PENERAPAN *FOREST OPTIMIZATION ALGORITHM* (FOA) DALAM MENYELESAIKAN MASALAH OPTIMASI FUNGSI NONLINEAR TANPA KENDALA

SKRIPSI

Oleh**:**
RAKA YHUDA PRATAMA
145090407111020

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENERAPAN *FOREST OPTIMIZATION ALGORITHM* (FOA) DALAM MENYELESAIKAN MASALAH OPTIMASI FUNGSI NONLINEAR TANPA KENDALA

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana Matematika

Oleh:
RAKA YHUDA PRATAMA
145090407111020

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG
2018

LEMBAR PENGESAHAN SKRIPSI

PENERAPAN *FOREST OPTIMIZATION ALGORITHM* (FOA) DALAM MENYELESAIKAN MASALAH OPTIMASI FUNGSI NONLINEAR TANPA KENDALA

oleh:
RAKA YHUDA PRATAMA
145090407111020

Setelah dipertahankan di depan Majelis Penguji dan dinyatakan memenuhi syarat untuk memperoleh gelar Sarjana Matematika

Pembimbing,

Indah Yanti, S.Si., M.Si.
NIP. 197911292005012002

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 : Raka Yhuda Pratama
NIM : 145090407111020
Jurusan : Matematika
Penulis Skripsi Berjudul : Penerapan *Forest Optimization Algorithm* (FOA) dalam Menyelesaikan Masalah Optimasi Fungsi Nonlinear Tanpa Kendala

dengan ini menyatakan bahwa :

1. **Isi Skripsi ini adalah benar-benar karya saya sendiri dan bukan hasil menjiplak karya orang lain. Karya-karya yang tercantum dalam Daftar Pustaka Skripsi ini semata-mata digunakan sebagai acuan/referensi.**
2. **Apabila di kemudian hari diketahui bahwa isi Skripsi saya merupakan hasil jiplakan, maka saya bersedia menanggung akibat hukum dari keadaan tersebut.**

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 25 Juli 2018
Yang menyatakan,

(Raka Yhuda Pratama)
NIM. 145090407111020

PENERAPAN *FOREST OPTIMIZATION ALGORITHM* (FOA) DALAM MENYELESAIKAN MASALAH OPTIMASI FUNGSI NONLINEAR TANPA KENDALA

ABSTRAK

Optimasi adalah proses membuat sesuatu menjadi lebih baik dalam mencari solusi maksimum maupun minimum. Masalah optimasi dapat kita temui dalam berbagai bidang antara lain: ekonomi, keuangan, transportasi, persediaan, dan sains komputasi. Pada optimasi, masalah yang sering ditemukan adalah sebuah solusi berhenti pada optimum lokal dan jauh dari solusi optimum global. Algoritma heuristik dikenal baik mampu menyelesaikan masalah optimasi fungsi, contoh algoritma heuristik adalah *Particle Swarm Optimization* (PSO). Pada skripsi ini dibahas metode optimasi *Forest Optimization Algorithm* (FOA) yang digunakan untuk menyelesaikan masalah optimasi nonlinear tanpa kendala dengan beberapa fungsi uji yang dibandingkan dengan metode *Particle Swarm Optimization* (PSO). Fungsi uji yang digunakan mempunyai ukuran dimensi 2, 10, dan 30. Hasil perbandingan optimasi fungsi uji dengan menggunakan FOA dan PSO memperlihatkan bahwa FOA mampu menyelesaikan masalah optimasi lebih baik dari segi akurasi nilai *fitness*, rata-rata *fitness*, standar deviasi dengan waktu komputasi yang relatif lebih cepat dibandingkan PSO.

Kata Kunci: *FOA, Optimasi, Hutan, Fungsi Nonlinear, Tanpa Kendala.*

APPLICATION OF FOREST OPTIMIZATION ALGORITHM (FOA) IN OPTIMIZING UNCONSTRAINT NONLINEAR FUNCTION PROBLEMS

ABSTRACT

Optimization is the process of making things better for finding the maximum and minimum solutions. Optimization problems can be found in various fields including: economy, finance, transportation, inventory, and computational science. On optimization, the most common problem is a solution stopping at the local optimum and away from the global optimum solution. The heuristic algorithm is well known to solve the problem of function optimization, the example of the heuristic algorithm is Particle Swarm Optimization (PSO). This thesis discusses the optimization method of Forest Optimization Algorithm (FOA) which is used to solve unconstraint nonlinear optimization problem with some test functions compared to Particle Swarm Optimization (PSO) method. The test functions used have dimension dimensions of 2, 10, and 30. The result of optimization ratio of test functions using FOA and PSO show that FOA able to solve optimization problem better in terms of fitness value accuracy, fitness average, standard deviation with computation time relatively faster than PSO.

Keywords: *FOA, Optimization, Forest, Nonlinear Function, Unconstraint.*

KATA PENGANTAR

Puji syukur kehadirat Allah SWT atas berkat dan rahmat-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul **"Penerapan *Forest Optimization Algorithm* (FOA) dalam Menyelesaikan Masalah Optimasi Fungsi Nonlinear Tanpa Kendala"** sebagai salah satu syarat untuk memperoleh gelar Sarjana Matematika.

Dalam penyusunan skripsi ini, banyak pihak yang telah memberikan dukungan dan bantuan. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Dra. Ari Andari, MS selaku dosen pembimbing akademik atas dukungan, doa, dan saran yang membangun selama penulis menempuh perkuliahan awal hingga akhir semester.
2. Indah Yanti, S.Si., M.Si selaku dosen pembimbing atas waktu, bimbingan, motivasi yang telah diberikan dan kesabaran yang luar biasa dalam membimbing penulis untuk menyusun skripsi.
3. Syaiful Anam, S.Si., MT., Ph.D dan Ummu Habibah, S.Si., M.Si., Ph.D selaku dosen penguji atas bimbingan yang telah diberikan.
4. Ratno Bagus Edy Wibowo, S.Si., M.Si., Ph.D. selaku Ketua Jurusan Matematika Universitas Brawijaya.
5. Seluruh dosen, staff, dan karyawan Jurusan Matematika Fakultas MIPA Universitas Brawijaya atas segala bantuannya.
6. Sunarman (bapak), Parwanti (ibu), Wendy Agil Nugroho (adik), dan seluruh keluarga di Solo, Balikpapan, Lampung, Tulungagung, dan Samarinda yang paling penulis sayangi atas doa, nasihat, dan motivasi yang diberikan kepada penulis dalam menyelesaikan skripsi.
7. Sahabat-sahabat dan keluarga besar Matematika 2014 yang telah memberikan doa dan semangat dalam penulisan skripsi.
8. Seluruh teman RITMA FMIPA UB yang banyak memberikan pengalaman dalam hal berkompetisi dan kepenulisan.
9. Salma Maulida yang menjadi salah satu *moodbooster* andalan.
10. Seluruh pihak yang telah membantu selama penulisan skripsi ini yang tidak dapat penulis sebutkan satu persatu.

Semoga Tuhan Yang Maha Kuasa memberikan anugerah dan karunia-Nya kepada semua pihak yang telah membantu menyelesaikan skripsi ini.

Penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan. Oleh karena itu, dengan segala kerendahan hati penulis mengharapkan kritik dan saran yang membangun demi perbaikan kedepannya. Kritik dan saran dapat dikirim melalui email penulis pratamaraka10@gmail.com.

Semoga skripsi ini dapat bermanfaat bagi segenap pihak yang membutuhkan dan mampu menjadi sumber motivasi serta inspirasi dalam dunia penelitian.

Malang, Juli 2018

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

DAFTAR LAMPIRAN

Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang

Optimasi adalah proses membuat sesuatu menjadi lebih baik (Rajabioun, 2011). Dalam kehidupan sehari-hari sering kali ditemui masalah optimasi yang membutuhkan alat atau metode optimasi terbaik dengan harapan agar menghasilkan *output* yang maksimum atau minimum. Masalah optimasi dapat kita temui dalam berbagai bidang antara lain: ekonomi, keuangan, transportasi, persediaan, dan sains komputasi. Ruang pencarian nilai optimal dibedakan antara optimum lokal dan optimum global. Optimum lokal adalah nilai optimal yang dicapai dalam rentang nilai tertentu. Sedangkan optimum global adalah nilai optimal dari seluruh anggota domain. Masalah yang sering dihadapi dalam masalah optimasi adalah metode optimasi berhenti pada solusi optimum lokal dan jauh dari solusi optimum global. Selain itu jumlah dimensi atau variabel yang banyak serta bentuk persamaan fungsi yang rumit juga berpengaruh pada hasil optimasi.

Dalam dunia komputasi, permasalahan optimasi global dapat diselesaikan dengan algoritma heuristik. Algoritma heuristik digunakan untuk mencari solusi optimum dengan cara pendekatan serta *trial and error* dari segala kemungkinan yang ada. Solusi yang didapatkan tidak menjamin optimal seperti halnya solusi eksak namun hasilnya sudah cukup layak untuk diterima (Foulds, 1984). Beberapa metode optimasi heuristik yang telah dikembangkan antara lain: *Genetic Algorithm* (GA) (Sivanandam dan Deepa, 2008), *Simulted Annealing* (Pahm dan Karaboga, 2000), dan *Particle Swarm Optimization* (PSO) (Kennedy dan Eberhart, 1995).

Penerapan metode optimasi heuristik juga diteliti dalam menyelesaikan masalah optimasi fungsi antara lain: penerapan *Particle Swarm Optimization* untuk optimasi fungsi pada masalah kebisingan lingkungan yang diteliti oleh Pan, dkk., (2006). *Fuzzy Adaptive Turbulance Particle Swarm Optimization* (FATPSO*)* untuk masalah optimasi fungsi nonlinear oleh Dianto (2009). Penerapan *Ant Colony Optimization* dalam optimasi fungsi kontinu menggunakan *Novel Pheromone Updating* (NPU) oleh Seckiner, dkk., (2013), dan Optimasi fungsi uji menggunakan *Genetic Algorithm* oleh Yadaf dan Ahmad (2013).

Pada skripsi ini dibahas metode optimasi *Forest Optimization Algorithm* (FOA) yang digunakan untuk menyelesaikan masalah optimasi dengan beberapa fungsi uji tanpa kendala yang dibandingkan dengan metode *Particle Swarm Optimization* merujuk pada artikel yang ditulis oleh Ghaemi dan Derakhshi (2014). *Forest Optimization Algorithm* (FOA) terinspirasi oleh proses alam yaitu perilaku pepohonan di suatu hutan yang mampu bertahan hidup lama selama beberapa dekade. Tiap pohon mewakili solusi optimum masalah yang dihadapi dan pohon terbaik dengan umur dan nilai variabel terbaik menjadi solusi optimum dari masalah yang dihadapi. Keunggulan FOA dikenal mampu menyelesaikan masalah optimasi fungsi nonlinear dan *feature weighting* (fitur pembobotan) yang telah diteliti oleh Ghaemi dan Derakhshi (2014). Indikator uji dari kedua metode tersebut adalah sejauh mana algoritma menemukan solusi optimum, lama waktu komputasi, dan standar deviasi dari tiap solusi. Simulasi numerik dilakukan dengan bantuan *software* Matlab 2017a.

1.2 Rumusan Masalah

Berdasarkan latar belakang, maka rumusan masalah dalam skripsi ini adalah sebagai berikut:

1. Bagaimana menyelesaikan masalah optimasi nonlinear tanpa kendala dengan FOA?
2. Bagaimana perbandingan FOA dan PSO dalam menyelesaikan masalah optimasi nonlinear tanpa kendala?

1.3 Tujuan

Dari rumusan masalah tersebut, tujuan dari skripsi ini adalah sebagai berikut:

1. Menyelesaikan masalah optimasi nonlinear tanpa kendala dengan FOA.
2. Mengetahui perbandingan FOA dan PSO dalam menyelesaikan masalah optimasi nonlinear tanpa kendala.

1.4 Batasan Masalah

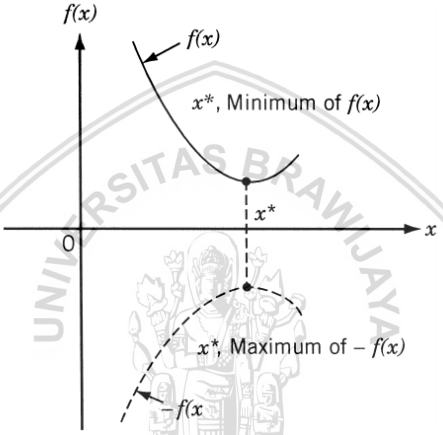
Dari rumusan masalah tersebut, terdapat batasan, yakni:

1. Fungsi uji yang digunakan pada skripsi ini adalah fungsi *Griewank, Sum of Different Power*, *Egg Holder*, dan *Rastrigin*.
2. Dimensi fungsi yang digunakan dalam pengujian optimasi adalah 2, 10, dan 30 dimensi.

BAB II
DASAR TEORI

2.1 Optimasi Nonlinear Tanpa Kendala

Optimasi adalah proses mendapatkan hasil terbaik dalam keadaan tertentu. Usaha yang diperlukan atau manfaat yang diinginkan dalam situasi praktis dapat diekspresikan dalam bentuk fungsi dari variabel keputusan tertentu, optimasi dapat didefinisikan sebagai proses untuk menemukan kondisi yang memberikan nilai maksimum atau minimum dari suatu fungsi.



Gambar 2.1. Minimum dari $f(x)$ setara maksimum dari $-f(x)$

Bisa dilihat dari Gambar. 2.1 bahwa jika titik x^* sesuai dengan nilai minimum fungsi $f(x)$, titik yang sama juga sesuai dengan $-f(x)$ atau nilai maksimum negatif dari fungsi. Suatu optimasi dikatakan nonlinear dan tanpa kendala jika berbentuk:

$$\min : z = f(\vec{x}),$$

dimana $\vec{x} = (x_1, x_2, \dots, x_n)$ merupakan vektor pada daerah $D \subseteq \mathbb{R}^n$.

(Rao, 2009).

Berikut adalah penjelasan dari optimum lokal dan optimum global.

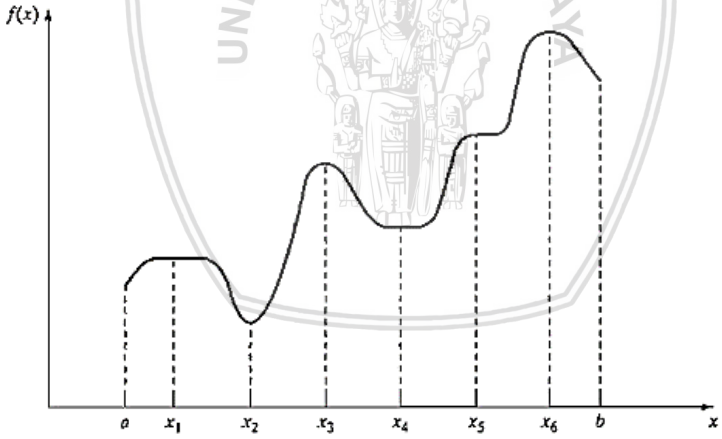
Suatu titik ekstrim dari fungsi $f(\vec{x})$ didefinisikan maksimum maupun minimum. Secara matematis, suatu titik di $\vec{x}^* = (x_1, x_2, \dots, x_n)$ adalah maksimum jika memenuhi

$$f(\vec{x}^* + \vec{h}) \leq f(\vec{x}^*)$$

untuk semua $\vec{h} = (h_1, \dots, h_j, \dots, h_n)$ dimana $|h_j|$ adalah nilai yang cukup kecil untuk semua j. Dengan kata lain $\vec{x}^*$ adalah titik maksimum jika $f(\vec{x}^*)$ pada setiap titik di persekitaran $\vec{x}^*$ tidak melebihi $f(\vec{x}^*)$. Pada kasus yang sama, $\vec{x}^*$ bernilai minimum jika:

$$f(\vec{x}^* + \vec{h}) \geq f(\vec{x}^*)$$

berikut adalah ilustrasi dari optimum (minimum dan maksimum) lokal dan global pada fungsi dengan interval $[a, b]$.



Gambar 2.2 Ilustrasi optimum lokal dan global

Terlihat pada fungsi tersebut global maksimum pada $f(x_6)$ dengan lokal maksimum pada $f(x_1)$, $f(x_3)$, dan $f(x_5)$. Global minimum terletak pada $f(x_2)$ dengan lokal minimum di $f(x_4)$.

(Taha, 2007).

Berikut adalah teorema kalkulus dalam menyelesaikan masalah optimasi fungsi multivariabel.

Teorema 2.1

Jika $f(\vec{x})$ memiliki suatu titik optimum (maksimum atau minimum) di $\vec{x} = \vec{x}^*$ dan jika turunan parsial pertama $(\nabla f(\vec{x}))$ ada maka:

$$\nabla f(\vec{x}) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n}\right] = 0. \tag{2.1}$$

Teorema 2.2

Misalkan $\vec{x}^*$ merupakan titik optimum dan H_f matriks turunan parsial kedua (Hessian matriks) $f(\vec{x})$ di $\vec{x} = \vec{x}^*$ maka:

(i) $\vec{x}^*$ adalah titik minimum, jika H_f definit positif.

(ii) $\vec{x}^*$ adalah titik maksimum, jika H_f definit negatif.

Berdasarkan komponennya, matriks Hessian ditulis dalam bentuk:

$$H_f = \begin{bmatrix} \frac{\partial^2 f}{\partial {x_1}^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial {x_n}^2} \end{bmatrix}. \tag{2.2}$$

Untuk mengetahui sifat definit matriks Hessian digunakan perhitungan nilai determinan. Misal diberikan matriks A berorde n dan diperoleh persamaan sebagai berikut

$$A_1 = |a_{11}|; A_2 = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}; A_n = \begin{vmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{vmatrix}; \tag{2.3}$$

Matriks A akan bernilai definit positif, jika dan hanya jika semua nilai determinan dari $A_1, A_2, \dots, A_n$ semuanya positif. Matriks A bernilai definit negatif, jika dan hanya jika tanda dari A_j adalah $(-1)^j$ untuk $j = 1,2, \dots, n$.

(Rao, 2009).

Berikut contoh masalah optimasi *nonlinear* tak berkendala:

$$\min \quad f(x_1, x_2) = 70x_1 + 4{x_1}^2 + 150x_2 + 15{x_2}^2 - 15x_1 - 15x_2$$

langkah pertama mencari titik stasioner menggunakan rumus $\nabla f(x_1, x_2) = 0$, sehingga

$$\frac{\partial f}{\partial x_1} = 70 + 8x_1 - 15 = 0; \frac{\partial f}{\partial x_2} = 150 + 30x_2 - 15 = 0$$

diperoleh titik stasioner pada $(-\frac{55}{8}, -\frac{9}{2})$, lalu untuk mengetahui sifat titik tersebut maka diperlukan uji matriks Hessian

$$\mathrm{H}_{f(x_1,x_2)} = \begin{bmatrix} 8 & 0 \\ 0 & 30 \end{bmatrix}$$

berdasarkan persamaan 2.3, matriks Hessian fungsi tersebut bersifat definit positif sehingga titik tersebut adalah titik minimum sehingga fungsi tersebut minimum pada $f\left(-\frac{55}{8}, -\frac{9}{2}\right) = 114.68$.

2.2 Algoritma Evolusi

Algoritma evolusi diilhami dari beberapa gejala yang ada di alam. Algoritma evolusi dimulai dengan inisialisasi populasi acak. Populasi kemudian berkembang menjadi beberapa generasi. Di setiap generasi, individu baru dipilih untuk menjadi orang tua. Mereka saling silang untuk menghasilkan yang baru individu, yang kemudian disebut keturunan individu. Individu yang dipilih secara acak individu kemudian mengalami mutasi tertentu. Setelah itu, algoritma memilih individu yang optimal untuk bertahan hidup menjadi generasi berikutnya sesuai dengan skema seleksi kelangsungan hidup yang dirancang terlebih dahulu (Jong, 2006). Salah satu contoh algoritma evolusi adalah *Particle Swarm Optimization*, berikut adalah penjelasan dari *Particle Swarm Optimization*:

***Particle Swarm Optimization* (PSO)**

Pada tahun 1995 algoritma *Particle Swarm Optimization* dikenalkan oleh Dr. Eberhart dan Dr. Kennedy yang merupakan algoritma optimasi yang terinspirasi dari kehidupan populasi burung

dan ikan dalam bertahan hidup. Perkembangan PSO sudah cukup pesat baik dari sisi pengaplikasian maupun pengembangan metodenya (Haupt dan Haupt, 2004).

Swarm di dalam algoritma *Particle Swarm Optimization* (PSO) merupakan kawanan yang diasumsikan mempunyai ukuran tertentu dengan setiap partikel posisi awalnya terletak di suatu lokasi yang acak dalam ruang multidimensi. Setiap partikel mempunyai 2 karakter yakni posisi (*position*) dan kecepatan (*velocity*). Setiap partikel bergerak dalam ruang atau *space* tertentu dan mengingat posisi terbaik yang pernah dilalui atau ditemukan terhadap sumber makanan atau nilai fungsi objektif. Setiap partikel menyampaikan informasi atau posisi terbaiknya kepada partikel yang lain dan menyesuaikan posisi dan kecepatan masing masing berdasarkan informasi yang diterima mengenai posisi yang bagus tersebut.

Partikel-partikel dalam PSO bergerak melalui pencarian dalam ruang pencarian dengan *velocity* yang dinamis yang disesuaikan berdasarkan perilaku historisnya. Oleh karena itu, partikel-partikel mempunyai kecenderungan untuk bergerak ke area penelusuran yang lebih baik setelah melewati proses penelusuran.

Pada algoritma PSO vektor *velocity* diperbaharui untuk masing-masing partikel. *Update velocity* dipengaruhi oleh kedua solusi yaitu *global best* yang berhubungan dengan nilai yang paling rendah yang pernah diperoleh dari suatu partikel dan solusi *local best* yang berhubungan dengan biaya yang paling rendah pada populasi awal. Jika solusi *local best* mempunyai suatu biaya yang kurang dari nilai solusi global yang ada, maka solusi *local best* menggantikan solusi *global best*.

Menurut Chen dan Shih (2013) dalam menyelesaikan masalah optimasi, algoritma PSO mempunyai beberapa tahap, yakni:

1. Menginisialisasi jumlah partikel sejumlah n_{pop}. Kecepatan dan posisi awal dari tiap partikel dalam n_{pop} dimensi ditentukan secara *random* (acak).
2. Menghitung kecepatan dari semua partikel. Semua partikel bergerak menuju titik optimal dengan suatu kecepatan. Semua kecepatan dari partikel diasumsikan sama dengan nol, iterasi dimulai dari $i = 1$.
3. Mengevaluasi nilai terbaik setiap partikel ditaksir menurut fungsi objektif (*objective function*) yang ditetapkan. Jika nilai terbaik

setiap partikel pada lokasi saat ini lebih baik dari posisi terbaik $\vec{P}_{best}$, maka $\vec{P}_{best}$ diatur untuk posisi saat ini.

4. Nilai terbaik partikel dibandingkan dengan nilai global terbaik G_{best} . Jika $\vec{G}_{best}$ yang terbaik maka $\vec{G}_{best}$ yang diperbaharui.
5. Memperbaharui kecepatan ($velocity$) dan posisi ($position$) setiap partikel ditunjukkan pada persamaan di bawah ini.

$$\vec{V}_{id}^{k+1} = \left(\omega \times \vec{V}_{id}^{k}\right) + c_1 \times rand_1 \times \left(\vec{P}_{id} - \vec{X}_{id}\right) + c_2 \times rand_2 \times \left(\vec{G}_{id} - \vec{X}_{id}\right), \quad (2.4)$$

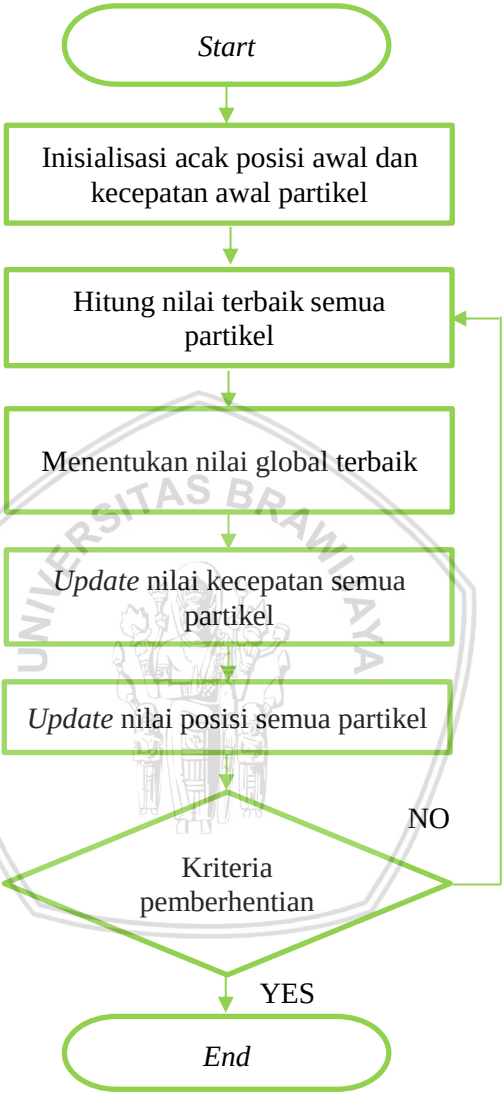
$$\vec{X}_{id}^{k+1} = \vec{X}_{id} + \vec{V}_{id}^{k+1}, \quad (2.5)$$

dimana:

$\vec{V}_{id}$	:	komponen kecepatan individu ke i pada d dimensi
$\vec{X}_{id}$	:	posisi individu i pada d dimensi
ω	:	parameter *inertia weight*
c_1, c_2	:	konstanta akselerasi (*learning rate*)
$rand_1, rand_2$	:	parameter nilai acak
$\vec{P}_{id}$	:	P_{best} (*local best*) individu i pada d dimensi
$\vec{G}_{id}$	:	G_{best} (*global best*) individu i pada d dimensi.

6. Cek apakah solusi sekarang sudah konvergen, jika posisi semua partikel menuju satu nilai yang sama, maka ini dikatakan konvergen. Jika belum memenuhi maka langkah 2 diulang dengan perubahan iterasi $i = i + 1$ dan menghitung kembali nilai baru $P_{best,j}$ dan G_{best} .
7. Kriteria pemberhentian (*stopping criteria*) dari algoritma ini jika dalam beberapa iterasi tidak terjadi perubahan nilai variabel G_{best} dalam rentang maksimum iterasi yang diberikan.

Algoritma dari *Particle Swarm Optimization* dapat dilihat pada Gambar 2.3 di bawah ini.



Gambar 2.3. Algoritma PSO

2.3 Fungsi Uji

Fungsi uji merupakan fungsi yang menjadi standar pilihan untuk uji kehandalan (*reliability*), efisiensi (*efficient*) dan validitas (*validation*) pada suatu metode optimasi global. Beberapa fungsi uji antara lain: fungsi unimodal (*Sum of Different Power)* dan multimodal (*Griewank, Egg Holder, Rastrigin)* (Jamil dan Yang, 2013). Fungsi tersebut dipilih untuk menguji performa dalam pencarian solusi optimum global dengan menggunakan metode FOA dan PSO.

2.3.1 Fungsi *Griewank*

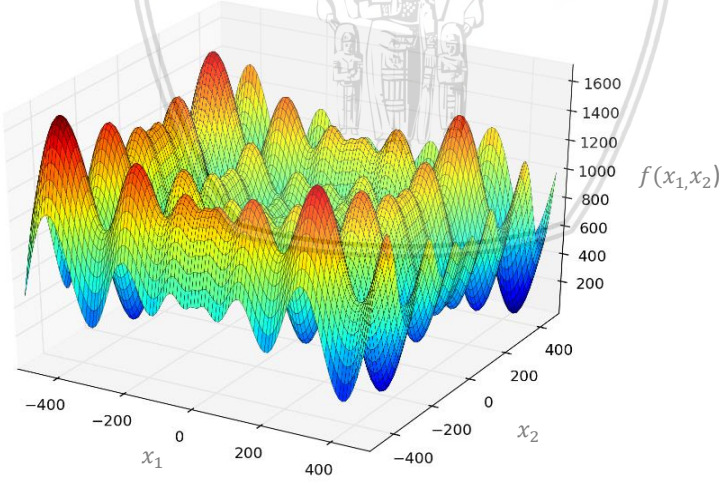
Fungsi *Griewank* merupakan fungsi multimodal. Fungsi ini memiliki banyak lokal minimum. Namun, lokasi lokal minimum didistribusikan secara teratur, berikut adalah persamaan fungsi *Griewank*.

$$f_n(\vec{x}) = 1 + \frac{1}{4000}\sum_{i=1}^{n} {x_i}^i - \prod_{i=1}^{n} \cos\frac{x_i}{\sqrt{i}}, \qquad (2.6)$$

$$domain: -600 \le x_i \le 600, i = 1,2,..,n.$$

global minimum $f(\vec{x}) = 0$, pada $\vec{x} = \vec{0}$.

Ilustrasi fungsi *Griewank* dapat dilihat pada Gambar 2.4



Gambar 2.4. Ilustrasi fungsi *Griewank*

(Molga dan Smutnicki, 2005)

2.3.2 Fungsi *Sum of Different Power*

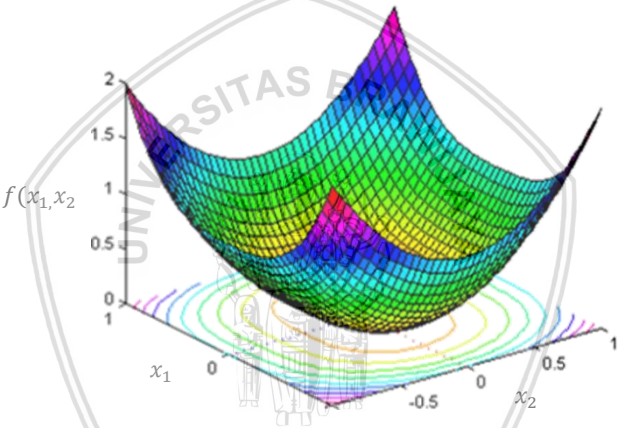
Fungsi *Sum of Different Power* sering juga digunakan dalam pengujian optimasi, fungsi ini merupakan fungsi unimodal yang mempunyai satu puncak. Fungsi tersebut didefinisikan sebagai berikut.

$$f_n(\vec{x}) = \sum_{i=1}^{n} |x_i|^{i+1}, \tag{2.7}$$

$$domain: -1 \leq x_i \leq 1, i = 1,2,..,n.$$

global minimum $f(\vec{x}) = 0$, pada $\vec{x} = \vec{0}$.

Ilustrasi fungsi *Sum of Different Power* dapat dilihat pada Gambar 2.5



Gambar 2.5. Ilustrasi fungsi *Sum of Different Power*

(Molga dan Smutnicki, 2005)

2.3.3 Fungsi *Egg Holder*

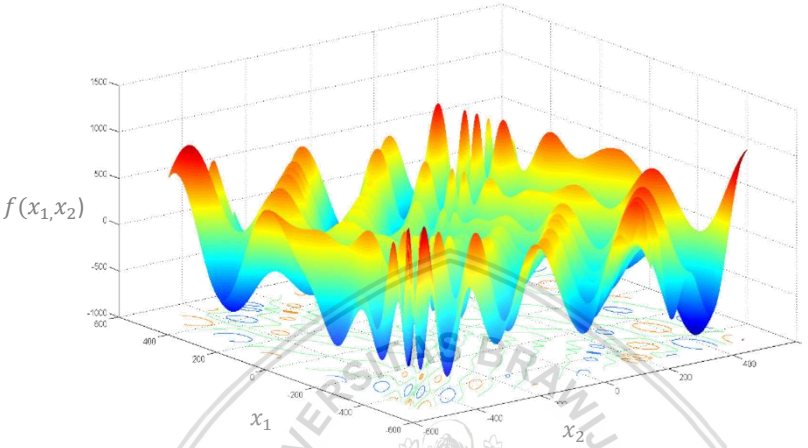
Fungsi *Egg Holder* mempunyai banyak optimum lokal yang tersebar dan merupakan fungsi multimodal, didefinisikan sebagai berikut:

$$f(x_1,x_2) = -(x_2+47)\sin\left(\sqrt{\left|x_2+\frac{x_1}{2}+47\right|}\right) - x_1\sin\left(\sqrt{|x_1-(x_2+47)|}\right), \tag{2.8}$$

$$domain: -512 \leq x_i \leq 512, i = 1,2.$$

global minimum $f(x_1, x_2) = -959.641$, pada $(x_1, x_2) = (512, 404.232)$.

Ilustrasi fungsi *Egg Holder* dapat dilihat pada Gambar 2.6



Gambar 2.6. Ilustrasi fungsi *Egg Holder*

(Jamil dan Yang, 2013).

2.3.4 Fungsi *Rastrigin*

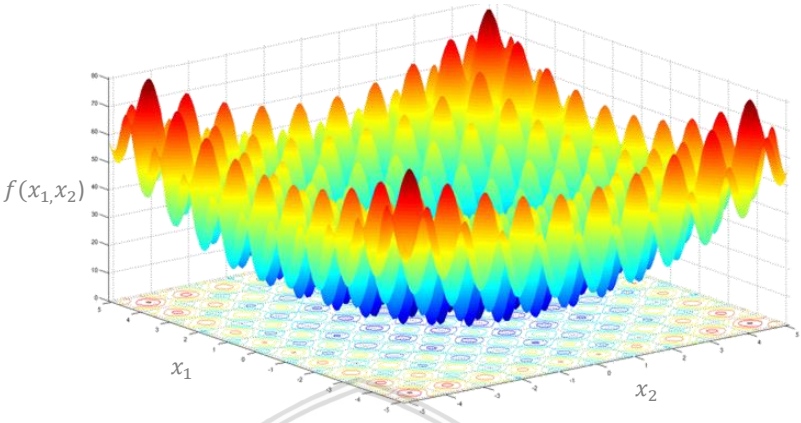
Sama halnya dengan fungsi *Griewank*, fungsi *Rastrigin* mempunyai banyak optimum lokal yang tersebar secara teratur dan didefinisikan sebagai berikut:

$$f_n(\vec{x}) = 10n + \sum_{i=1}^{n} [x_i{}^2 - 10\cos(2\pi x_i)], \qquad (2.9)$$

$$domain: -5.12 \leq x_i \leq 5.12, i = 1,2,..,n.$$

global minimum $f(\vec{x}) = 0$ pada $\vec{x} = \vec{0}$.

Ilustrasi fungsi *Rastrigin* dapat dilihat pada Gambar 2.7



Gambar 2.7. Ilustrasi fungsi *Rastrigin*

(Molga dan Smutnicki, 2005)

BAB III
HASIL DAN PEMBAHASAN

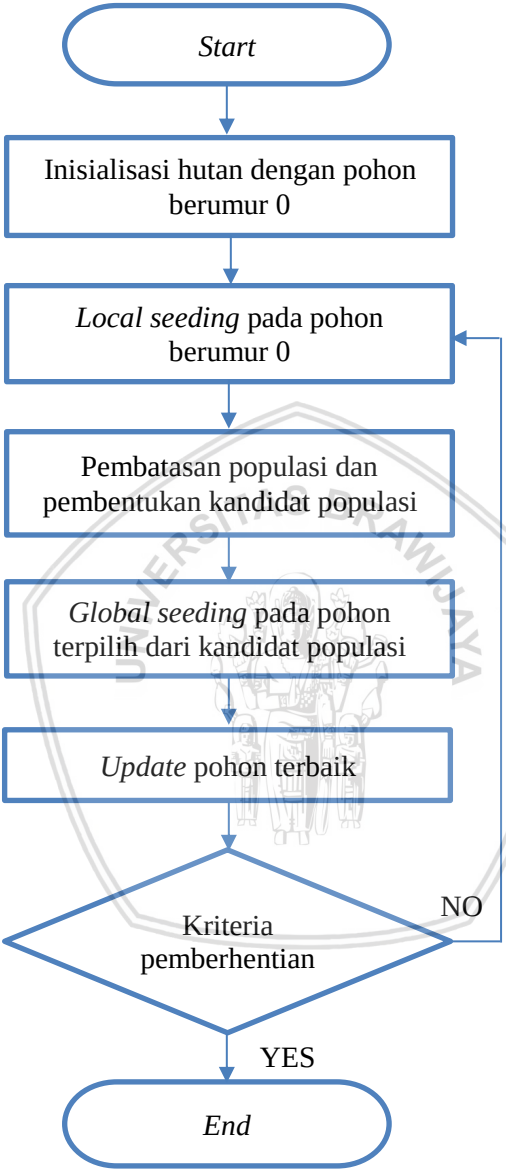
Pada bab ini dibahas tentang penerapan algoritma *Forest Optimization Algortihm* (FOA) dalam menyelesaikan masalah optimasi fungsi nonlinear tanpa kendala. FOA dibandingkan dengan algoritma *Particle Swarm Optimization* (PSO) dalam menyelesaikan 4 fungsi uji dengan *search area* (domain) dan minimum global yang sudah diketahui. Perbandingan dilakukan dengan simulasi numerik. Hasil perbandingan digunakan untuk penarikan kesimpulan dari performa algoritma FOA dalam menyelesaikan masalah optimasi fungsi nonlinear tanpa kendala.

3.1 *Forest Optimization Algortihm* (FOA)

FOA algoritma evolusioner yang diilhami oleh beberapa pohon di hutan yang bisa bertahan selama beberapa dekade, sementara pohon lainnya hanya bisa hidup untuk jangka waktu terbatas. Metode ini diusulkan oleh Ghaemi dan Derakhshi pada tahun 2014.

FOA dimulai dengan inisialisasi populasi pohon awal, masing-masing pohon mewakili suatu potensi solusi dari masalah, selain itu setiap pohon memiliki jumlah variabel dan umur. Pada awalnya usia pohon diatur ke 0. Selanjutnya penebaran benih lokal akan menghasilkan pohon baru (muda) dari pohon sebelumnya dengan umur 0 dan tambahkan pohon baru ke hutan. Kemudian, semua pohon, kecuali yang baru dihasilkan, usia mereka bertambah sebesar 1. Selanjutnya, ada kontrol terhadap populasi pepohonan di hutan dan beberapa pohon akan dihilangkan dari hutan dan mereka akan membentuk populasi kandidat untuk pembenihan tahap global.

Pada tahap pembenihan global menambahkan beberapa solusi potensial baru ke hutan untuk menyingkirkan optimum lokal. Lalu, pohon-pohon di hutan digolongkan menurut nilai terbaik dan pohon dipilih sebagai pohon terbaik dan usianya diatur ke 0 agar terhindar penuaan dan setelah itu mengeluarkan pohon terbaik dari hutan (karena tahap pembenihan setempat meningkatkan umur semua pohon termasuk umur pohon terbaik). Tahapan ini akan berlanjut sampai kriteria pemberhentian terpenuhi. Berikut adalah algoritma dari FOA pada Gambar 3.1.



Gambar 3.1. Algoritma FOA

(Ghaemi dan Derakhshi, 2014)

Inisialisasi pohon

Pada metode FOA, pohon-pohon dalam hutan merupakan solusi potensial dari setiap masalah. Setiap pohon mewakili array dengan dimensi 1 x $(N_{var} + 1)$ dimana, N_{var} adalah jumlah variabel dari masalah dan *Age* adalah umur dari pohon. Setiap pohon dalam tahap ini diatur dengan umur 0 dan nilai variabel diinisialisasi secara acak di domain fungsi objektif. Berikut adalah representasi variabel dari sebuah fungsi pada sebuah pohon.

$$Tree = [Age, v_1, v_2, \dots, v_{N_{var}}],$$

dengan $v_1, v_{2,} \dots, v_{Nvar}$ merupakan variabel dari masalah.

Contoh:

$$f(x_1, x_2, x_3) = 2x_1 - 3{x_1}^2 - {x_2}^2 + 0.5x_1x_3,$$

maka dapat didefinisikan pada setiap pohon:

$$Tree = [Age, v_1, v_2, v_3].$$

Local seeding

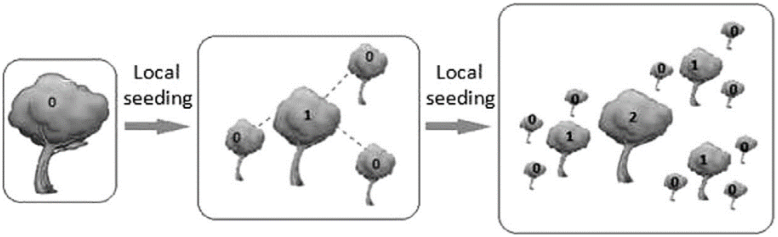
Di alam ketika proses pembenihan pohon dimulai, beberapa benih jatuh di dekat pohon dan menghasilkan pohon-pohon baru. Jumlah benih yang jatuh di dekat pohon dan menjadi pohon tetangga dengan umur 0 jatuh disebut "*Local Seeding Changes* (*LSC*)" dan ketika *LSC* berlangsung umur pohon induk bertambah 1.

Pada iterasi pertama dari algoritma semua pohon mempunyai umur 0, operator pada *local seeding* dijalankan pada semua pohon di hutan. Sehingga, setiap pohon berumur 0, jumlah *LSC* pohon baru akan ditambahkan di dalam hutan. Pada iterasi selanjutnya, jumlah pohon yang ditambahkan pada hutan akan berkurang dikarenakan akan ada pohon yang berumur lebih dari 0 yang tidak ambil bagian dalam tahap *local seeding*. *Local seeding* melakukan simulasi pencarian lokal dalam FOA.

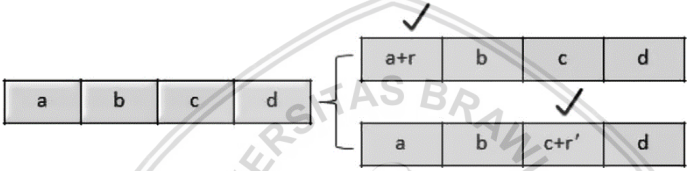
Ketika proses *local seeding* berlangsung akan dipilih variabel secara acak dari pohon yang terpilih. Setelah itu ditambahkan angka yang acak r, dimana $r \in [-\Delta x, \Delta x]$ dengan Δx merupakan angka kecil yang lebih kecil dari batas atas dari variabel terkait.

Pada kasus penambahan nilai, memungkinkan keadaan dimana nilai dari variabel menjadi kurang atau berlebih dibandingkan batas atas dan bawah dari variabel terkait. Untuk menghindari situasi

ini, nilai yang kurang dari batas bawah variabel dan nilai yang lebih dari batas atas variabel dipotong pada limit atau batasnya. Berikut contoh operator *local seeding* pada Gambar 3.2, Gambar 3.3, Gambar 3.4.



Gambar 3.2. Contoh *local seeding* pada satu pohon untuk 2 iterasi.



Gambar 3.3. Contoh *local seeding* untuk ruang pencarian yang berkelanjutan

Gambar 3.4. Contoh numerik *local seeding* pada satu pohon $LSC = 1, r' \in [-\Delta x, \Delta x] = [-1,1]$.

Pembatasan populasi

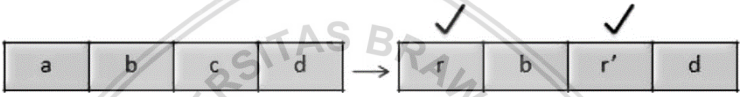
Jumlah pohon dalam hutan harus dibatasi untuk menghindari ledakan populasi. Ada dua parameter yang digunakan untuk membatasi populasi yakni "*life time*" dan "*area limit*". Pertama, pohon yang umurnya melebihi "*life time*" dikeluarkan dari hutan dan ditambahkan ke daftar kandidat populasi. Langkah selanjutnya adalah mengurutkan pohon di hutan menurut nilai optimalnya, jika jumlah pohon lebih besar daripada "*area limit*" dan ditambahkan ke daftar kandidat populasi.

Global seeding

Pilih "*transfer rate*" persentase populasi kandidat untuk melakukan pembenihan global. Selanjutnya pilih variabel acak dari

pohon yang dipilih. Nilai dari tiap variabel diubah dengan angka acak yang dibangkitkan dari rentang nilai variabel dan ditambahkan pohon baru dengan umur 0 di hutan. Jumlah variabel yang nilainya akan diubah disebut "*Global seeding Changes*" atau *GSC*.

Sebagai contoh operasi dari *global seeding* pada satu pohon *GSC* = 2, sehingga dua variabel dipilih secara acak dan nilai mereka diubah dengan dua angka acak seperti *r* dan *r'* pada rentang variabel terkait. Contoh numerik dari operasi *global seeding* digambarkan pada gambar dibawah dengan *GSC* = 2 dan rentang semua variabel dibuat sama [-5, 5]. Sebagai hasilnya, nilai dari 2 variabel acak yang terpilih diganti dengan nilai acak yang ada di rentang [-5, 5] seperti -0.7 dan 1.5. Berikut contoh dari operator global seeding pada Gambar 3.5 dan Gambar 3.6.



Gambar 3.5. Contoh *global seeding* pada satu pohon

Gambar 3.6. Contoh numerik *global seeding* pada satu pohon dengan *GSC* = 2

***Update* pohon terbaik**

Pada tahap ini, pohon diurutkan berdasarkan nilai optimal. Usia pohon dengan nilai terbaik diatur ke 0 dengan tujuan untuk menghindari penuaan dari pohon terbaik sebagai hasil dari *local seeding*. Dalam proses ini, ada kemungkinan bahwa pohon terbaik optimal lokal pada lokasinya oleh operasi *local seeding* dikarenakan *local seeding* dijalankan pada pohon berumur 0.

Kriteria pemberhentian

Kriteria pemberhentian adalah jika dalam beberapa iterasi tidak terjadi perubahan nilai pohon terbaik dalam rentang maksimum iterasi yang diberikan.

3.2 Uji Coba

Dalam tahap uji coba, algoritma FOA dan PSO diuji dalam menyelesaikan masalah optimasi fungsi nonlinear tanpa kendala dari fungsi uji yang disediakan, Fungsi tersebut diuji pada dimensi 2 (untuk fungsi *Egg holder*) dan dimensi 10, dimensi 30 (untuk fungsi *Griewank*, *Sum of Different Power*, dan *Rastrigin*). Setelah itu dilakukan simulasi numerik dan dibandingkan hasil optimasi dengan FOA dan PSO dari segi solusi $f(\vec{x})$ terbaik, rata-rata $f(\vec{x})$, standar deviasi, dan waktu komputasi. Simulasi numerik dilakukan dengan bantuan *software* Matlab 2017a.

3.2.1 Fungsi Uji

Fungsi yang diberikan mempunyai karakteristik tersendiri, Fungsi *Egg holder* mempunyai dimensi berukuran 2 pada persamaan 2.5, fungsi ini digunakan untuk menguji FOA dalam menyelesaikan optimasi fungsi berdimensi kecil. Fungsi *Griewank, Sum of Different Power* dan *Rastrigin* mempunyai banyak puncak optimum lokal yang menyebar secara teratur. Hal tersebut juga memungkinkan suatu metode optimasi menghadapi proses pencarian yang lebih rumit dan menyeluruh sehingga mampu menjadi pertimbangan dalam uji kehandalan dari performa FOA yang dibandingkan dengan PSO.

3.2.2 Parameter FOA dan PSO

Parameter dalam metode FOA dan PSO digunakan untuk mendukung perhitungan optimasi yang dilakukan secara numerik. Parameter dibawah merujuk dari artikel yang ditulis oleh Ghaemi dan Derakhshi (2014). Berikut adalah parameter yang digunakan dalam optimasi numerik fungsi uji dengan FOA dan PSO pada Tabel 3.1 dan Tabel 3.2:

Tabel 3.1. Parameter FOA

FOA	Keterangan
$nPop = 100$	Jumlah populasi
$LSC = 6$	*Local seeding changes*
$GSC = 1$	*Global seeding changes*
$lifeTime = 10$	Pembatasan umur
$areaLimit = 10$	Pembatasan populasi
$transferRate = 0.1$	Prosentase kandidat populasi

Tabel 3.2. Parameter PSO

PSO	Keterangan
$\vec{V}_{id}$ dan $\vec{X}_{id}$	Kecepatan dan posisi acak
$nPop = 100$	Jumlah populasi
$w = 0.8$	Momen inersia
$c_1 = 2$	Akselerasi personal
$c_2 = 1$	Akselerasi sosial

Dari Tabel 3.1 digunakan parameter jumlah populasi pohon FOA yakni $nPop = 100$ disamakan dengan parameter PSO. $LSC = 6$ berarti ada sebanyak 6 variabel dari tiap pohon yang nilainya ditambah dengan suatu nilai *random* dari batas bawah dan batas atas yang ditentukan yakni $[-1,1]$. Pembatasan populasinya adalah pembatasan umur $lifeTime = 100$ dan pembatasan populasi jumlah pohon sebanyak $areaLimit = 10$ Hasil dari pembatasan populasi adalah terbentuknya daftar kandidat populasi. Parameter $transferRate = 0.1$ artinya ada 10% pohon dari daftar kandidat populasi yang diikutkan dalam pencarian global. $GSC = 1$ berarti bahwa ada 1 variabel dari pohon yang nilainya akan diganti dengan angka *random.*

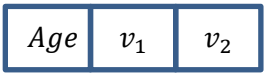
Parameter PSO pada Tabel 3.2 merujuk pada penelitian yang dilakukan oleh Ghaemi dan Derakhshi (2014). Digunakan parameter jumlah populasi $nPop = 100$. Setelah itu $\vec{V}_{id}$ dan $\vec{X}_{id}$ diinisialisasi secara acak. Momen inersia $\omega = 0.8$ mengatur kecepatan iterasi sebelumnya mempengaruhi kecepatan iterasi berikutnya . Sedangkan koefisien akselerasi personal partikel adalah $c_1 = 2$ dan akselerasi sosial dari partikel $c_2 = 1$.

3.2.3 Representasi masalah pada FOA

Pada tahap ini fungsi yang diujikan diinisialisasi sesuai karakteristik pendefinisian fungsi ke dalam bentuk algoritma FOA. Pohon dalam hutan diinisialiasi dengan jumlah $nPop = 100$. Pada algoritma FOA kandidat solusi dari masalah direpresentasikan dalam array sebuah pohon. Berikut adalah proses representasi masalah pada FOA:

Inisialisasi pohon

Fungsi uji yang digunakan dalam menguji performa FOA salah satunya adalah fungsi *Egg Holder* yang dapat dilihat pada persamaan 2.8. Karena dimensi atau variabel dari fungsi *Egg Holder* berjumlah 2 maka ukuran arraynya adalah (1×3) dengan batas atas pencarian 512 dan batas bawah pencarian -512. Berikut adalah representasi variabel dari sebuah fungsi pada sebuah pohon.



$$Tree = [Age, v_1, v_2].$$

Local seeding

Pada tahap ini pepohonan menebarkan benih dengan jumlah benih tiap pohon berbeda-beda secara alami. Operator *Local Seeding Changes* (LSC) memilih beberapa variabel dari pohon dan ditambahkan dengan angka r, acak (random) dimana $r \in [-\Delta x, \Delta x]$ dengan Δx merupakan angka kecil yang lebih kecil dari batas atas dari variabel terkait. Pada program FOA ini parameter nilai batas terbesar digunakan adalah $\Delta x = 1$, sehingga $r \in [-1, 1]$.

Pembatasan populasi

Pembatasan populasi dilakukan agar tidak terjadi ledakan populasi pohon dan untuk mempermudah membentuk kandidat populasi yang akan masuk pada tahap pencarian solusi global. Parameter dalam tahap ini adalah *lifeTime* $= 10$, jika umur pohon melebihi *lifeTime* maka pohon dikeluarkan dari hutan dan dimasukkan dalam kandidat. Setelah itu pohon-pohon diurutkan sesuai nilai terbaik. Langkah selanjutnya bergerak pada parameter *areaLimit* $= 10$, jika jumlah pohon dalam hutan melebihi *areaLimit*, maka pohon-pohon masuk pada daftar kandidat populasi.

Global seeding

Langkah pertama adalah memiilih "*transfer rate*" persentase populasi kandidat untuk melakukan pembenihan global. Selanjutnya pilih variabel acak dari pohon yang dipilih untuk diubah nilainya. Jumlah variabel yang nilainya akan diubah disebut "*Global seeding Changes*" atau GSC.

Pada program ini parameter *transferRate* $= 0.1$ atau 10% dan *GSC* $= 1$. Misalkan ada 50 pohon yang masuk kandidat populasi maka ada $50 \times 0.1 = 5$ pohon yang akan dilanjutkan pada operator *GSC*. *GSC* memilih 1 variabel dari pohon-pohon dan diganti nilainya dari $r \in [-1, 1]$.

***Update* pohon terbaik**

Pohon diurutkan sesuai nilai terbaik dan pohon-pohon terbaik umurnya diatur menjadi "0" untuk menghindari penuaan. Nilai terbaik dari pohon-pohon tersebut adalah solusi minimum global yang dicari dari fungsi non liner tanpa kendala.

Kriteria pemberhentian

Kriteria pemberhentian FOA adalah jika dalam 100 iterasi tidak terjadi perubahan nilai pohon terbaik dalam rentang 5000 iterasi.

Contoh perhitungan manual optimasi fungsi dengan FOA tercantum pada Lampiran 1.

3.2.4 Representasi masalah pada PSO

Masalah fungsi uji pada tahap ini disesuaikan dan diinisialisasi sesuai karakteristik pendefinisian fungsi ke dalam bentuk algoritma PSO. Kawanan atau partikel dalam populasi diinisialiasi dengan jumlah *nPop* $= 100$. Pada algoritma PSO kandidat solusi dari masalah direpresentasikan dalam suatu partikel yang memiliki *position* (posisi) dan *velocity* (kecepatan). Berikut adalah proses representasi masalah pada PSO:

Inisialisasi populasi

Pertama jumlah partikel dari populasi diinisialiasi sejumlah *nPop* $= 100$. Kecepatan dan posisi awal dari tiap partikel dalam *nPop* $= 100$ dimensi ditentukan secara *random* (acak) dengan nilai dari tiap variabel acak.

Menghitung kecepatan semua partikel

Semua partikel bergerak menuju titik optimal dengan suatu kecepatan. Semua kecepatan dari partikel diasumsikan sama dengan nol, iterasi dimulai dari $i = 1$.

Menghitung nilai terbaik semua partikel

Mengevaluasi nilai terbaik setiap partikel ditaksir menurut 4 fungsi uji yang ada pada persamaan 2.6 hingga persamaan 2.9. Jika nilai terbaik setiap partikel pada lokasi saat ini lebih baik dari $\vec{P}_{best}$, maka $\vec{P}_{best}$ diatur untuk posisi saat ini.

Menentukan partikel terbaik dan partikel global terbaik

Pada tahap ini nilai terbaik partikel dibandingkan dengan $\vec{G}_{best}$. Jika $Gbest$ yang terbaik maka $\vec{G}_{best}$ yang diperbaharui.

***Update* kecepatan dan posisi**

Memperbaharui kecepatan ($velocity$) dan posisi ($position$) setiap partikel ditunjukkan pada persamaan di bawah ini. Rumus kecepatan merujuk pada persamaan 2.4 dan rumus posisi merujuk pada persamaan 2.5.

Kriteria pemberhentian

Kriteria pemberhentian PSO adalah jika dalam 100 iterasi tidak terjadi perubahan nilai variabel $\vec{G}_{best}$ dalam rentang 5000 iterasi.

Contoh perhitungan manual optimasi fungsi dengan PSO tercantum pada Lampiran 2.

3.3 Perbandingan Hasil FOA dan PSO

Perbandingan hasil optimasi fungsi uji menggunakan FOA dan PSO dilakukan sebanyak 30 kali percobaan. Indikator yang diujikan adalah:

1. Nilai terbaik (*Best* $f(\vec{x})$) untuk mengetahui sejauh mana algoritma mampu mencapai nilai minimum global.

2. Rata-rata $f(\vec{x})$ untuk mengetahui rata-rata nilai yang dicapai dari 30 kali uji dengan rumus:

$$\overline{f(\vec{x})} = \frac{1}{n}\sum_{i=1}^{n} f_i(\vec{x})$$

dimana:

n : jumlah data
$\overline{f(\vec{x})}$: rata-rata $f(\vec{x})$ hingga data ke$-n$
$f_i(\vec{x})$: nilai data ke$-i$

3. Standar deviasi (SD) untuk mengetahui rata-rata penyimpangan data dari rata-rata (*mean*). Jika nilai SD jauh lebih besar dibanding nilai rata-rata merupakan representasi yang buruk dari keseluruhan data. Sedangkan jika nilai SD sangat kecil dibandingkan *mean*, maka nilai *mean* dapat digunakan sebagai representasi dari keseluruhan data.

$$SD = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (f_i(\vec{x}) - \overline{f(\vec{x})})^2}$$

dimana:

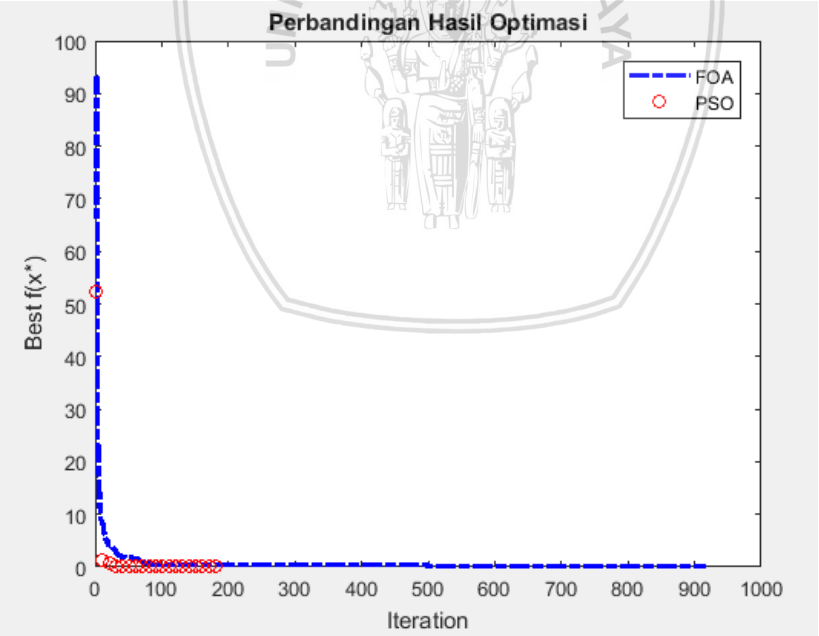
n : jumlah data
$\overline{f(\vec{x})}$: rata-rata $f(\vec{x})$ hingga data ke$-n$
$f_i(\vec{x})$: nilai data ke$-i$
SD : standar deviasi

(Brown, 1982).

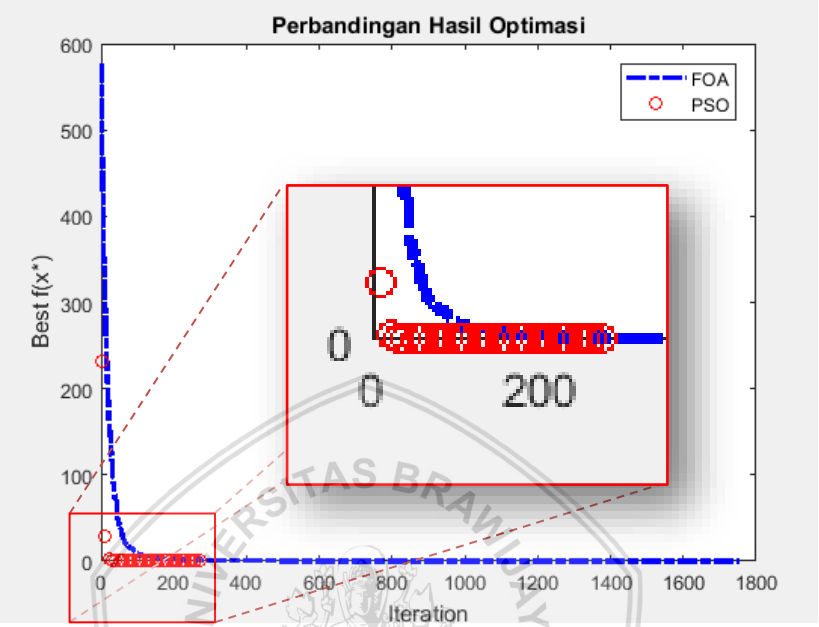
4. Waktu komputasi untuk mengetahui lamanya waktu (satuan detik) algoritma dalam menyelesaikan masalah optimasi.

Tabel. 3.3 Perbandingan optimasi fungsi 1

Fungsi 1 - *Griewank*				
Global Minimum $f(\vec{x}) = 0, \vec{x} = \vec{0}$				
Uji Coba: 30 kali				
	FOA		**PSO**	
Dimensi	**10**	**30**	**10**	**30**
Best $f(\vec{x})$	0.017	0.000	0.030	0.209
Rata-rata $f(\vec{x})$	0.229	0.006	0.130	0.806
Standar deviasi	0.135	0.005	0.061	0.367
Rata-rata waktu komputasi (detik)	2.308	9.779	2.482	4.590



Gambar 3.7 Hasil optimasi fungsi 1 (10 dimensi)



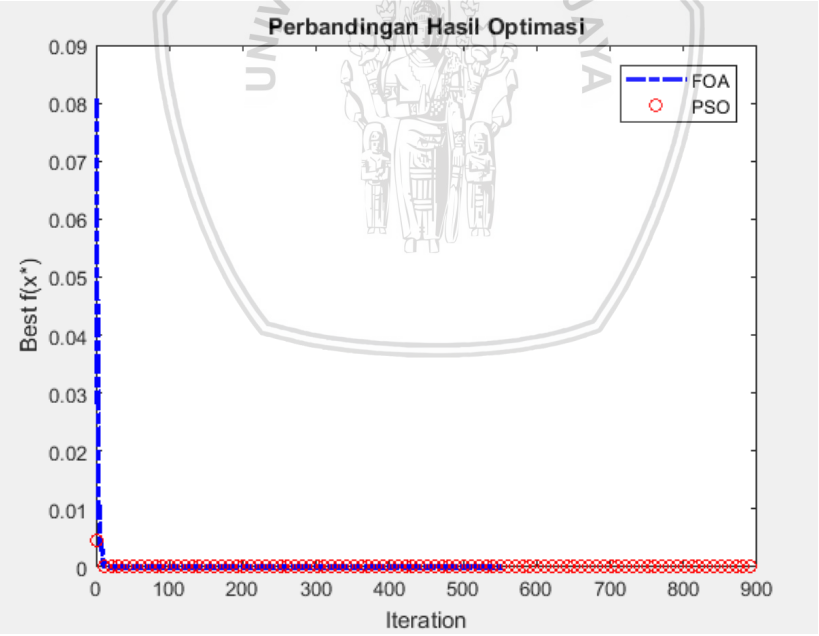
Gambar 3.8 Hasil optimasi fungsi 1 (30 dimensi)

Fungsi *Griewank* mempunyai solusi global minimum $f(\vec{x}) = 0$, pada $\vec{x} = 0, i = 1,2,..,n$. Dari (Tabel 3.3) dan (Gambar 3.7) dan (Gambar 3.8) hasil optimasi fungsi *Griewank* dengan FOA dan PSO diperoleh perbandingan sebagai berikut:

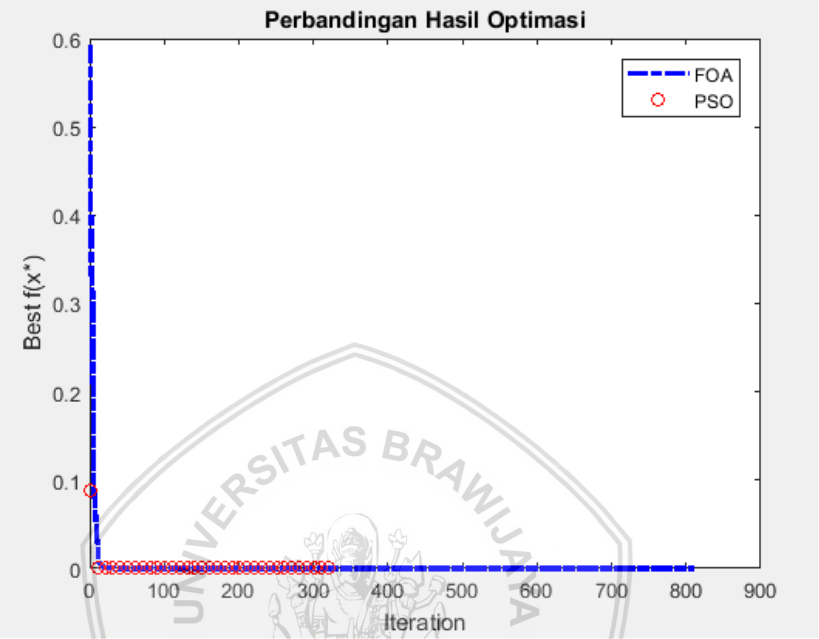
1. Pada optimasi fungsi 10 dimensi, *best* $f(\vec{x})$ dan rata-rata waktu komputasi FOA lebih baik dibandingkan PSO, sedangkan rata-rata $f(\vec{x})$ dan standar deviasi $f(\vec{x})$ yang dihasilkan oleh PSO lebih baik dibandingkan FOA.
2. Pada optimasi fungsi 30 dimensi, *best* $f(\vec{x})$, rata-rata $f(\vec{x})$, standar deviasi $f(\vec{x})$, dan rata-rata waktu komputasi yang dihasilkan oleh FOA lebih baik dibandingkan PSO.
3. PSO lebih cepat konvergen namun belum mampu mencapai nilai global minimum dibandingkan FOA.

Tabel. 3.4 Perbandingan optimasi fungsi 2

Fungsi 2 – *Sum of Different Power*				
Global Minimum $f(\vec{x}) = 0, \vec{x} = \vec{0}$				
Uji Coba: 30 kali				
	FOA		**PSO**	
Dimensi	**10**	**30**	**10**	**30**
Best $f(\vec{x})$	0.000	0.000	0.000	0.000
Rata-rata $f(\vec{x})$	0.000	0.000	0.000	0.000
Standar deviasi	0.000	0.000	0.000	0.000
Rata-rata waktu komputasi (detik)	2.276	3.905	12.673	6.283



Gambar 3.9 Hasil optimasi fungsi 2 (10 dimensi)



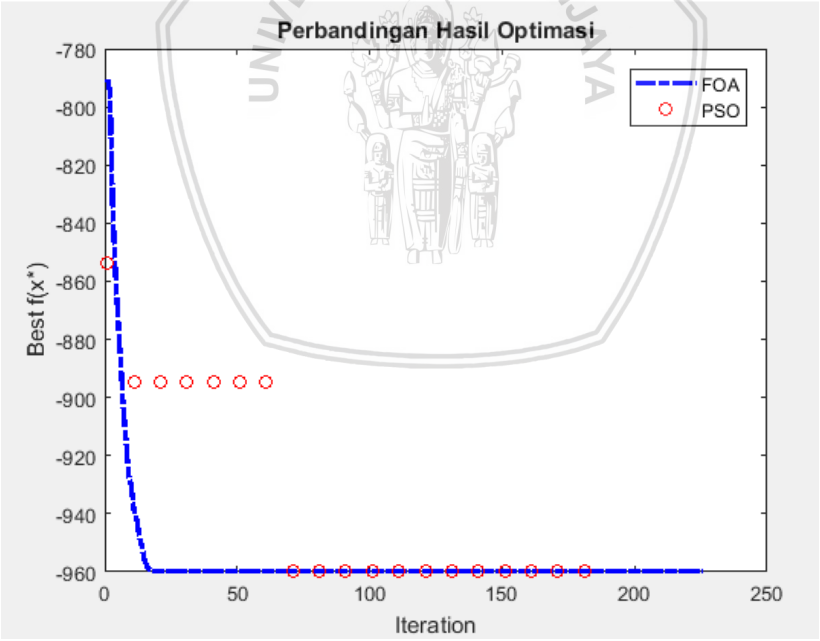
Gambar 3.10 Hasil optimasi fungsi 2 (30 dimensi)

Fungsi *Sum of Different Power* mempunyai solusi global minimum $f(\vec{x}) = 0$, pada $\vec{x} = 0, i = 1,2,..,n$. Dari (Tabel 3.4) dan (Gambar 3.9) dan (Gambar 3.10) hasil optimasi fungsi *Sum of Different Power* dengan FOA dan PSO diperoleh perbandingan sebagai berikut:

1. Pada optimasi fungsi 10 dimensi dan 30 dimensi *best* $f(\vec{x})$, rata-rata $f(\vec{x})$, standar deviasi $f(\vec{x})$, dan rata-rata waktu komputasi yang dihasilkan oleh FOA sama dengan PSO namun, untuk rata-rata waktu komputasi FOA lebih baik dari PSO.
2. PSO lebih cepat konvergen daripada FOA namun dari 4 kriteria yang dibandingkan performa FOA lebih baik.

Tabel. 3.5 Perbandingan optimasi fungsi 3

Fungsi 3 – *Egg Holder*		
Global Minimum $f(x_1, x_2) = -959.641$ $(x_1, x_2) = (512, 404.232)$		
Uji Coba: 30 kali		
	FOA	**PSO**
Dimensi	**2**	**2**
Best $f(\vec{x})$	-959.641	-959.641
Rata-rata $f(\vec{x})$	-925.452	-926.302
Standar deviasi	32.601	56.639
Rata-rata waktu komputasi (detik)	1.359	2.359



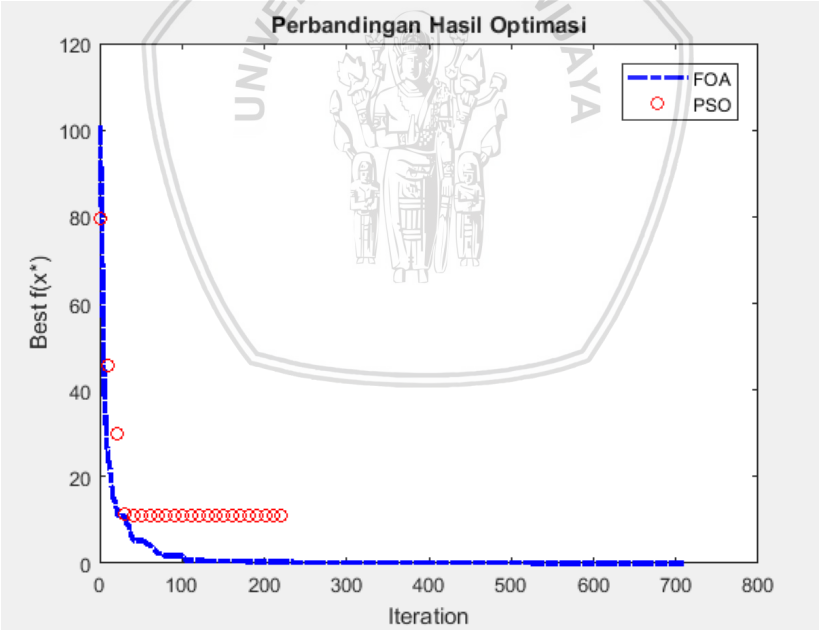
Gambar 3.11 Hasil optimasi fungsi 3 (2 dimensi)

Fungsi *Egg Holder* mempunyai solusi global minimum $f(x_1, x_2) = -959.641$, pada $(x_1, x_2) = (512, 404.232)$. Dari (Tabel 3.5) dan (Gambar 3.11) hasil optimasi fungsi *Egg Holder* dengan FOA dan PSO diperoleh perbandingan sebagai berikut:

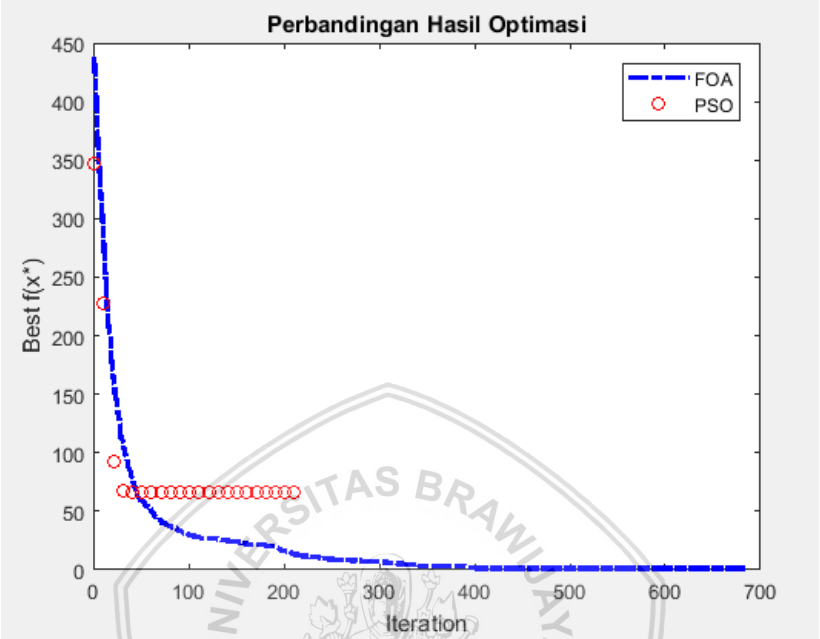
1. Pada optimasi fungsi 2 dimensi, *best* $f(\vec{x})$, rata-rata $f(\vec{x})$, standar deviasi $\boldsymbol{f}(\vec{x})$, dan rata-rata waktu komputasi yang dihasilkan oleh FOA lebih baik dari PSO.
2. PSO lebih cepat konvergen namun untuk 4 kriteria yang dibandingkan performa FOA masih lebih baik.

Tabel. 3.6 Perbandingan optimasi fungsi 4

Fungsi 4 – *Rastrigin*				
Global Minimum $f(\vec{x}) = 0, \vec{x} = \vec{0}$				
Uji Coba: 30 kali				
	FOA		**PSO**	
Dimensi	**10**	**30**	**10**	**30**
Best $f(\vec{x})$	0.004	0.080	2.985	39.912
Rata-rata $f(\vec{x})$	0.037	0.228	11.841	89.750
Standar deviasi	0.030	0.140	5.488	27.400
Rata-rata waktu komputasi (detik)	1.998	4.847	3.743	4.108



Gambar 3.12 Hasil optimasi fungsi 4 (10 dimensi)



Gambar 3.13 Hasil optimasi fungsi 4 (30 dimensi)

Fungsi *Rastrigin* mempunyai solusi global minimum $f(\vec{x}) = 0$, pada $\vec{x} = 0, i = 1,2,..,n$. Dari (Tabel 3.6) dan (Gambar 3.12) dan (Gambar 3.13) hasil optimasi fungsi *Rastrigin* dengan FOA dan PSO diperoleh perbandingan sebagai berikut:

1. Pada optimasi fungsi 10 dimensi dan 30 dimensi, *best* $f(\vec{x})$, rata-rata $f(\vec{x})$, standar deviasi $f(\vec{x})$, dan rata-rata waktu komputasi yang dihasilkan oleh FOA lebih baik dari PSO.
2. PSO lebih cepat konvergen namun masih jauh untuk mendekati nilai global minimum dibandingkan FOA.

BAB III
HASIL DAN PEMBAHASAN

Pada bab ini dibahas tentang penerapan algoritma *Forest Optimization Algortihm* (FOA) dalam menyelesaikan masalah optimasi fungsi nonlinear tanpa kendala. FOA dibandingkan dengan algoritma *Particle Swarm Optimization* (PSO) dalam menyelesaikan 4 fungsi uji dengan *search area* (domain) dan minimum global yang sudah diketahui. Perbandingan dilakukan dengan simulasi numerik. Hasil perbandingan digunakan untuk penarikan kesimpulan dari performa algoritma FOA dalam menyelesaikan masalah optimasi fungsi nonlinear tanpa kendala.

3.1 *Forest Optimization Algortihm* (FOA)

FOA algoritma evolusioner yang diilhami oleh beberapa pohon di hutan yang bisa bertahan selama beberapa dekade, sementara pohon lainnya hanya bisa hidup untuk jangka waktu terbatas. Metode ini diusulkan oleh Ghaemi dan Derakhshi pada tahun 2014.

FOA dimulai dengan inisialisasi populasi pohon awal, masing-masing pohon mewakili suatu potensi solusi dari masalah, selain itu setiap pohon memiliki jumlah variabel dan umur. Pada awalnya usia pohon diatur ke 0. Selanjutnya penebaran benih lokal akan menghasilkan pohon baru (muda) dari pohon sebelumnya dengan umur 0 dan tambahkan pohon baru ke hutan. Kemudian, semua pohon, kecuali yang baru dihasilkan, usia mereka bertambah sebesar 1. Selanjutnya, ada kontrol terhadap populasi pepohonan di hutan dan beberapa pohon akan dihilangkan dari hutan dan mereka akan membentuk populasi kandidat untuk pembenihan tahap global.

Pada tahap pembenihan global menambahkan beberapa solusi potensial baru ke hutan untuk menyingkirkan optimum lokal. Lalu, pohon-pohon di hutan digolongkan menurut nilai terbaik dan pohon dipilih sebagai pohon terbaik dan usianya diatur ke 0 agar terhindar penuaan dan setelah itu mengeluarkan pohon terbaik dari hutan (karena tahap pembenihan setempat meningkatkan umur semua pohon termasuk umur pohon terbaik). Tahapan ini akan berlanjut sampai kriteria pemberhentian terpenuhi. Berikut adalah algoritma dari FOA pada Gambar 3.1.

DAFTAR PUSTAKA

Brown, G.W. 1982. Standard Deviation, Standard Error Which 'Standard' Should We Use?. *American Journal of Diseases of Children.***136**:937-941.

Chen, R.M. dan Shih, H.F. 2013. Solving University Course Timetabling Problems Using Constriction Particle Swarm Optimization with Local Search. *Article Algorithms*. **6**: 227-244.

Dawkins, P. 2010. *Calculus III-Notes*. Texas:Lamar University.

Dianto, I. N. 2009. *Fuzzy Adaptive Turbulence Particial Swarm Optimation (FATPSO) untuk Masalah Optimasi Fungsi Nonlinear*. Skripsi. Malang: Universitas Brawijaya.

Foulds, L.R. 1984. *Combinatorial Optimization for Graduates*. New york: Springer-Verlag.

Ghaemi, M., dan Derakhshi, M. 2014. Forest Optimization Algorithm. *Expert System with Applications*. **41**: 6676-6687.

Green, D. S. 1983. *The Efficacy of Dispersal in Relation to Safe Site Density*. Oecologia. **56**: 356–358.

Haupt, R.L., dan Haupt, S.E. 2004. *Practical Genetic Algorithms. 2nd Edition*. New Jersey: Wiley-Interscience.

Jamil, M., dan Yang, X. 2013. A Literature Survey of Benchmark Functions for Global Optimization Problems, *Int. Journal of Mathematical Modelling and Numerical Optimization*. **4**: 150–194.

Jong, K. A. D. 2006. *Evolutionary Computation. A Unified Approach*. Cambrigde, MA, USA: MIT Press.

Kennedy, J., dan Eberhart, R. 1995. Particle Swarm Optimization. *Proceedings of IEEE International Conference on Neural Networks*. **4**:1942–1948.

Kenny, V., Nathal, M., dan Saldana. S. 2014. *Heuristic Algorithm*.

Luknanto, D. 2000. *Pengantar Optimasi Nonlinear*. Yogyakarta: Universitas Gadjah Mada.

Molga M., dan Smutnicki, C. 2005. *Test Functions for Optimization Needs*.Available:http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf. (diakses pada 5 Maret 2018).

Pan, H., Wang. L., dan Liu. B. 2006. Particle Swarm Optimization for Function Optimization in Noisy Environment. *Applied Mathematics and Computation*. **181**: 908-919.

Pham, D., dan Karaboga, D. 2000. *Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*. London: Springer-Verlag.

Rajabioun, R. 2011. Cuckoo Optimization Algorithm. *Applied Soft Computing*. **11**: 5508–5518.

Rao, S. S. 2009. *Engineering Optimization Theory and Practice*. John Wiley & Sons, Inc., Hoboken, New Jersey

Seckiner, S.U., Eroglu, Y., Emrullah, M., dan Dareli, T. 2013. Ant Colony Optimization for Continuous Functions by Using Novel Pheromone Updating. *Applied Mathematics and Computation*. **219**: 4163-4175.

Sivanandam, S. N., dan Deepa, S. N. 2008. *Introduction to Genetic Algorithms*. Berlin, Heidelberg: Springer-Verlag.

Taha, H.A., 2007. *Operations Research: An Introduction*. New Jersey: Pearson Education.

Weise, T. 2008. Global Optimization Algorithms: *Theory and Application. 2nd Edition*.

Yadav, R., dan Ahmad, W. 2013. Benchmark Function Optimization Using Genetic Algorithm. *Journal of Engineering, Computers & Applied Sciences*. **2**:2319-5606.