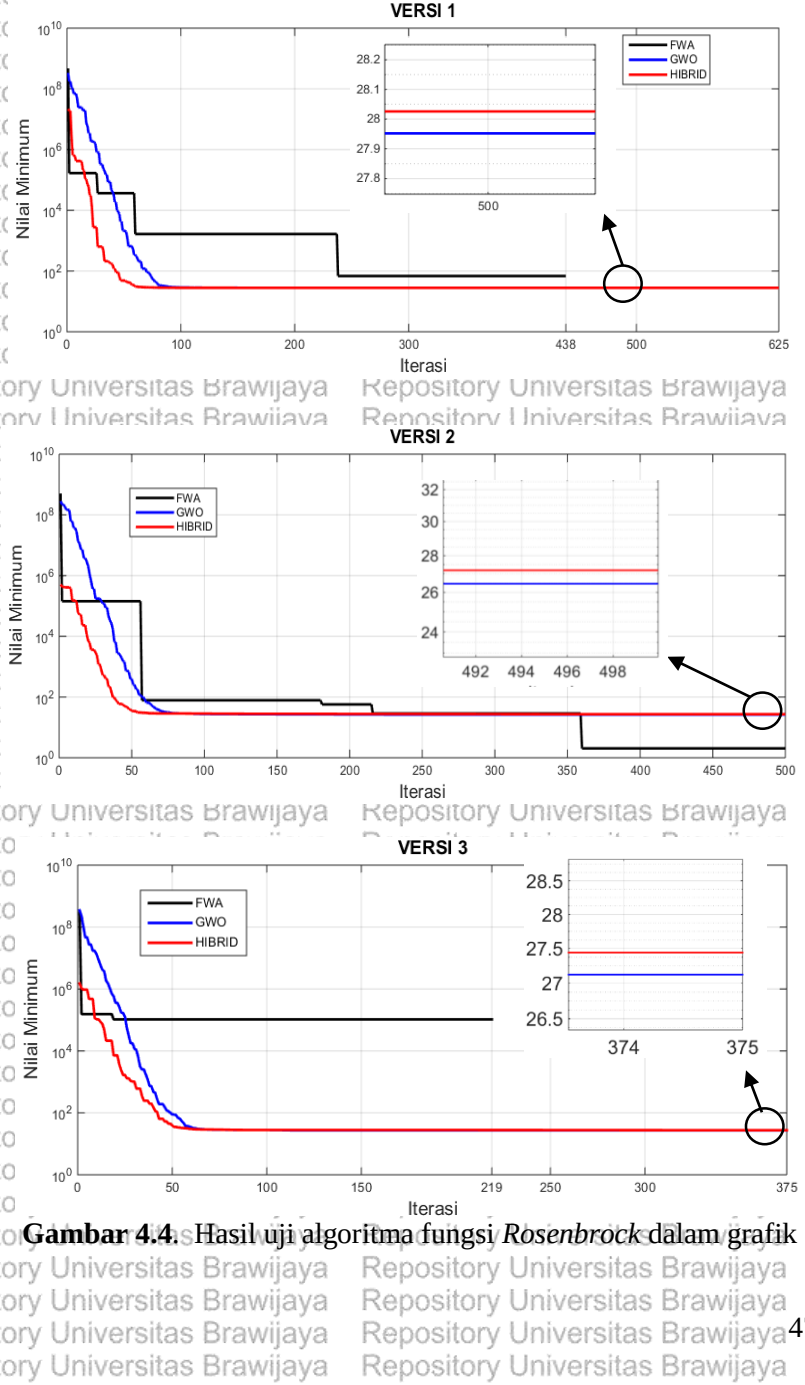
Selain diujikan pada fungsi unimodal, algoritma hibrid FWA-GWO juga diterapkan untuk menyelesaikan permasalahan fungsi multimodal. Apabila diperhatikan pada Tabel 4.3 dapat dilihat bahwa rata-rata nilai minimum yang didapatkan oleh ketiga algoritma belum mendekati solusi eksaknya. Saat menggunakan dimensi 30 rata-rata nilai minimum GWO lebih unggul dibandingkan FWA dan hibrid FWA-GWO. Untuk nilai terbaik FWA lebih unggul dibandingkan GWO dan hibrid FWA-GWO. Jika menggunakan dimensi 60 dan 90 rata-rata nilai minimum hibrid FWA-GWO lebih unggul dibandingkan FWA dan GWO. Hal ini menandakan bahwa hibrid FWA-GWO mampu meminimalkan fungsi *Rosenbrock* pada dimensi yang 60 dan 90 jika dibandingkan dengan FWA dan GWO.

Apabila dianalisis berdasarkan standar deviasi, hibrid FWA-GWO memiliki standar deviasi yang lebih rendah. Hal ini menunjukkan bahwa meskipun belum mampu mendapatkan hasil yang optimal namun kinerjanya konstan untuk 30 kali perulangan. Untuk waktu komputasi, algoritma GWO tetap lebih cepat dibandingkan FWA dan hibrid FWA-GWO.

Grafik solusi pada Gambar 4.4 menunjukkan hasil yang sama dengan Tabel 4.3 dalam meminimalkan fungsi *Rosenbrock* berdimensi 30. Pada Versi 1, Versi 2, dan Versi 3 GWO berbeda tipis dengan hibrid FWA-GWO tetapi tetap GWO lebih unggul dibanding kedua algoritma pembandingnya. Pada Versi 2 FWA lebih unggul dibanding GWO dan hibrid FWA-GWO, pada Versi 1 dan Versi 3 FWA konvergen sebelum mencapai iterasi maksimal meskipun belum mencapai nilai minimum global. Jarak antara posisi titik minimum global dan $\vec{x}^*$ yang dihasilkan menggunakan hibrid FWA-GWO adalah 5.373893. Hal ini mengakibatkan hibrid FWA-GWO memiliki kemungkinan untuk terjebak pada titik optimum lokal.

Tabel 4.3 Hasil uji algoritma pada fungsi *Rosenbrock*

		Rata-rata Nilai Minimum	Standar Deviasi	Nilai Terbaik	Rata-rata Waktu Komputasi (detik)
30 DIMENSI					
V1	FWA	5.7541E+04	1.65566E+06	5.4269	4.7009
	GWO	27.1746	0.7972	26.1241	2.6655
	HIBRID	28.1928	0.5172	26.8626	6.8549
V2	FWA	3.7291E+05	1.6439E+06	2.4231E-02	6.7400
	GWO	26.8710	0.8756	25.4437	2.7239
	HIBRID	28.3179	0.4683	27.2107	8.7978
V3	FWA	2.9283E+03	7.3726E+03	2.6138E-01	11.8009
	GWO	27.2203	1.0012	25.5113	3.2126
	HIBRID	28.4212	0.4011	27.2950	8.1821
60 DIMENSI					
V1	FWA	3.5410E+06	7.7518E+06	6.3245E-04	4.6653
	GWO	58.0573	0.6554	56.2674	5.3286
	HIBRID	57.9633	0.5763	57.0427	11.5991
V2	FWA	2.9498E+06	1.0141E+07	1.6430E-01	5.1663
	GWO	59.1687	3.9039	56.8115	5.3679
	HIBRID	57.8471	0.4798	56.9298	12.5357
V3	FWA	1.0150E+06	4.2437E+06	7.2865	7.2050
	GWO	58.49114	0.38934	57.2963	5.3048
	HIBRID	57.8345	0.3831	56.9497	12.5222
90 DIMENSI					
V1	FWA	3.4393E+07	1.1099E+08	57.4855	4.5845
	GWO	89.7898	4.1668	87.6275	5.7303
	HIBRID	88.6879	0.1881	88.0333	19.2372
V2	FWA	3.0065E+06	1.2001E+07	23.5723	4.5602
	GWO	93.7489	9.3663	88.2416	6.7887
	HIBRID	88.7247	0.1407	88.0879	26.7390
V3	FWA	2.0307E+06	8.0801E+06	9.7296E-01	24.0951
	GWO	119.9371	34.6955	91.1584	7.7881
	HIBRID	88.7691	0.1970	88.0821	28.2636



Gambar 4.4. Hasil uji algoritma fungsi *Rosenbrock* dalam grafik

4. Fungsi *Ackley*

Berdasarkan Tabel 4.4 dapat dilihat bahwa rata-rata nilai minimum yang dihasilkan oleh hibrid FWA-GWO dalam meminimalkan salah satu fungsi multimodal, yaitu fungsi *Ackley* lebih baik dibandingkan hasil yang diperoleh menggunakan FWA dan GWO. Rata-rata nilai minimum yang didapatkan hibrid FWA-GWO jika menggunakan dimensi 30 akan lebih baik dibandingkan menggunakan dimensi 60 dan 90. Jika dianalisis berdasarkan versinya, semakin kecil versinya maka rata-rata nilai minimum yang didapatkan akan semakin kecil atau semakin baik.

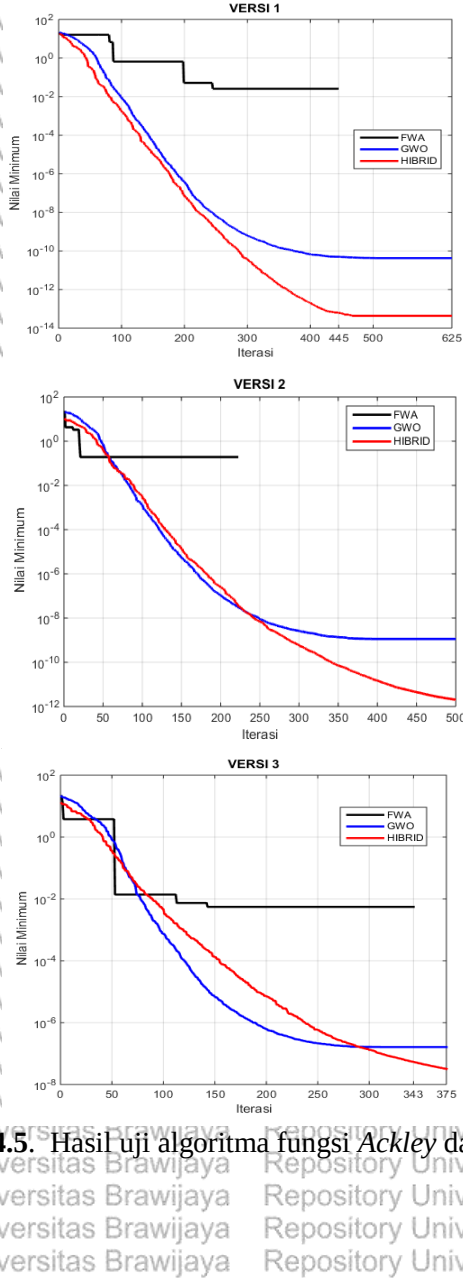
Hibrid FWA-GWO menghasilkan standar deviasi yang lebih baik dibandingkan kedua algoritma, hal ini menunjukkan rata-rata nilai minimum yang diperoleh dapat digunakan sebagai representasi dari keseluruhan data. Sementara FWA menunjukkan representasi yang buruk dalam hal penyebaran data untuk 90 dimensi. Hal ini dikarenakan standar deviasi FWA cukup besar.

Berdasarkan rata-rata waktu komputasi, GWO memiliki waktu komputasi yang lebih cepat apabila dibangkan dengan FWA dan hibrid FWA-GWO, hal ini dikarenakan tahapan yang dilakukan untuk satu iterasi pada GWO lebih sederhana dibandingkan dengan FWA dan hibrid FWA-GWO.

Secara umum solusi yang dihasilkan hibrid FWA-GWO pada Versi 1, Versi 2, dan Versi 3 lebih unggul dibandingkan FWA dan GWO meskipun belum mencapain nilai minimum global fungsi *Ackley* dapat dilihat pada Gambar 4.5. Pada Versi 3 sebelum mencapai iterasi 100 GWO mengalami solusi yang lebih baik, setelah mendekati iterasi terakhir, yaitu iterasi 375 solusi hibrid FWA-GWO mengalami solusi yang lebih baik. Sedangkan FWA pada Versi 1, Versi 2, dan Versi 3 konvergen sebelum mencapai iterai terakhir, akan tetapi solusi yang dihasilkan FWA belum mencapai nilai minimum global.

Tabel 4.4 Hasil uji algoritma pada fungsi *Ackley*

		Rata-rata Nilai Minimum	Standar Deviasi	Nilai Terbaik	Rata-rata Waktu Komputasi (detik)
30 DIMENSI					
V1	FWA	0.7655	1.8478	1.0505E-03	7.3794
	GWO	2.1417E-10	1.6599E-10	3.6821E-11	2.1997
	HIBRID	1.1919E-13	1.6707E-13	3.2863E-14	8.3055
V2	FWA	1.1646	3.3122	5.0432E-04	8.5958
	GWO	3.8020E-09	2.2493E-09	8.7722E-10	2.2372
	HIBRID	1.9454E-12	2.5318E-12	1.5010E-13	7.6444
V3	FWA	0.2701	7.3506E-01	1.5475E-04	15.8705
	GWO	9.8834E-08	4.9769E-08	2.2208E-08	2.0949
	HIBRID	4.0086E-08	7.5776E-08	1.3040E-09	9.8959
60 DIMENSI					
V1	FWA	2.4612	2.8539	1.6676E-03	6.3171
	GWO	1.7415E-05	1.1207E-05	1.9117E-06	4.3390
	HIBRID	1.9392E-10	7.9693E-10	5.9774E-13	13.8867
V2	FWA	2.3097	3.8608	3.8110E-03	7.9646
	GWO	7.9433E-05	3.8689E-05	2.8884E-05	4.6009
	HIBRID	2.1611E-10	1.7130E-10	2.7264E-11	12.8955
V3	FWA	0.6946	1.1430	2.8681E-03	11.5797
	GWO	5.0326E-04	2.1054E-04	1.9106E-04	4.2409
	HIBRID	6.5516E-07	1.0407E-06	5.8906E-08	11.8045
90 DIMENSI					
V1	FWA	5.9352	5.7774	2.8681E-03	6.3657
	GWO	1.1217E-03	4.3989E-04	5.0796E-04	7.6026
	HIBRID	3.9783E-10	9.7091E-10	9.8277E-12	20.2903
V2	FWA	4.1684	5.2005	3.6577E-02	12.0600
	GWO	3.6304E-03	1.2364E-03	1.7968E-03	8.0873
	HIBRID	2.4009E-09	1.8270E-09	4.4788E-10	26.0541
V3	FWA	2.1147	3.0427	8.4134E-03	9.9573
	GWO	1.6425E-02	5.9541E-03	6.6884E-03	7.2016
	HIBRID	2.3427E-05	7.7636E-05	2.9946E-07	25.5708



Gambar 4.5. Hasil uji algoritma fungsi *Ackley* dalam grafik

5. Fungsi *Camel Six-Humps*

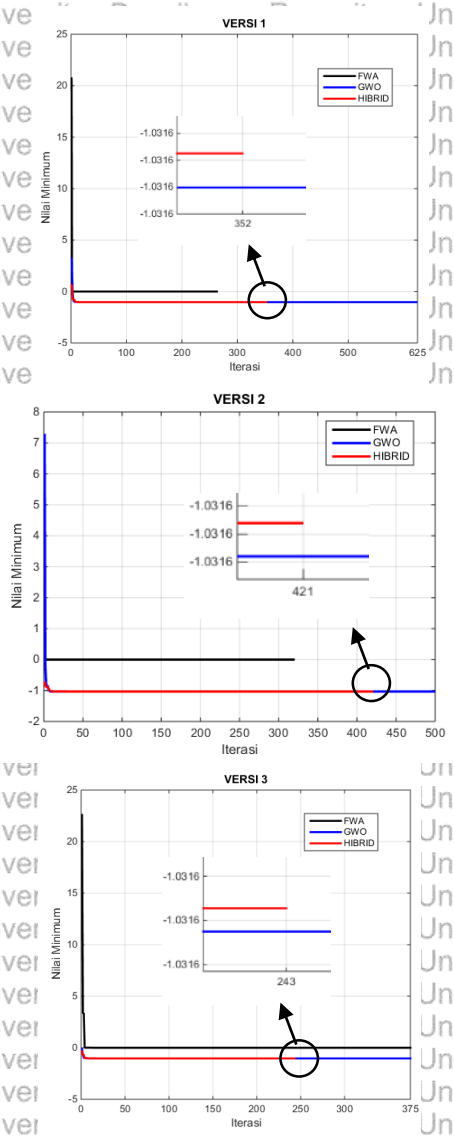
Fungsi *Camel Six-Humps* merupakan fungsi multimodal berdimensi tetap, yaitu berdimensi 2. Berdasarkan Tabel 4.5 menunjukkan bahwa algoritma GWO dan hibrid FWA-GWO mampu mencapai solusi eksak yaitu $f(\vec{x}) = -1.0316$. Akan tetapi, untuk waktu komputasi algoritma GWO lebih unggul dibandingkan FWA dan hibrid FWA-GWO.

Berdasarkan Gambar 4.6 dapat dilihat bahwa grafik solusi dari ketiga algoritma, pada Versi 1 dan Versi 2 FWA konvergen sebelum mencapai iterasi maksimal, tetapi solusi yang dihasilkan belum mencapai nilai minimum global. Pada Versi 1, Versi 2 dan Versi 3 nilai minimum yang didapatkan GWO dan hibrid FWA-GWO mendekati nilai minimum global. Pada hibrid FWA-GWO mencapai nilai terbaik di titik $(-0.0899, 0.7126)$, dimana titik minimum global fungsi *Camel Six-Humps* $(-0.0898, 0.7126)$. Hal ini menunjukkan bahwa hibrid FWA-GWO tidak terjebak pada minimum lokal.

Tabel 4.5 Hasil uji algoritma pada fungsi *Camel Six-Humps*

		Rata-rata Nilai Minimum	Standar Deviasi	Nilai Terbaik	Rata-rata Waktu Komputasi (detik)
V1	FWA	-0.0415	0.1384	-0.6594	3.9304
	GWO	-1.0316	8.8224E-06	-1.0316	0.2855
	HIBRID	-1.0316	1.0689E-05	-1.0316	1.1402
V2	FWA	-0.0038	0.0210	-0.1150	4.7912
	GWO	-1.0316	6.4807E-06	-1.0316	0.3085
	HIBRID	-1.0316	2.0934E-05	-1.0316	1.2820
V3	FWA	-0.1088	0.2541	-0.8293	7.5732
	GWO	-1.0316	2.0348E-06	-1.0316	0.3999
	HIBRID	-1.0316	3.2862E-06	-1.0316	2.8100

: Nilai paling minimum



Gambar 4.6. Hasil uji algoritma fungsi *Camel Six-Humps* dalam grafik

6. Fungsi *Goldstein Price*

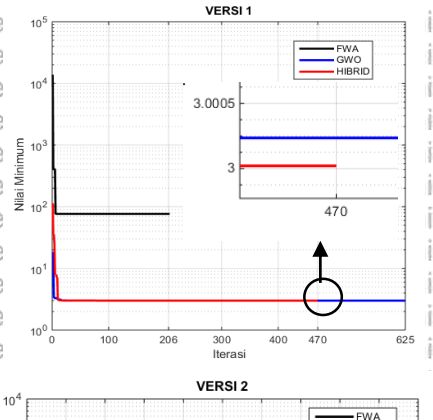
Fungsi *Goldstein Price* merupakan fungsi multimodal berdimensi tetap, yaitu berdimensi 2. Hasil percobaan dengan fungsi *Goldstein Price* dapat dilihat pada Tabel 4.6. Tabel tersebut menunjukkan bahwa algoritma GWO lebih unggul dibandingkan FWA dan hibrid FWA-GWO karena algoritma GWO memliki rata-rata nilai minimum yang lebih rendah dibandingkan kedua algoritma lainnya. Algoritma FWA sendiri belum mendekati solusi eksaknya dan tingginya standar deviasi yang dihasilkan. Kinerja algoritma hibrid FWA-GWO untuk versi 1 lebih besar dari hasil agortitma GWO sejumlah 0.0001237 dan waktu komputasi algoritma GWO lebih cepat 1.8283 detik.

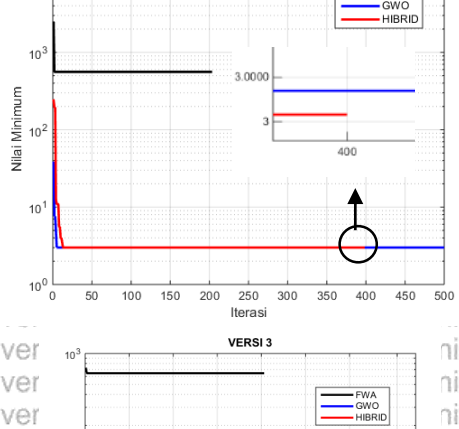
Walaupun dalam meminimalkan fungsi *Goldstein Price* rata-rata nilai minimum dan standar deviasi yang dihasilkan hibrid FWA-GWO lebih besar dibandingkan GWO, jika dianalisis berdasarkan nilai terbaik untuk ketiga versi, algoritma hibrid FWA-GWO mampu mencapai solusi eksak, yaitu $f(\vec{x}) = 3$.

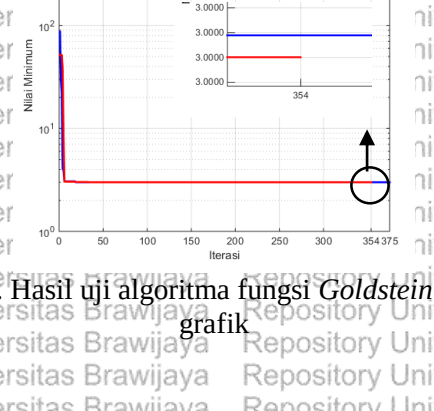
Pada Gambar 4.7 dapat dilihat grafik solusi dari ketiga algoritma. Pada ketiga versi FWA lebih dulu konvergen dibandingkan GWO dan hibrid FWA-GWO namun belum mencapai nilai minimum global. Sedangkan GWO dan hibrid FWA-GWO pada ketiga versi mencapai nilai minimum global, akan tetapi hibrid FWA-GWO lebih unggul karena lebih dulu konvergen dibandingkan GWO.

Tabel 4.6 Hasil uji algoritma pada fungsi *Goldstein Price*

		Rata-rata Nilai Minimum	Standar Deviasi	Nilai Terbaik	Rata-rata Waktu Komputasi (detik)
V1	FWA	361.6065	394.9524	32.7245	3.3386
	GWO	3.000043	7.6085E-05	3	0.2944
	HIBRID	3.0001667	2.4810E-04	3	2.1227
V2	FWA	225.8318	281.7695	5.0936	3.8512
	GWO	3.000093	1.0306E-04	3	0.3633
	HIBRID	3.00025	4.1533E-04	3	2.1379
V3	FWA	160.2237	195.8309	12.0375	11.1518
	GWO	3.00005	6.1914E-05	3	0.4807
	HIBRID	3.00038	5.6769E-04	3	2.2690







Gambar 4.7. Hasil uji algoritma fungsi *Goldstein Price* dalam grafik

Hasil uji minimasi menggunakan metode FWA, GWO, dan hibrid FWA-GWO pada enam fungsi uji dengan tiga versi yang berbeda dan tiga dimensi yang berbeda, secara umum hibrid FWA-GWO menghasilkan kinerja yang lebih baik. Pada dua fungsi unimodal dan dua fungsi multimodal yang telah diuji, hibrid FWA-GWO dapat menghasilkan nilai minimum yang lebih baik dibandingkan FWA dan GWO, dari segi rata-rata nilai minimum, standar deviasi, dan nilai terbaik. Pada dua fungsi multimodal berdimensi tetap yang telah diuji, GWO lebih unggul dibandingkan FWA dan hibrid FWA GWO, dari segi rata-rata nilai minimum, standar deviasi, nilai terbaik, dan waktu kompotasi, akan tetapi pada fungsi multimodal berdimensi tetapi hibrid FWA-GWO mampu mencapai nilai minimum global. Apabila dilihat berdasarkan waktu komputasi pada enam fungsi uji, GWO membutuhkan waktu komputasi yang lebih lambat dibandingkan FWA dan hibrid FWA-GWO. Hal ini dikarenakan tahapan yang dilakukan untuk satu iterasi pada GWO lebih sederhana.

BAB V
KESIMPULAN

Hibrid FWA-GWO merupakan gabungan dua algoritma metaheuristik, yaitu FWA dan GWO. Hibrid FWA-GWO dapat digunakan untuk menyelesaikan masalah optimasi. Hal-hal yang dapat disimpulkan dari percobaan dan hasil analisis yang telah dilakukan adalah sebagai berikut.

1. Masalah optimasi fungsi nonlinear dapat diselesaikan menggunakan algoritma hibrid FWA-GWO yang pada skripsi ini digunakan untuk melakukan pencarian solusi optimal (nilai yang paling minimum).
2. Berdasarkan percobaan yang telah dilakukan terhadap keenam fungsi nonlinear, terbukti bahwa algoritma hibrid FWA-GWO menghasilkan kinerja yang lebih baik dibandingkan FWA dan GWO dikarenakan hibrid FWA-GWO mampu menghasilkan nilai yang lebih minimum dari segi rata-rata nilai minimum, standar deviasi dari nilai minimum, dan nilai terbaik (paling minimum). Hal ini dapat dilihat berdasarkan data hasil percobaan pada fungsi unimodal dan juga fungsi multimodal berdimensi tinggi.

DAFTAR PUSTAKA

Barraza, J., Rodriguez, L., Castillo, O., Melin, P., dan Valdez, F. 2018. A New Hybridization Approach between The Fireworks Algorithm and Grey Wolf Optimizer Algortihm. *Journal of Optimization*. Vol. 208, pp. 1-18.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., dan Stein, C. 2001. *Introduction to Algorithms (Second Edition)*. Massachusetts Institute of Technology: Mit Press.

Fitiyani, Away, Y., dan Gani, T. A. 2018. Pengaruh Inisialisasi Populasi Random Search pada Algoritma Berevolusi dalam Optimasi Travelling Salesman Problem (TSP). *Jurnal Nasional Komputasi dan Teknologi Informasi*. Vol.1, No. 2, pp. 49-55.

Hillier, F.S., dan Lieberman, G.J. 2010. *Introduction to Operation Research (Seventh Edition)*. McGraw Hill International.

Horn, R.A., dan Johnson, C.R. 2013. *Matrix Analysis: Second Edition*. New York: Cambridge University Press.

Madi, M., Markovi, D., dan Radovanovi, M. 2013. Comparasion of Meta-Heuristic Algorithms for Solving Machining Optimization Problems. *Mechanical Engineering*. Vol. 11, No. 1, pp.29-44.

Million, E. 2007. The Hadamard Product. *Creative Commons Attribution 3.0 United States License*.

Mirjalili, S., Mirjalili, S.M., dan Lewis, A. 2014. Grey Wolf Optimizer. *Advances in Engineering Software*. Vol. 69, pp. 46-61.

Rajabioun, R. 2011. Cuckoo Optimization Algorithm. *Applied Soft Computing*. Vol. 11, pp. 5508-5518.

Rao, S.S. 2009. *Engineering Optimization: Theory and Practice (Forth Edition)*. Hoboken: John Willey & Sons, Inc.

Sugioko, A. 2013. Perbandingan Algoritma Bee Colony dengan Algoritma Bee Colony Tabu List dalam Penjadwalan FlowShop. *Jurnal Metris*. Vol. 14, pp. 113-120.

Sun, W., dan Yuan, Y. 2006. *Optimization Theory and Methods*: *Nonlinear Programming*. New York: Springer.

Talbi, E.G. 2009. *Metaheuristic: From Design to Implementation*. Hoboken: John Wiley and Sons Ltd.

Tan, Y., dan Zhu, Y. 2010. *Fireworks Algorithm for Optimization*. Verlag, Berlin, Heidelberg, Germany: Springer.

Tan, Y. 2015. *Fireworks Algorithm*.Verlag, Berlin, Heidelberg, Germany: Springer.

Yang, X., Deb, S., dan Fong, S. 2012. Accelerated Particle Swarm Optimization and Support Vector Machine for Business Optimization and Applications. *Communications in Computer and Information Science*. Vol.136, pp. 53-66.

Yang, X., Fong, S., dan Deb, S. 2010. *Nature-Inspired Metaheuristic Algorithms*. Frome: Luniver Press.

Yang, X. 2014. *Nature-Inspired Optimization Algorithms*. First Edition. London: Elsevier.