Majalah Ilmiah Matematika dan Statistika Volume 19 Nomor 1, 2019, 29 – 38
https://jurnal.unej.ac.id/index.php/MIMS/index ISSN 1411-6669

PERBANDINGAN ALGORITMA *PARTICLE SWAR*M OPTIMIZATION (PSO) DAN ALGORITMA *GLOWWORM SWARM OPTIMIZATION* (GSO) DALAM PENYELESAIAN SISTEM PERSAMAAN NON LINIER

(Comparison of *Particle Swar*m Optimization (PSO) and *Glowworm Swarm Optimization* (GSO) Algorithms in Solving Non Linear Equation System)

Ana Ulul Azmi, Rusli Hidayat, M. Ziaul Arif

Jurusan Matematika, Fakultas MIPA, Universitas Jember
Jl. Kalimantan 37 Jember 68121, Indonesia
E-mail: ziaul.fmipa@unej.ac.id

Abstract. Non-linear equation system is one of the mathematics problems which difficult to solve. Several methods have been introduced to solve the problems. Newton-Raphson method is the most common and widely used as the basis for evolving the latest numerical methods. However, this method requires the derivative of each equation with respect to every variable when calculating the Jacobian. Naturally, obtaining the derivative is challenging in certain cases. In addition, it needs a proper initial value to obtain the converged solution. Therefore, the new technique with a simple random initial value is urgently needed. In this study, it is shown the implementation of the two metaheuristic optimization methods, including Particle Swarm Optimization (PSO) and the Glowworm Swarm Optimization (GSO) to estimate the solution of a non-linear equation system. Several examples of non-linear equation system were used for evaluating and testing the performance and the accuracy of both algorithms. In this simulation, the results show that PSO converged to the exact solution (global optimum) better than Glowworm Swarm Optimization (GSO).

Keywords: Non-Linear Equation Systems, Particle Swarm Optimization (PSO), Glowworm Swarm Optimization (GSO)
MSC2010: 90B05

1. Pendahuluan

Kajian matematika yang ada, salah satunya yaitu sistem persamaan. Sistem persamaan ada dua, yaitu sistem persamaan linier dan sistem persamaan non linier. Permasalahan yang sulit untuk diselesaikan secara analitik yaitu sistem persamaan non linier, karena merupakan kumpulan dari suatu kalimat matematika terbuka yang variabel berderajat lebih dari satu atau mengandung nilai fungsi non linier, seperti *log, trigonometri*, *ekponensial*, dan lain sebagainya. Sistem persamaan non linier yang seringkali muncul bersifat kompleks, sehingga sulit diselesaikan menggunakan metode analitik, akan tetapi memungkinkan untuk diselesaikan dengan metode numerik. Algoritma metode numerik

yang digunakan untuk menyelesaikan persamaan non linier cukup banyak. Penelitian tentang menyelesaikan sistem persamaan non linier dengan membandingkan Metode Newton-Raphson dan Metode Jacobian sudah banyak dilakukan. Kesimpulan yang didapat pada penelitian ini Metode Newton-Raphson lebih baik untuk menyelesaikan sistem persamaan non linier daripada Metode Jacobian[9]. Metode Newton-Raphson, meskipun terbilang baik namun masih saja terdapat beberapa kekurangan. Kekurangan Metode Newton-Raphson yang lain yaitu metode ini harus menghitung turunan dari fungsi $f(x_n)$, sedangkan tidak semua fungsi dapat dicari turunannya dengan mudah. Selanjutnya pada referensi [1], pengembangan metode numerik yang telah diteliti memiliki tingkat konvergensi yang jauh lebih baik dari metode Newton-Raphon. Akan tetapi, beberapa dekade terakhir *Computational intelligent* banyak digunakan dalam penyelesaian permasalahan matematika.

Computational intelligent adalah solusi yang dapat mengatasi hal-hal yang sulit diselesaikan secara analitik. *Computational intelligent* adalah riset penelitian yang digunakan dalam bidang teknik optimasi berdasarkan perhitungan cerdas dari suatu langkah yang terstruktur[6]. Algoritma yang tergabung dalam *computational intelligent* misalnya *Genetic Algoritma* (GA), *Ant Colony Optimization* (ACO), *Artifical Bee Colony Optimization* (ABCO), *Cat Swarm Optimization* (CSO), *Cockroach Swarm Optimization Algorithm* (CSOA), *Glowworm Swarm Optimization* (GSO), *Particle Swarm Optimization* (PSO) [4]. Beberapa penerapan *Computational intelligent* dalam penyelesaian permasalahan matematika terutama persamaan non liner telah dilakukan oleh beberapa peneliti pada referensi [2] dan [3]. *Computational intelligent* jenis *swarm intelligent* lebih popular dalam menyelesaikan permasalahan optimasi. *Swarm intelligent* memiliki inspirasi wilayah penelitian dari kawanan hewan atau serangga. Metode *Particle Swarm Optimization* (PSO) terinspirasi oleh perilaku sosial dari binatang, seperti sekumpulan burung dalam suatu gerombolan yang sudah diterapkan pada beberapa hal seperti pada [5]. Kelebihan dari metode PSO yaitu kecepatan dalam menyelesaikan suatu permasalahan optimasi lebih cepat. Algoritma lain yang efektif dalam persoalan optimasi yaitu Glowworm Swarm Optimization (GSO). Metode *Glowworm Swarm Optimization* (GSO) yaitu metode yang berdasarkan pada perilaku cacing bercahaya.

Berdasarkan penjelasan diatas, penelitian tentang penerapan dari Algoritma *Particle Swarm Optimization* (PSO) dan Algoritma *Glowworm Swarm Optimization* (GSO) dalam penyelesaian Sistem Persamaan Non linier menarik untuk dilakukan. Peneliti akan membandingkan solusi dan hasil iterasi untuk mendapatkan hasil yang konvergen antara Algoritma *Particle Swarm Optimization* (PSO) dan Algoritma *Glowworm Swarm Optimization* (GSO).

Sistem Persamaan Non linier

Sistem persamaan non linier merupakan sistem persamaan yang mempunyai beberapa persamaan non linear yang kompleks dimana persamaan tersebut sulit dipecahkan secara

analitik. Apabila suatu persamaan sulit diselesaikan dengan metode analitik, maka persamaan tersebut dapat diselesaikan menggunakan metode numerik [8].

Bentuk umum sistem persamaan non linier dapat ditulis sebagai berikut:

$$\begin{cases} f_1(x_1, x_2, x_3, \dots, x_n) = b_1 \\ f_2(x_1, x_2, x_3, \dots, x_n) = b_1 \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ f_n(x_1, x_2, x_3, \dots, x_n) = b_n \end{cases}$$

Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) mensimulasikan perilaku dari sekelompok burung atau ikan. Misalnya, sekelompok burung sedang terbang secara acak mencari makanan di sebuah area tetapi hanya ada satu potong makanan pada area tersebut. Semua burung tidak mengetahui dimana lokasi makanan yang sebenarnya, hanya saja mereka dapat mengetahui seberapa jauh letak makanan yang ada pada setiap pencarian. Strategi terbaik yang dapat digunakan untuk mencari makanan adalah dengan cara mengikuti burung yang lokasinya paling deka dengan makanan tersebut. Satu burung menyatakan satu solusi di dalam ruang masalah dan burung tersebut dipresentasikan sebagai "partikel" dalam Algoritma *Particle Swarm Optimization* (PSO). Semua partikel memiliki nilai *fitness* yang yang akan dioptimalkan, dan memiliki kecepatan (*velocity*) setiap partikel berpindah dengan kecepatan yang diadaptasi dari daerah pencarian dan disimpan sebagai posisi terbaik yang pernah dicapai. Sehingga dari skenario kumpulan burung tersebut yang kemudian digunakannya untuk memecahkan masalah optimasi[6]. Nilai fitness disebut p_{best}. Ketika partikel mengambil semua populasi sebagai tetangga topologinya, posisi terbaik adalah global terbaik yang disebut g_{best}. Melalui p_{best} dan g_{best}, partikel memperbarui diri untuk menghasilkan generasi berikutnya dari kawanan.

Partikel dalam Algoritma *Particle Swarm Optimization* (PSO) merupakan solusi potensial untuk masalah ini, dan kawanan terdiri dari partikel P. Setiap patikel p dapat direpresentasikan melalui vektor n-dimensi: vektor pertama didenisikan sebagai $C_p^t = (C_{p1}^t, C_{p2}^t, \dots, C_{pn}^t)$ dengan $p = (1,2, \dots, pn)$ yang menunjukkan posisi partikel p dalam ruang pencarian di iterasi t. Vektor kedua dituliskan $V_p^t = (v_{p1}^t, v_{p2}^t, \dots, v_{pn}^t)$ yang merepresentasikan kecepatan pergerakan partikel p. Vektor ketiga dituliskan $P_{best_p}^t = (p_{best_{p1}}^t, p_{best_{p2}}^t, \dots, p_{best_{pn}}^t)$ menunjukkan posisi terbaik dari partikel ke-p dan vektor terakhir dituliskan $G_{best_p}^t = (g_{best_{p1}}^t, g_{best_{p2}}^t, \dots, g_{best_{pn}}^t)$ yang merepresentasikan posisi terbaik secara global dalam kawanan sampai iterasi ke-t. Kawanan diperbarui oleh persamaan berikut:

$$\begin{aligned} v_{pn}^{t+1} &= \theta v_{pn}^t + k_1 \times rand1(k_1)\left(p_{best_{pn}}^t - x_{pn}^t\right) + k_2 \times rand2(k_2)\left(g_{best_{pn}}^t - x_{pn}^t\right) \\ C_{pn}^{t+1} &= C_{pn}^t + v_{pn}^{t+1} \end{aligned} \tag{1}$$

θ merupakan nilai bobot inersia untuk mengurangi kecepatan. Biasanya nilai θ dibuat sedemikian hingga membuat hubungan iterasi dan kecepatan berbanding terbalik. Hal ini berarti semakin meningkat jumlah iterasi, kecepatan gerak partikel akan semakin mengecil. Nilai ini bervariasi secara linier dalam rentang menurun 0.9 hingga 0.4. Nilai bobot inersia yang tinggi menambah porsi pencarian global (global exploration), sedangkan nilai yang rendah lebih menekankan pencarian lokal (local search). Untuk tidak terlalu menitik beratkan pada salah satu bagian dan tetap mencari area pencarian yang baru dalam ruang berdimensi tertentu, maka perlu dicari nilai bobot inersia (θ) yang secara imbang menjaga pencarian global dan lokal. Untuk mencapai itu dan mempercepat konvergensi, hubungan terbalik antara bobot inersia dan iterasi ditulis dengan formula:

$$\theta = \theta_{max} - Iterasi \frac{\theta_{max} - \theta_{min}}{Maksimum\ Iterasi}$$

dengan θ_{max} dan θ_{min} masing-masing adalah nilai maksimum dan nilai minimum bobot inersia, i_{max} adalah jumlah iterasi maksimum yang digunakan dan i adalah iterasi yang sekarang. Biasanya digunakan nilai θ_{max} =0,9 dan θ_{min} =0,4. k_1 dan k_2 adalah parameter, $rand1(k_1)$ dan $rand1(k_2)$ adalah bilangan acak yang berdistribusi seragam dalam selang [0,1]. Koefisien k_1 dan k_2 adalah konstanta positif untuk mengontrol seberapa jauh sebuah partikel akan bergerak dalam setiap iterasi. Nilai-nilai yang rendah memungkinkan partikel untuk menjelajah jauh dari daerah sasaran, sementara nilai-nilai yang tinggi memungkinkan untuk menjelajah daerah sasaran yang sudah terlewati pada iterasi sebelumnya. Hal ini efektif untuk pencarian solusi optimum global. Biasanya adalah kedua nilai 2.0, meskipun pemberian nilai berbeda untuk k_1 dan k_2 dapat dilakukan untuk mengetahui kinerja algoritma.

Konstanta v_{max} digunakan untuk membatasi kecepatan dari partikel v_{pn}^t dan meningkatkan ruang solusi pencarian. Ketika v_{max} besar, kecepatan partikel juga besar, hal ini adalah kondusif untuk pencarian global, yang mungkin terbang melalui solusi optimal. Ketika v_{max} kecil, kecepatan partikel juga kecil, hal itu mengarah ke pencarian terbaik di wilayah tertentu, tetapi mudah untuk terjebak ke optimum lokal. Secara singkat, efisiensi pencarian tergantung pada v_{max}.

Setiap partikel bergerak dalam ruang pencarian dengan kecepatan sesuai dengan solusi terbaik sendiri dan solusi terbaik kelompok sebelumnya. Kecepatan v_{pn}^{t+1}, pada persamaan (1) terdiri dari tiga bagian yaitu bagian pertama adalah v_{pn}^t menunjukkan kecepatan partikel sebelumnya. Bagian kedua $k_1 \times rand1(k_1)$ menunjukkan proses penyelesaian dari pengalaman individu. Selanjutnya bagian ketiga $k_2 \times rand1(k_2)$ menunjukkan proses penyelesaian dari pengalaman partikel lain, yang mewakili berbagai informasi dan kerjasama sosial antar partikel. Keseimbangan antara bagian-bagian ini menentukan kinerja Algoritma *Particle Swarm Optimization* (PSO) [10].

Secara garis besar, Algoritma *Particle Swarm Optimization* (PSO) dapat dijabarkan sebagai berikut:

a. Inisialisasi posisi awal (C_i) dan kecepatan awal partikel (V_i), dengan $i = 1,2, \dots, S$ dan S adalah ukuran *swarm*.
b. Evaluasi nilai fungsi tujuan untuk setiap partikel $(f(C_i))$.
c. Tentukan p_{best} awal dan g_{best} awal.
d. Update kecepatan v^{t+1} dengan persamaan (1)
e. Update posisi individu baru C^{t+1} dengan persamaan (1)
f. Evaluasi kembali $f(C_i)$, jika $f(C_i) \leq f(p_{ibest})$ maka $p_{ibest} = C_i$, setelah mendapatkan p_{ibest} baru, maka didapatkan $f(p_{ibest})$ baru.
g. Jika iterasi sudah maksimum maka Algoritma berhenti, jika tidak maka kembali ke langkah (d).

Glowworm Swarm Optimization (GSO)

Algoritma *Glowworm Swarm Optomization* (GSO) merupakan pengembangan dari Algoritma *Ant Colony Optimization* (ACO) yang digambarkan sebagai karakter cacing bercahaya. Setiap individu agen (*glowworm)* memiliki jarak penglihatan untuk mengetahui keberadaan agen lain atau tetangga, yang disebut *local decision range*. *Local decision range* tergantung dari jumlah tetangga, ketika jumlah tetangga terlalu sedikit maka *local decision range* membesar untuk menemukan lebih banyak tetangga, sebaliknya *local decision range* akan mengecil apabila jumlah tetangga terlalu banyak. Nilai *luciferin* setiap agen dikaitkan dengan nilai fungsi objektif, sedangkan parameter nilai fungsi objektif dikaitkan dengan posisi dari setiap agen. Setiap agen selalu mengubah arah gerak sesuai dengan posisi tetangga yang dipilih dan hanya satu tetangga yang dipilih oleh setiap agen, yaitu tetangga yang memiliki nilai *luciferin* yang tertinggi diantara seluruh tetangga dari agen tersebut. Pada akhirnya, sebagian besar agen akan berkumpul di beberapa lokasi [7].

Pengkodean adalah kunci untuk menyelesaikan masalah pada *Glowworm Swarm Optomization* (GSO) untuk masalah optimasi. Prosesnya sebagai berikut : $\boldsymbol{x}_i^t$ adalah lokasi dari *Glowworm* i pada iterasi t dimana $\boldsymbol{x}_i^t = [x_{i1}^t, x_{i2}^t, \dots, x_{id}^t]$ (d adalah banyaknya dimensi) dan $\boldsymbol{y}_i^t$ adalah solusi *Glowworm* dengan $\boldsymbol{y}_i^t = [y_{i1}^t, y_{i2}^t, \dots, y_{id}^t]$ yang elemennya tergantung pada solusi dari masalah[6].

Langkah-langkah Algoritma *Glowworm Swarm Optomization* (GSO) adalah sebagai berikut:

a. Inisialisasi parameter-parameter *Glowworm Swarm Optomization* (GSO) antara lain: banyaknya *Glowworm* (n), nilai awal *luciferin* $(l_0 > 0)$, nilai awal *adaptive decision-range* $(r_d > 0)$, jarak antar *Glowworm* $(r_s > 0)$, tingkat *update* nilai *luciferin* $(\gamma > 0)$, tingkat *update dynamic decision-range* $(\beta > 0)$, batas ketetanggaan $(n_t > 0)$, *step-size* $(s > 0)$, tingkat disiplin nilai *luciferin* $(0 < \rho < 1)$, iterasi (t), nilai maksimum iterasi. Bangkitkan inisial *Glowworm* sebanyak n secara random.

b. Hitung nilai *fitness*.

$$fitness_i = \begin{cases} f(\boldsymbol{y}_\mathrm{i}), & untuk\ kasus\ maksimasi \\ \dfrac{1}{f(\boldsymbol{y}_\mathrm{i})}, & untuk\ kasus\ minimasi \end{cases}$$

c. Perbarui nilai *luciferin*, nilai *local decision-range*, dan lokasi *Glowworm* sebagai berikut:

$$\begin{aligned} l_i(t+1) &= (1-\rho)l_i(t) + \gamma(fitness_i) \\ N_i(t) &= \{j: \|X_j - X_i\| < r_d^i(t); l_i(t) < l_j(t)\} \\ p_{ij}(t) &= \frac{l_j(t) - l_i(t)}{\sum_{keN_i(t)} l_k(t) - l_i(t)} \end{aligned} \quad (2)$$

Setiap *Glowworm* i memilih satu tetangga j dengan maksimum kemungkinan $p_{ij}(t)$ dan berpindah ke tetangga j

$$X_i(t+1) = X_i(t) + s\left(\frac{X_j(t) - X_i(t)}{|X_j(t) - X_i(t)|}\right)$$

$$r_d^i(t+1) = min\left\{r_s, max\{0, r_d^i(t) + \beta(n_t - |N_i(t)|\}\right\}$$

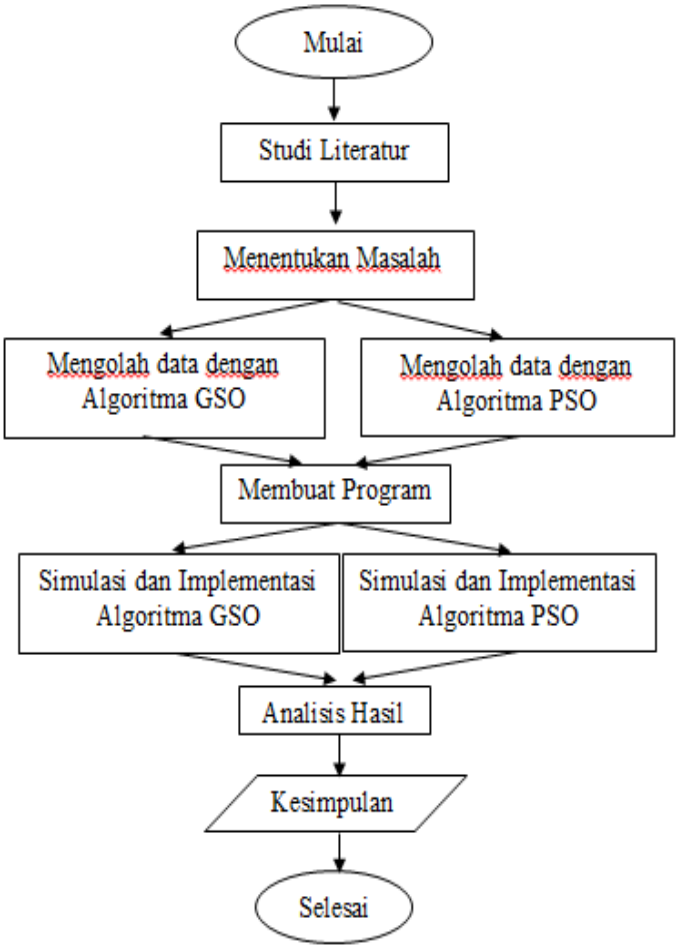
d. Jika nilai maksimum iterasi terpenuhi maka iterasi dihentikan, jika belum maka kembali ke langkah b.

2. Metodologi

Langkah-langkah penelitian yang akan dilakukan untuk menyelesaikan sistem persamaan non linier digambarkan dalam skema pada Gambar 1.

3. Hasil dan Pembahasan

Pada penelitian ini terdapat beberapa contoh sistem persamaan non linier dari referensi yang telah dirujuk. Hasil penyelesaian sistem persamaan non linier dari beberapa referensi tersebut akan dijadikan tolok-ukur perbandingan algoritma *Particle Swarm Optimization* dan *Glowworm Swarm Optimization*. Untuk menyelesaikan sistem persamaan non linier menggunakan algoritma *Particle Swarm Optimization*, peneliti menggunakan parameter $npop = 20, lb\ (batas\ atas) = 1,$ dan $ub(batas\ bawah) = -1$. Sedangkan untuk algoritma *Glowworm Swarm Optimization*, peneliti menggunakan parameter $V0(kecepatan\ awal) = 0, c1 = c2 = 1, t\_max = 0.9,$ dan $t\_min = 0.4$. Parameter untuk algoritma *Glowworm Swarm Optimization* yaitu $l0 = 1, r0 = 75, rs = 100,\ gamma = 0.8, beta = 0.09, nt = 8, s = 0.05,$ dan $rho = 0.8$. Berikut adalah beberapa contoh sistem persamaan non linier beserta grafik kekonvergenan nilai fungsinya dan banyaknya iterasi. Penyelesaian sistem persamaan non linear ditulis dalam bentuk fungsi fitness sebagai permasalahan minimasi sebagai berikut:



Gambar 1. Skema metode penelitian

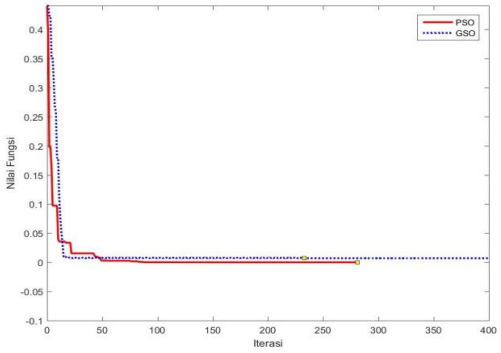
$$(p') = \begin{cases} abs\left(f_1(x_1, x_2, x_3, \dots, x_n)\right) + \\ abs\left(f_2(x_1, x_2, x_3, \dots, x_n)\right) + \\ \quad \vdots \qquad\quad \vdots \qquad\quad \vdots \\ +abs\left(f_n(x_1, x_2, x_3, \dots, x_n)\right) \end{cases} = 0$$

Contoh 1: Perhatikan sistem persamaan non linier berikut.

$$\cos(2x) - \cos(2y) - 0.4 = 0$$
$$2(\mathrm{y} - \mathrm{x}) + \sin(2\mathrm{y}) - \sin(2x) - 1.2 = 0$$

Tabel 1. Hasil penyelesaian spnl contoh 1

Metode	Nilai awal	Solusi
PSO	$x = 0.3562761185$ $y = 0.6521301239$	$x =.,1565200697$ $y = 0{,}4933763742$
GSO	$x = -0.1757942073$ $y = 0.1300070743$	$x = 0.1563927449$ $y = 0.4917706759$
NR	$x = 0$ $y = 0$	$x = 0.15$ $y = 0.49$



Gambar 2. Grafik nilai *fitness* SPNL contoh 1

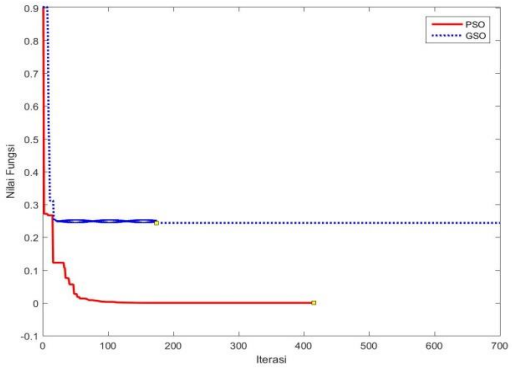
Gambar 2 merupakan *output* dari sistem persamaan contoh 1 dengan sumbu *x* merupakan banyaknya iterasi dan sumbu *y* merupakan nilai fungsi. Sistem persamaan non linier contoh 1 dengan dua variabel menggunakan maksimum iterasi sebanyak 400. Pada Gambar 2 untuk algoritma *Partcle Swarm Optimization* menunjukkan bahwa nilai $fungsi = 0$ atau mencapai nilai optimum pada iterasi ke-281 dan solusi yang dihasilkan adalah $x = 0{,}1565200697$ dan $y = -0{,}4933763742$ dengan waktu=24.5215 detik, sedangkan untuk algoritma *Glowworm Swarm Optimization* konvergen saat nilai $fungsi = 0{,}0064212840470351273$ pada iterasi ke-233 dan solusi yang dihasilkan $x = 0.1563927449$ dan $y = 0.4917706759$ dengan waktu=26.7442 detik.

Contoh 2: Perhatikan sistem persamaan non linier berikut.

$$x + \cos(xy) - z^2 - 1{,}1 = 0$$
$$x^2 - 10y - e^{xy} + 0{,}8 = 0$$
$$xz + y^2 - z - 0{,}3 = 0$$

Tabel 2. Hasil penyelesaian SPNL contoh 2

Metode	Nilai awal	Solusi
PSO	$x = -0.0872108122$ $y = 0.0438547305$ $z = -0.251569332$	$x = 0.2676238813$ $y = -0.0125036768$ $z = -0.4094121182$
GSO	$x = -0.0872108122$ $y = 0.0438547305$ $z = -0.251569332$	$x = -0.0420091244$ $y = -0.0199657728$ $z = -0.2508212616$
NR	$x = 0$ $y = 0$ $z = 0$	$x = 0.26756623$ $y = -0.0133904733$ $z = -0.409348541$



Gambar 3. Grafik nilai *fitness* SPNL contoh 2

Gambar 3 merupakan *output* dari sistem persamaan contoh 2 dengan sumbu *x* merupakan banyaknya iterasi dan sumbu *y* merupakan nilai fungsi. Sistem persamaan non linier contoh 2 dengan tiga variabel menggunakan maksimum iterasi sebanyak 700. Pada Gambar 3 untuk algoritma *Partcle Swarm Optimization* menunjukkan bahwa nilai $fungsi = 0$ atau mencapai nilai optimum pada iterasi *ke-415* dan solusi yang dihasilkan adalah $x = 0{,}2676238813$, $y = -0{,}0125036768$, dan $z = -0{,}4094121182$ dengan waktu=44.997 detik, sedangkan untuk algoritma *Glowworm Swarm Optimization* menunjukan nilai $fungsi = 0{,}24374750389040273$ pada iterasi ke-174 dan solusi yang dihasilkan $x = -0.0420091244$, $y = -0.0199657728$, dan $z = -0.2508212616$ dengan waktu = 47.2696 detik.

Dalam studi kasus ini, solusi contoh 1 dan 2 yang diselesaikan dengan PSO dan GSO menunjukkan bahwa PSO memiliki performa yang lebih baik dari pada GSO. perbedaan waktu pencapaian konvergensi solusi antara PSO dan GSO tidak begitu berarti pada penelitian ini selama hanya 3 detik selisih waktu perhitungannya. Jika dilihat dari Gambar 2 dan 3, dapat disimpulkan bahwa GSO mudah terjebak pada solusi local optimum. Hal ini dikarenakan setiap individu di GSO memiliki ruang gerak pencarian solusi yang terbatas hanya pada individu tetangga terdekat yang solusinya belum tentu optimal. Sedangkan pada PSO, individu bergerak sesuai dengan informasi dari individu optimal yang terpilih. Sehingga global optimal mudah dicapai.

4. Kesimpulan

Berdasarkan pembahasan kasus contoh diatas, dapat disimpulkan bahwa algoritma *Particle Swarm Optimization* memiliki performa yang lebih baik dari pada algoritma *Glowworm Swarm Optimization* dalam penyelesaian sistem persamaan non linier. Hal ini disebabkan karena algoritma *Particle Swarm Optimization* selalu konvergen ke global optimal dibandingkan algoritma *Glowworm Swarm Optimization* yang hanya konvergen pada local optimal seperti pada contoh 2. Konvergensi ke lokal optimum pada GSO dikarenakan *Glowworm* akan berpindah menuju ke tetangganya yang lebih baik yang

lebih banyak *luciferin*, tetapi posisi baru setelah berpindah tidak menjamin solusi menuju ke global optimal.

Daftar Pustaka

[1] Arif, M. Z, Farida, Y. dan Kamsyakawuni, A. (2017). "Penerapan Metode Newton-Cotes Open Form 5 Titik Untuk Menyelesaikan Sistem Persamaan Nonlinier." *Unnes Journal of Mathematics* 6(1): 102-107.

[2] Anwar, Z. (2016). Penerapan Gabungan Metode Zero Crossing Dan Virus Evolutionary Genetic Algorithm (Vega) Pada Penyelesaian Persamaan Non-Linier. *Skripsi*. Jember: Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

[3] Baihaki, N. A. (2016). Penerapan Algoritma Cat Swarm Optimization (CSO) Pada Penyelesaian Sistem Persamaan Non-linier. *Skripsi*. Jember: Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

[4] Chu, S. C., Tsai, P. W., dan Pan, J. S. (2006). "Computational Intelligence Based on the Behavior of Cats". *International Journal of innovative Computing, Information and Control.* 3 (1): 163-173.

[5] Erny, (2013). Optimasi Pola Penyusunan Barang Dalam Peti Kemas Menggunakan Algoritma Particle Swarm Optimization. *Skripsi*. Makasar : UNHAS.

[6] Gong, Q., Zhou, Y, dan Yang, Y. (2011). Artificial Glowworm Swarm Optimization Algoritma For Solving 0-1 Knapsack Problem. *Journal of Advanced Materials Research.* 143-144: 166-171.

[7] Krishnanand, K. N, Ghose D. (2008). Theorical foundations for rendezvous of glowworm-inspired agent swarms at multiple locations. *Journal of Robotics and Autonomous Sistem*. 56 (7): 549-569.

[8] Munir, R. (2003). *Metode Numerik*. Jakarta: Erlangga.

[9] Utami, N. N. R., Widana, I. N., dan Asih, N. M. (2013). Perbandingan Solusi Sistem Persamaan Non linier Menggunakan Metode Newton-Raphson dan Metode Jacobian. *E-Jurnal Matematika*. 2 (2): 11-17.

[10] Xu, S.H, Liu, J. P, Zhang, F. H, Wang, L., dan Sun, L. J. (2015). *A Combination of Genetic Algorithm and Particle Swarm Optimization for Vehicle Routing Problem With Time Windows*. Swithzerland: Licensee MDPI, B|asel, Switzerland.