Penerapan Flower Pollination Algorithm dengan Teknik Clustering dalam Penyelesaian Masalah Diophantine

Implementation of Flower Pollination Algorithm with Clustering Technique for Solving Diophantine Problems

Rahmat Karim[1], Salim[2]
[1] Program Studi Ilmu Komputer, Universitas Sembilanbelas November Kolaka
[1] Program Studi Matematika, Universitas Sembilanbelas November Kolaka

E-mail: [1]rahmatkarim.usn@gmail.com, [2]salimmath@usn.ac.id

Abstrak

Permasalahan *Diophantine* adalah suatu permasalahan yang diwakili persamaan atau sistem persamaan yang memerlukan bilangan bulat *non-negatif* sebagai solusi. Permasalahan ini banyak dijumpai di berbagai bidang termasuk *Computer engineering* seperti pengelolaan jaringan dan sinyal. Akan tetapi belum ada metode umum yang secara efektif dapat menyelesaikan permasalahan *Diophantine.* Tujuan utama dalam penelitian ini adalah untuk melakukan penyesuaian metode FPAC agar FPAC tidak hanya dapat digunakan pada permasalahan *Multimodal* tetapi juga dapat dijadikan sebagai alternatif pada permasalahaan *Diophantine*. Transformasi persamaan ataupun sistem persamaan ke dalam bentuk fungsi optimasi dan transformasi output bilangan real ke bilangan bulat pada setiap tahapan algoritma merupakan kunci utama FPAC dalam menyelesaikan permasalahan *Diophantine*. Hasil penelitian ini menunjukan bahwa FPAC dapat menemukan seluruh solusi dari persamaan *Diophantine* baik persamaan yang memiliki jumlah variabel dan pangkat yang berbeda maupun persamaan dalam bentuk eksponensial. FPAC juga dapat menemukan seluruh solusi yang tersedia pada sistem persamaan *Diophantine* baik yang berdimensi rendah (kasus 1) maupun dimensi tinggi (kasus 2 dan 3). Secara umum, FPAC terbukti efektif dalam menyelesaikan permasalahan *Diophantine* baik dalam bentuk persamaan maupun sistem persamaan yang memiliki solusi tunggal maupun jamak dalam sekali *running*.

Kata kunci: Flower Pollination Algorithm; Teknik Clustering; Permasalahan *Diophantine*

Abstract

The Diophantine problem is a problem represented by an equation or system of equations that require a non-negative integer as a solution. This problem is often found in various fields, including computer engineering, such as network and signal management. However, there is no general method that can effectively solve the Diophantine problem. The main objective of this research is to adjust the FPAC method so that FPAC can be used not only for multimodal problems but also as an alternative to Diophantine problems. Transformation of equations or systems of equations into the form of optimization functions and transformation of real output numbers to integers at each stage of the algorithm is the primary key to FPAC in solving Diophantine problems. The results of this study indicate that FPAC can find all the solutions of the Diophantine equation, both equations that have a different number of variables and exponentials, as well as equations in exponential form. FPAC can also find all available solutions to the Diophantine equation system, both low-dimensional (case 1) and high-dimensional (cases 2 and 3). FPAC has proven effective in solving Diophantine problems in equations and systems of equations with single and multiple solutions in a single run..

Keywords: Flower Pollination Algorithm; Clustering technique; The Diophantine Problem

1. PENDAHULUAN

Pada dasarnya setiap permasalahan rumit di dunia nyata dapat disimplifikasi melalui pendekatan model matematika. Akan tetapi, model matematika tersebut hanya merepresentasikan masalah tetapi tidak serta-merta menginformasikan letak titik-titik *equilibrium* dari model. Titik-titik *equilibrium* adalah representasi solusi dari model yang umumnya tidak mudah ditemukan nilainya, baik disebabkan oleh kerumitan model maupun keunikan dari solusi [1].

Salah satu permasalahan yang sering ditemukan dalam bidang *engineering* adalah permasalahan *Diophantine*. Permasalahan *Diophantine* memainkan peran yang sangat penting dalam bidang-bidang khusus seperti kriptografi, faktorisasi bilangan bulat, teori bilangan, geometri aljabar, teori kontrol, analisis ketergantungan data pada super komputer, komunikasi dan lain-lain [2][3][4]. Permasalahan *Diophantine* adalah permasalahan dalam bentuk persamaan ataupun sistem persamaan yang memerlukan bilangan bulat positif sebagai solusi [2]. Karakteristik solusi dalam bentuk bilangan bulat positif inilah yang menempatkan permasalahan *Diophantine* menjadi permasalahan yang menarik dan menantang. Secara matematik, jika x adalah bilangan real yang memenuhi suatu persamaan atau sistem persamaan maka x adalah solusi yang dimaksud. Akan tetapi x tersebut tidak akan dianggap sebagai solusi pada permasalahan *Diophantine*.

Sebenarnya para matematikawan sejak dahulu telah menghadapi masalah *Diophantine* secara analitik dengan menggunakan *Euclidean Algorithm* (EA) yang bekerja secara sistematis ataupun rekursif [5]. Akan tetapi metode tersebut hanya bekerja dengan baik pada keadaan-keadaan tertentu saja, misalnya pada *identitas atau lemma Bezout*. Disisi lain, permasalahan *Diophantine* tidak hanya dalam bentuk persamaan linear tetapi juga dapat direpresentasikan oleh persamaan ataupun sistem persamaan non-linear bahkan eksponensial yang umumnya memiliki lebih dari satu solusi.

Pemanfaatan algoritma yang berbasis *stokastik* adalah hal yang perlu dilakukan untuk mengatasi masalah yang umum dijumpai jika menggunakan algoritma yang berbasis *gradient*. Algoritma *metaheuristic* yang merupakan salah satu jenis dari algoritma yang berbasis *stokastik* memiliki dua karakteristik utama yaitu *Diversifikasi* dan *Intensifikasi*. Diversifikasi atau *exploration* bermakna bahwa algoritma menyebarkan sejumlah calon solusi yang berbeda secara acak untuk mengeksplorasi daerah pencarian pada skala global, sedangkan *Intensifikasi* atau *exploitation* bermakna bahwa algoritma fokus melakukan pencarian pada skala lokal dengan memanfaatkan informasi yang diperoleh. Kombinasi dari kedua karakteristik tersebut yang menjamin setiap algoritma *metaheuristik* baik yang bersifat *trajectory-based* maupun *population-based* untuk selalu *konvergen* ke solusi optimal [6] .

Beberapa Algoritma *metaheuristic* seperti *Genetic algorithm* (GA), *Differential Evolution* (DE), *Cuckoo search* (CS), dan *Flower Pollination Algorithm* (FPA) merupakan algoritma yang dikembangkan khusus untuk menyelesaikan permasalahan optimasi global. Dalam hal penerapan, algoritma tersebut telah banyak dijumpai termasuk di bidang *engineering.* Akan tetapi algoritma-algoritma tersebut perlu penyesuaian atau pengembangan agar dapat diterapkan pada permasalahan *Diophantine*.

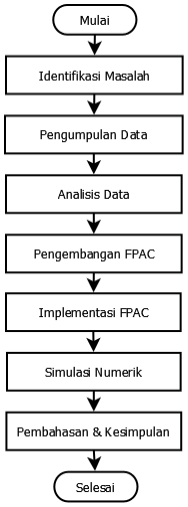
Sibby Abraham menggunakan *Ant Colony Optimization* (ACO) untuk menemukan solusi numerik dari persamaan *Diophantine*. ACO diterapkan pada persamaan *Diophantine* dengan jumlah pangkat dan variabel yang berbeda. Hasil eksperimen menunjukan bahwa efektivitas ACO dalam menyelesaikan persamaan *Diophantine* cukup baik [7]. Oscar Peres, menggunakan *Particle Swarm Optimization* (PSO) untuk menemukan solusi numerik dari sistem persamaan *Diophantine*. PSO diterapkan pada sistem persamaan *Diophantine* dengan jumlah persamaan pembentuk sistem yang berbeda. Hasil eksperimen menunjukan bahwa PSO dapat menemukan solusi dari sistem persamaan *Diophantine* yang diujikan [5]. Meskipun kedua pendekataan di atas dapat menemukan solusi dari Persamaan dan Sistem Persamaan *Diophantine* akan tetapi masih dilakukan *running* berulang kali untuk mendapatkan seluruh solusi dari permasalahan *Diophantine* yang diujikan.

Dmitry Zaitsev menyelesaikan sistem persamaan *Diophantine* linear dengan memanfaatkan aplikasi *ParAd*. Aplikasi tersebut dikembangkan untuk mendukung komputasi paralel yang menerapkan konsep MPI dan OpenMP. Hasil simulasi numerik menunjukan bahwa metode yang diusulkan 10 kali lebih cepat dibandingkan dengan aplikasi *PardAd* standar [8]. Sementara Yui-Kwong Man melakukan pendekatan *Top-Down* untuk menyelesaikan persamaan *Diophantine* linear. Metode yang diusulkan ini lebih baik dari pendekatan *Bottom-Up*. Selain itu juga pendekatan *Top-Down* dapat dijadikan sebagai salah satu alternatif pada persamaan *Diophantine* linear selain *Euclidean Algorithm* (EA) ataupun *Extended Euclidean Algorithm* (EEA) yang telah diterapkan pada berbagai disiplin ilmu [9].

Berdasarkan hasil penelusuran literatur di atas, ditemukan bahwa belum ada metode umum yang dapat menyelesaikan persamaan ataupun sistem persamaan *Diophantine* baik yang linear maupun non-linear dalam sekali *running*. Oleh karena itu penelitian ini perlu dilakukan sebagai gagasan alternatif pada permasalahan *Diophantine*. Sebagai algoritma optimasi yang berbasis metaheuristik, *Flower Pollination Algorithm* (FPA) telah banyak diterapkan pada berbagai disiplin ilmu [10][11][12]. Selain itu juga, FPA telah dimodifikasi dengan Teknik *Clustering* dan terbukti efektif menemukan semua solusi dari masing-masing *benchmark* permasalahan multimodal dalam sekali *running* [13]. Oleh karena itu, pada penelitian ini diimplementasikan hasil modifikasi *Flower Pollination Algorithm* dengan Teknik *Clustering* (FPAC) untuk menyelesaikan permasalahan *Diophantine*.

2. METODE PENELITIAN

Secara prosedural penelitian ini dibagi atas beberapa tahap yaitu: identifikasi masalah, pengumpulan data, analisis data, pengembangan FPAC, implementasi FPAC, simulasi numerik, pembahasan dan kesimpulan. Pada tahap identifikasi masalah diperoleh informasi bahwa masalah *Diophantine* banyak diterapkan diberbagai disiplin ilmu akan tetapi belum ada metode umum untuk menyelesaikan permasalahan *Diophantine*. Pengumpulan dan Analisis data dilakukan untuk mendukung tahapan pengembangan metode FPAC agar sesuai dengan masalah *Diophantine*. Tahapan pengembangan FPAC perlu dilakukan karena FPAC awalnya dimodifikasi untuk melokalisir semua solusi pada permasalahan Multimodal. Permasalahan Multimodal dalam bentuk fungsi sementara Permasalahan *Diophantine* dalam bentuk persamaan, sehingga perlu ada transformasi fungsi objektif. Selain itu juga permasalahan *Diophantine* memerlukan bilangan bulat non-negatif sebagai solusi sehingga perlu ada penyesuaian pada beberapa tahapan dalam FPAC. FPAC yang telah disesuaikan dengan permasalahan *Diophantine* kemudian dikonstruksi dalam bentuk program untuk mendukung tahapan simulasi numerik. Tahapan simulasi numerik difokuskan pada *benchmark* persamaan *Diophantine* dan sistem persamaan *Diophantine*.



Gambar 1. Alur penelitian

2.1 Persamaan dan Sistem Persamaan Diophantine

Dalam sejarahnya, nama *Diophantine* adalah bentuk apresiasi terhadap *Diophantus* sebagai orang yang pertama kali mempelajari persamaan *Diophantine* sekaligus meletakan risalah pertama kali tentang Aljabar dalam karya Buku *Arithmetica*. Pada praktiknya masalah *Diophantine* ternyata banyak ditemukan di kehidupan nyata seperti dalam sistem diskrit, pengecekan model, Bahasa formal dan logika automata, pemrograman, kriptografi, jaringan dan pengolahan sinyal [8].

Secara prinsip masalah *Diophantine* berkaitan erat dengan solusi persamaan atau sistem persamaan dalam domain bilangan bulat atau bilangan rasional. Yang membedakan adalah masalah *Diophantine* mensyaratkan bilangan bulat *non-negatif* sebagai solusi. Sebagai sebuah sistem, masalah *Diophantine* dapat terbentuk oleh hanya satu persamaan atau beberapa persamaan yang terdiri dari satu atau beberapa variabel.

Misalkan, persamaan (1) adalah sistem persamaan *Diophantine*:

$$\begin{aligned} h_1(x_1, x_2, \cdots, x_n) &= 0 \\ h_2(x_1, x_2, \cdots, x_n) &= 0 \\ &\vdots \\ h_n(x_1, x_2, \cdots, x_n) &= 0 \end{aligned} \tag{1}$$

maka persamaan (1) dapat ditulis dalam bentuk vektor:

$$\boldsymbol{h}(\boldsymbol{x}) = 0 \tag{2}$$

Dimana $\boldsymbol{h} = (h_1, h_2, \cdots, h_n)^T$ dan $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)^T$. Solusi atau akar dari sistem persamaan *Diophantine* (2) adalah $\boldsymbol{x}^* = (x_1^*, x_2^*, \cdots, x_n^*)^T$ yang merupakan bilangan bulat non negatif dalam domain tertentu jika memenuhi $\boldsymbol{h}(\boldsymbol{x}^*) = 0$.

2.2 Masalah Optimasi

Masalah optimasi secara matematis dapat dituliskan dalam bentuk

$$\begin{aligned} max/\min \quad & f_i(\boldsymbol{x}) \quad , i = 1,2, \cdots, n \\ subject\ to\ & h_j(\boldsymbol{x}) = 0 \quad , j = 1,2, \cdots, m \\ & g_k(\boldsymbol{x}) \leq 0 \quad , k = 1,2, \cdots, r \end{aligned} \tag{3}$$

dimana $f_i(\boldsymbol{x})$, $h_j(\boldsymbol{x})$ dan $g_k(\boldsymbol{x})$ adalah fungsi dari $\boldsymbol{x} = (x_1, x_2, \cdots, x_d) \in \mathbb{R}^d$ dan vektor $\boldsymbol{x}$ adalah variabel keputusan [6].

Dengan menggunakan konsep yang dikemukakan oleh Aggarwal [14] maka masalah sistem persamaan *Diophantine* (persamaan 2) dapat ditransformasi ke dalam bentuk masalah optimasi.

$$\begin{aligned} H(\boldsymbol{x}) &= H(x_1, x_2, \cdots, x_n) \\ &= \frac{1}{1 + \sum_{i=1}^{n} |h_i(x_1, x_2, \cdots, x_n)|} \\ &= \frac{1}{1 + \sum_{i=1}^{n} |h_i(\boldsymbol{x})|} \end{aligned} \tag{4}$$

Jika ditemukan $\boldsymbol{x}^*$ yang menyebabkan $\boldsymbol{H}(\boldsymbol{x}^*) = 1$, maka $\boldsymbol{x}^*$ adalah solusi maksimum global dari $G(\boldsymbol{x})$. Hal ini sekaligus memberikan informasi bahwa $\boldsymbol{x}^*$ adalah solusi atau akar-akar dari persamaan (2) yang memenuhi sistem $h_i(\boldsymbol{x}^*) = 0$ untuk $i = 1,2, \cdots, n$.

2.3 Flower Pollination Algorithm (FPA)

Flower pollination Algorithm (FPA) adalah algoritma metaheuristik berbasis populasi yang masuk dalam kelompok *swarm intelligence*. Seperti algoritma metaheristik lainnya, FPA juga terinspirasi dari Alam yaitu proses penyerbukan. Algoritma ini dikembangkan Xin-She Yang pada tahun 2012 untuk menyelesaikan permasalahan optimasi global[15].

Sebagai sebuah metode yang terinspirasi dari alam, karakteristik dari FPA berasal dari empat asumsi berikut, yaitu:

1. Penyerbukan global (*Global Pollination*) adalah proses penyerbukan di alam yang dapat terjadi secara biotik atau secara silang dengan bantuan pollinator sebagai agen utama. Pergerakan pollinator diasumsikan mengikuti gerak *Levy*.
2. Penyerbukan lokal (*local pollination*) adalah proses penyerbukan di alam yang dapat terjadi secara abiotik atau secara sendiri.
3. Pollinator sebagai agen utama memiliki sifat cerdas yang dapat mengenali bunga sejenis sehingga menjamin terjadinya reproduksi secara proporsional pada penyerbukan Global.
4. *Switch probability* $p \in [0,1]$ mengatur terjadinya proses penyerbukan global atau proses penyerbukan lokal.

Secara matematis, penyerbukan Global dan Penyerbukan Lokal, mengikuti persamaan (5) dan (6) berikut:

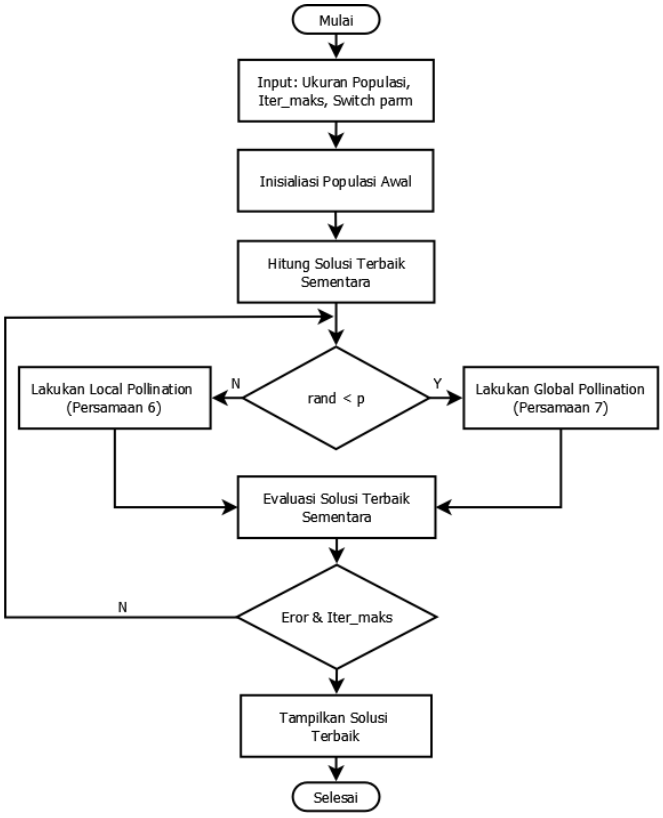
$$\boldsymbol{x}_i{}^{t+1} = \boldsymbol{x}_i{}^{t} + \gamma \boldsymbol{L}(\lambda)(\boldsymbol{g}_* - \boldsymbol{x}_i{}^{t}) \quad (5)$$

$$\boldsymbol{x}_i{}^{\mathrm{t+1}} = \boldsymbol{x}_{\mathrm{i}}{}^{\mathrm{t}} + \epsilon\left(\mathbf{x}_{\mathrm{j}}{}^{\mathrm{t}} - \boldsymbol{x}_{\mathrm{k}}{}^{\mathrm{t}}\right) \quad (6)$$

Keterangan:

$\boldsymbol{x}_i{}^{t}$: solusi ke i pada iterasi ke t
$\boldsymbol{g}_*$: adalah solusi terbaik sementara pada setiap iterasi t
$\boldsymbol{L}$: ukuran langkah pollinator (step size) yang berdistribusi *Levy*
γ : skala yang mengatur gerak *Levy*
λ : parameter gerak Levy ($1 \leq \lambda \leq 2$)
$\boldsymbol{x}_j{}^{t}$: serbuk sari yang berasal dari bunga j pada tanama yang sama
$\boldsymbol{x}_k{}^{t}$: serbuk sari yang berasal dari bunga k pada tanaman yang sama
ϵ : random walk yang berdistribusi $U(0,1)$

Misalkan terdapat masalah optimasi yang bersesuaian dengan persamaan 3, maka *flowchart* dari *Flower Pollination Algorithm* untuk permasalahan optimasi global adalah sebagai berikut:



Gambar 2. Algoritma FPA

2.4 Teknik Clustering

Algoritma FPA meskipun bekerja secara berkelompok dalam mencari solusi, akan tetapi tidak bisa menemukan semua solusi sekaligus dalam satu kali proses *running*. Artinya jika ingin mendapatkan solusi lain maka proses pencarian perlu dilakukan berulang. Mengingat solusi dalam sistem persamaan *Diophantine* tidak hanya satu maka diperlukan teknik *Clustering* untuk melokalisir kluster-kluster potensial yang kemungkinan mengandung solusi. Pada umumnya algoritma metaheuristik dikombinasikan dengan teknik *Niching* seperti *Fitness sharing*, *Crowding, Derating, dan Speciation* [16].

Pada penelitian ini , kami menggunakan teknik *Clustering* yang dikembangkan oleh Sidarto [17]. Teknik Clustering adalah proses pengelompokan berdasarkan kesamaan anggota dalam satu populasi. Pada prosesnya, teknik *Clustering* bertugas khusus untuk memetakan daerah potensial dalam suatu area pencarian. Daerah-daerah potensial kemudian menjadi pusat kluster yang mendukung terbentuknya sub-sub kluster. Kluster pada penelitian ini didefinisikan sebagai semua vektor-vektor $\boldsymbol{y}$ yang memenuhi $|\boldsymbol{x}-\boldsymbol{y}|<\rho$ dimana $\boldsymbol{x}$ adalah pusat kluster dan ρ adalah jari-jari kluster. Kemampuan teknik *Clustering* dalam melokalisir daerah potensial terbukti efektif untuk menemukan semua solusi fungsi multimodal baik untuk benchmark dimensi rendah maupun dimensi tinggi [13].

2.5 Flower Pollination Algorithm dengan Teknik Clustering untuk Permasalahan Diophantine

FPAC dalam [13] tidak dapat langsung digunakan untuk permasalahan *Diophantine*, karena permasalahan multimodal berbentuk fungsi, sementara permasalahan *Diophantine* berbentuk persamaan ataupun sistem persamaan baik yang linear, nonlinear. Selain itu juga, solusi yang diharapkan pada permasalahan multimodal adalah akar persamaan atau sistem persamaan yang merupakan bilangan bulat *non-negatif*. Oleh karena itu beberapa tahapan pada FPAC dikembangkan sebelum disimulasikan pada *benchmark* persamaan *Diophantine* ataupun sistem persamaan *Diophantine*.

Misalkan terdapat permasalahan *Diophantine* dalam bentuk persamaan ataupun sistem persamaan yang bersesuaian dengan persamaan 2, maka metode FPAC untuk menemukan semua solusi dari permasalahan *Diophantine* dalam sekali *running* adalah adalah sebagai berikut:

I. Input

$m_{di}, \gamma_{di}, \lambda_{di}, p_d, k_{di}$	: Parameter FPAC (diversifikasi)
$m_{in}, \gamma_{in}, \lambda_{in}, p_{in}, k_{max}$	: Parameter FPAC (intensifikasi)
$\varepsilon_{cs}(0<\varepsilon_{cs}<1)$	: Parameter FPAC (elitis: calon solusi)
$\delta_s\ (0<\delta_s\ <1)$	: Parameter FPAC (elitis: solusi)

II. Proses

Transformasi fungsi objektif

1. Definisikan permasalahan *Diophantine* sesuai persamaan 4

Tahap diversifikasi

2. Bangkitkan populasi awal $\boldsymbol{x}_i(0)\in\mathbb{R}^n, i=1,2,\cdots,m_{di}$ dalam daerah pencarian, dimana $D\subset\mathbb{R}^n$, dan atur $k_{di}=0$
3. Atur fitness terbaik $\boldsymbol{x}^*=round\left(\boldsymbol{x}_{ig}(0)\right),\ i_g=\arg\max F\left(\boldsymbol{x}_i(0)\right), i=1,2,\cdots,m_{di}$ sebagai pusat kluster awal dengan jari-jari $r=\frac{1}{2}(min|b_l-a_l|), l=1,2,\cdots,n.$
4. Untuk setiap $i=1,2,\cdots,m_{di}$ lakukan:
   Jika $\boldsymbol{x}_i$ bukan pusat kluster yang ada, maka lakukan *fungsi clustering* berikut dengan $\boldsymbol{x}_i$ sebagai input. Asumsikan $\boldsymbol{y}=\boldsymbol{x}_i$, maka pada *fungsi clustering* lakukan:
   4.1. Temukan pusat kluster yang terdekat dengan $\boldsymbol{y}$. Asumsikan pusat kluster tersebut adalah $\boldsymbol{x}_c$.
   4.2. Atur $\boldsymbol{x}_m$ sebagai vektor yang terletak di tengah-tengah $\boldsymbol{x}_c$ dan $\boldsymbol{y}$

4.3. Bandingkan nilai fungsi $F(\boldsymbol{x}_m)$, $F(\boldsymbol{x}_c)$ dan $F(\boldsymbol{y})$:

4.3.1 Jika $F(\boldsymbol{x}_m) < F(\boldsymbol{x}_c)$ dan $F(\boldsymbol{x}_m) < F(\boldsymbol{y})$ maka $round\ (\boldsymbol{y})$ adalah pusat kluster baru dengan jari-jari $\rho = \|\boldsymbol{x}_m - \boldsymbol{y}\|$

4.3.2 Jika $F(\boldsymbol{x}_m) > F(\boldsymbol{x}_c)$ dan $F(\boldsymbol{x}_m) > F(\boldsymbol{y})$ maka $round\ (\boldsymbol{y})$ adalah pusat kluster baru dengan jari-jari $\rho = \|\boldsymbol{x}_m - \boldsymbol{y}\|$. Lakukan kembali *fungsi clustering* (langkah 4) dengan $\boldsymbol{x}_m$ sebagai input $\boldsymbol{y} = \boldsymbol{x}_m$.

4.3.3 Jika $F(\boldsymbol{y}) > F(\boldsymbol{x}_c)$ maka *update* pusat kluster terdekat $\boldsymbol{x}_c = round\ (\boldsymbol{y})$, begitupun dengan jari-jari $\boldsymbol{x}_c$ yaitu $\rho = \|\boldsymbol{x}_m - \boldsymbol{y}\|$

5. Lakukan perubahan pada populasi awal untuk mengeksplorasi pusat kluster potensial lainnya.

for $i = 1{:}\, m_{di}$
  if $rand < p$

$$\boldsymbol{L}(\lambda) = \frac{randn(1,n) \times \left[\frac{r(1+\lambda)}{\lambda r(1+\lambda)/2} \times \frac{sin(\pi\lambda/2)}{2^{(\lambda-1)/2}}\right]^{1/\lambda}}{|randn(1,d)|^{1/\lambda}}$$

$$\boldsymbol{x}_i{}^{t+1} = round\left(\boldsymbol{x}_i{}^{t} + \gamma \boldsymbol{L}(\lambda)(\boldsymbol{g}_* - \boldsymbol{x}_i{}^{t})\right)$$

  else

$$\epsilon \sim U[0,1]$$

$$\boldsymbol{x}_i{}^{t+1} = round\left(\boldsymbol{x}_i{}^{t} + \epsilon\left(\boldsymbol{x}_j{}^{t} - \boldsymbol{x}_k{}^{t}\right)\right)$$

  end if
end for

6. Lakukan langkah 4-6 sebanyak k_{di} kali

Tahap intensifikasi

7. Bentuk kluster $C_1, C_2, \cdots, C_{n_k}$ pada setiap pusat kluster $\boldsymbol{x}_{C_i}$ dengan jari-jari adaptif ρ_i, dimana $i = 1,2, \cdots, n_C$
8. Lakukan optimasi pada setiap kluster potensial $C_1, C_2, \cdots, C_{n_k}$ dengan menggunakan algoritma *FPA* dimana setiap vektor hasil *global pollination* ataupun *local pollination* ditransformasi ke dalam bentuk bilangan bulat.

Tahap elitis

9. Simpan solusi optimum atau calon solusi $(\boldsymbol{x}_{cs})$ yang hanya memenuhi $1 - F(\boldsymbol{x}) < \varepsilon_{cs}$
10. Solusi dari persamaan atau sistem persamaan *Diophantine* adalah calon solusi pada n_g yang memenuhi $\|\boldsymbol{x}_i - \boldsymbol{x}_j\| > \delta_s$ untuk $i, j = 1,2, \cdots, n_g$. Untuk keadaan dimana $\|\boldsymbol{x}_i - \boldsymbol{x}_j\| < \delta$, maka $\boldsymbol{x}_i$ adalah solusi jika $F(\boldsymbol{x}_i) > F(\boldsymbol{x}_j)$, Jika tidak maka solusi adalah $\boldsymbol{x}_j$.

III. Output

Solusi dari persamaan atau sistem persamaan *Diophantine* adalah $\boldsymbol{x}$ yang memenuhi $F(\boldsymbol{x}) \approx 1$ atau $\boldsymbol{f}(\boldsymbol{x}) = 0$ dimana $\boldsymbol{x}$ adalah bilangan bulat.

3. HASIL DAN PEMBAHASAN

Jenis masalah *Diophantine* yang paling sederhana adalah *Diophantine* dalam bentuk persamaan linear dua variabel yaitu $ax_1 + bx_2 = c$ dimana a, b dan c bilangan bulat. Solusi dari persamaan tersebut adalah pasangan bilangan bulat *non-negatif* yang memenuhi $ax_1 + bx_2 = c$.

Misalkan diberikan persamaan *Diophantine* linear sederhana $3x_1 + 6x_2 = 18$ yang memiliki salah satu solusi $x_1 = 4$ dan $x_2 = 1$ dalam daerah pencarian $\{\boldsymbol{x} \in \mathrm{Z} | 0 \leq \boldsymbol{x} \leq 10\}$. Proses penyelesaian secara bertahap dengan menggunakan metode FPAC untuk mendapatkan seluruh solusi yang tersedia dalam sekali *running* adalah sebagai berikut:

I. Input

1. Parameter diversifikasi
   $m_{di} = 100, \gamma_{di} = 1, \lambda_{di} = 1.5, p_{di} = 0.8, k_{di} = 3$

   m_{di} adalah banyaknya populasi awal yang disebar dalam daerah pencarian untuk mendapatkan pusat kluster potensial. λ_{di} adalah parameter dari *Levy flight* sedangkan $0 \leq \gamma_{di} \leq 1$ adalah parameter dalam FPA yang mengatur besaran langkah atau skala dari *Levy flight* (*Global pollination*). p_{di} adalah *switch parameter* yang menentukan seberapa besar *local* dan *global pollination* digunakan dalam FPA. k_{di} adalah parameter yang digunakan untuk memperbesar peluang ditemukan pusat kluster potensial lainnya dalam daerah pencarian.
2. Parameter Intensifikasi
   $m_{in} = 25, \gamma_{in} = \gamma_{di}, \lambda_{in} = \lambda_{di}, p_{in} = p_{di}, k_{max} = 100$

   m_{in} adalah banyaknya populasi yang dibentuk dalam setiap kluster potensial. k_{max} adalah banyaknya iterasi yang digunakan untuk mendapatkan solusi optimum dalam setiap kluster potensial. Sedangkan γ_{di}, λ_{in}, dan p_{in} memiliki penjelasan yang sama dengan parameter *diversifikasi*.
3. Parameter elitis
   $\varepsilon_{el} = 10^{-6}, \delta_{el} = 10^{-2}$

   ε_{el} adalah parameter yang digunakan untuk menyeleksi calon solusi karena tidak semua solusi optimum pada setiap kluster potensial adalah calon solusi yang diharapkan. Sedangkan δ_{el} adalah parameter untuk menyeleksi solusi dari calon solusi. Parameter ini juga dapat menyeleksi solusi yang bertumpuk diposisi yang sama.

II. Proses

1. Berdasarkan persamaan 4 maka bentuk fungsi objektif dari persamaan *Diophantine* $3x_1 + 6x_2 = 18$ yaitu:
$$f_{obj} = \frac{1}{1 + |3x_1 + 6x_2 - 18|}$$
2. Populasi awal dibentuk secara acak sebanyak $m_{di} = 100$ individu dalam daerah pencarian $D = \{\boldsymbol{x} \in \mathrm{Z} | 0 \leq \boldsymbol{x} \leq 10\}$.
3. Individu yang memberikan nilai *fitness* (f_{obj}) terbaik adalah pusat kluster potensial pertama dalam daerah pencarian. Pada tahap ini, 1 dari 100 individu yang dibangkitkan terpilih sebagai pusat kluster potensial pertama ($\boldsymbol{x}_{c1}$).
4. Untuk mendapatkan pusat kluster potensial lainnya, dilakukan *fungsi clustering* sesuai langkah 4.1 - 4.3. *Fungsi clustering* melakukan seleksi pada 99 individu yang tersisa. *Fungsi clustering* secara detail dapat dijelaskan sebagai berikut:
   4.1 Diantara 99 individu yang tersisa, individu yang paling dekat dengan pusat kluster yang tersedia ($\boldsymbol{x}_{c1}$) akan terpilih sebagai calon pusat kluster potensial ($\boldsymbol{y}_1$) yang perlu diuji.
   4.2 Di tengah-tengah antara pusat kluster terdekat ($\boldsymbol{x}_{c1}$) dan calon pusat kluster potensial ($\boldsymbol{y}_1$) , dibentuk calon pusat kluster potensial lainnya ($\boldsymbol{x}_{m1}$).
   4.3 Nilai *fitnes* (f_{obj}) dari $\boldsymbol{x}_{c1}$, $\boldsymbol{y}_1$ dan $\boldsymbol{x}_{m1}$ akan dibandingkan. $\boldsymbol{y}_1$ akan terpilih sebagai pusat kluster potensial baru ($\boldsymbol{x}_{c2}$) jika memenuhi kondisi 4.3.1. $\boldsymbol{x}_{m1}$ akan terpilih menjadi pusat kluster potensial baru ($\boldsymbol{x}_{c2}$) jika memenuhi kondisi 4.3.2. $\boldsymbol{y}_1$ dapat menggantikan pusat $\boldsymbol{x}_{c1}$ jika memenuhi kondisi 4.3.3. Setiap pusat kluster memiliki jari-jari adaptif (ρ).
5. Untuk mendapatkan pusat kluster potensial lainnya, dilakukan perubahan populasi awal melalui skema *local* dan *global* pollination pada FPA.
6. Langkah 4-5 dilakukan sebanyak $k_{di} = 3$ kali. Banyaknya inputan k_{di} tergantung pada kerumitan masalah dan luasnya daerah pencarian. Pada tahap ini diperoleh 11 pusat kluster potensial dengan jari-jari adaptif (ρ) yang ditunjukan oleh tabel 1 (Langkah 1-6).

7. Pada setiap pusat kluster potensial dibentuk kluster dengan luas berbeda sesuai jari-jari (ρ) dari masing-masing pusat kluster. Setiap kluster potensial yang terbentuk memiliki populasi yang sama yaitu $m_{in} = 25$. Ukuran populasi pada kluster potensial bergantung kerumitan masalah.
8. Pada setiap kluster yang terbentuk dilakukan optimasi global sesuai algoritma FPA. Jika terdapat 11 kluster maka akan menghasilkan 11 hasil optimum juga. Hasil Optimasi Global pada setiap kluster tersebut ditunjukan oleh tabel 1 (Langkah 7-8).
9. Hasil optimasi dengan FPA akan menjadi calon solusi jika memenuhi kondisi $1 - F(\boldsymbol{x}) < \varepsilon$. Pada kasus ini keseluruhan hasil optimasi global memenuhi syarat untuk menjadi calon solusi. 11 calon solusi yang terpilih dapat dilihat pada tabel (Langkah 9).
10. Untuk menghindari tumpukan solusi pada posisi yang sama maka dilakukan seleksi. Pada tabel 1 (Langkah 10) dapat diketahui solusi $x_1 = 4$ dan $x_2 = 1$ dapat berasal dari kluster 1, dan 4. Solusi $x_1 = 2$ dan $x_2 = 2$ dapat berasal dari kluster 2,5,6,7,9 dan 10. Solusi $x_1 = 6$ dan $x_2 = 0$ hanya berasal dari kluster 3, dan solusi $x_1 = 0$ dan $x_2 = 3$ berasal dari kluster 8 dan 11.

III. Output

Dengan menggunakan metode FPAC maka ditemukan 4 solusi yang tersedia dalam daerah pencarian $D = \{\boldsymbol{x} \in \mathrm{Z} | 0 \leq \boldsymbol{x} \leq 10\}$ pada persamaan *Diophantine* $3x_1 + 6x_2 = 18$. Secara lebih rinci gambaran umum proses metode FPAC untuk permasalahan *Diophantine* ditunjukan oleh tabel 1 berikut:

Tabel 1. Tahapan metode FPAC dalam menemukan semua solusi persamaan *Diophantine* sederhana dalam sekali *running*

No	Langkah 1 - 6			Langkah 7 - 8			Langkah 9			Langkah 10		
	Pusat Kluster		Jari-Jari Kluster (ρ)	Hasil Optimasi FPA			Calon Solusi			Solusi		
	x_{1C}	x_{2C}		$x_{1\,opt}$	$x_{2\,opt}$	f_{obj}	$x_{1\,cs}$	$x_{2\,cs}$	f_{obj}	$x_{1\,s}$	$x_{2\,s}$	$f(\boldsymbol{x}) = 3x_1 + 6x_2 = 18$
1	3.6235	1.3719	1.1221	4	1	1	4	1	1	4	1	18
2	4.0893	0.9936	1.4613	2	2	1	2	2	1	2	2	18
3	5.3274	0.3264	2.3420	6	0	1	6	0	1	6	0	18
4	4.4487	0.7811	0.3955	4	1	1	4	1	1	-	-	-
5	2.6641	1.6624	1.5701	2	2	1	2	2	1	-	-	-
6	1.8657	2.1396	4.0193	2	2	1	2	2	1	-	-	-
7	2.6053	1.8082	0.1571	2	2	1	2	2	1	-	-	-
8	0	2.9482	0.9174	0	3	1	0	3	1	0	3	18
9	1.2438	2.1797	0.6232	2	2	1	2	2	1	-	-	-
10	0	4.3243	1.7262	2	2	1	2	2	1	-	-	-
11	0	3.4069	0.0266	0	3	1	0	3	1	-	-	-

3.1 Simulasi Numerik

Kontribusi penelitian ini adalah terletak pada kemampuan *Flower Pollination Algorithm with Clustering* (FPAC) dalam menemukan semua solusi permasalahan *Diophantine* dengan sekali *running*. Untuk mengkonfirmasi hal tersebut maka, FPAC diuji pada 2 bentuk permasalahan *Diophantine* yaitu *benchmark* persamaan *Diophantine* dengan 3 kondisi: (1) persamaan dengan jumlah variabel yang berbeda tetapi pangkatnya sama; (2) persamaan dengan besaran pangkat yang berbeda tapi jumlah variabelnya sama; (3) persamaan dengan jumlah variabel dan pangkat yang berbeda tapi dalam bentuk eksponensial, dan *Benchmark* Sistem Persamaan *Diophantine* dengan 2 kondisi: (1) sistem persamaan *Diophantine* dimensi rendah; dan (2) Sistem persamaan *Diophantine* dimensi tinggi.

3.1.1. Simulasi Numerik pada *Benchmark* Persamaan *Diophantine*

Metode FPAC sangat bergantung pada tahap diversifikasi, dimana pada tahap tersebut pusat-pusat kluster potensial akan ditemukan berdasarkan sebaran populasi awal yang dibangkitkan secara acak. Pada tahap intensifikasi juga terdapat tahap pembangkitan populasi untuk setiap pusat kluster potensial sebelum dilakukan proses optimasi dengan FPA. Oleh karena itu, penentuan parameter sangat penting dalam metode FPAC. Penggunaan parameter yang tidak

tepat dapat mempengaruhi kemampuan FPAC dalam menemukan jumlah solusi permasalahan *Diophantine*. Inputan parameter dalam penelitian ini, diklaim efektif jika FPAC dapat menemukan seluruh solusi yang tersedia pada *benchmark* yang diujikan dalam sekali *running*. Beberapa hal yang perlu dipertimbangkan dalam menentukan parameter metode FPAC adalah luas daerah pencarian, kerumitan fungsi objektif dan efisiensi kinerja algoritma.

Dengan menggunakan parameter berikut: $m_{di} = 500$, $k_{di} = 5$, $m_{in} = 35$, $p_{in} = 0.8$, $\lambda_{in} = 1.5$, $\varepsilon_{cs} = 10^{-6}$, $\delta_s = 10^{-2}$, $k_{max} = 200$, dan daerah pencarian solusi berada dalam rentang $[1 - 30]$, maka hasil simulasi numerik dari persamaan *Diophantine* dengan kondisi (1) dan (2) yang ditemukan oleh FPAC, dapat dilihat pada tabel 2 dan tabel 3.

Tabel 2. Solusi yang ditemukan pada *benchmark* persamaan *Diophantine* dengan jumlah variabel yang berbeda menggunakan FPAC dalam sekali *running* (kondisi 1)

No	Persamaan Diophantine	Jumlah Variabel	Sampel Solusi Yang Ditemukan	Jumlah Solusi Yang Ditemukan
1	$x_1^2 + x_2^2 = 149$	2	x_1: 7; x_2: 10	1
2	$x_1^2 + x_2^2 + x_3^2 = 244$	3	x_1: 6; x_2: 8; x_3: 12	1
3	$x_1^2 + x_2^2 + \cdots + x_4^2 = 295$	4	x_1: 3, 1, 3, 5, 1; x_2: 5, 1, 6, 5, 7; x_3: 6, 2, 9, 7, 7; x_4: 15, 17, 13, 14, 14	10
4	$x_1^2 + x_2^2 + \cdots + x_6^2 = 420$	6	x_1: 3, 2, 1, 1, 1; x_2: 3, 2, 1, 1, 5; x_3: 6, 4, 7, 2, 5; x_4: 7, 10, 10, 5, 7; x_5: 11, 10, 10, 10, 8; x_6: 14, 14, 13, 17, 16	> 50
5	$x_1^2 + x_2^2 + \cdots + x_8^2 = 590$	8	x_1: 2, 1, 1, 1, 1; x_3: 2, 2, 5, 1, 1; x_4: 3, 4, 7, 3, 5; x_5: 6, 6, 7, 9, 10; x_6: 7, 8, 7, 10, 11; x_7: 8, 8, 10, 10, 11; x_8: 10, 9, 11, 13, 14; x_8: 18, 18, 14, 15, 14	> 80

Hasil simulasi numerik yang ditunjukan oleh tabel 2 menginformasikan bahwa FPAC dapat menemukan semua solusi yang tersedia pada persamaan 1-3, dan menemukan lebih dari 50 solusi yang tersedia pada persamaan 4 serta lebih dari 80 solusi yang tersedia pada persamaan 5. Jumlah variabel yang berbeda yang membentuk persamaan *Diophantine* cukup mempengaruhi kemampuan FPAC dalam menemukan sebanyak mungkin solusi yang tersedia dalam sekali *running*.

Tabel 3. Solusi yang ditemukan pada *benchmark* persamaan *Diophantine* dengan besaran pangkat yang berbeda menggunakan FPAC dalam sekali *running* (kondisi 2)

No	Persamaan Diophantine	Besaran Pangkat	Sampel Solusi Yang Ditemukan	Jumlah Solusi Yang Ditemukan
1	$x_1^3 + x_2^3 = 1008$	3	$2, 10$	1
2	$x_1^5 + x_2^5 = 19932$	5	$5, 7$	1

3	$x_1^7 + x_2^7 = 4799353$	7	4, 9	1
4	$x_1^9 + x_2^9 = 1000019683$	9	3, 10	1
5	$x_1^{10} + x_2^{10} = 1356217073$	10	7, 8	1

Hasil simulasi numerik yang ditunjukan oleh tabel 3 menginformasikan bahwa FPAC dapat menemukan semua solusi yang tersedia pada persamaan 1-5, dimana keseluruhan persamaan yang diujikan memiliki masing-masing 1 solusi. Jumlah variabel yang sama dengan besaran pangkat berbeda yang yang membentuk persamaan *Diophantine* tidak mempengaruhi kemampuan FPAC dalam menemukan semua solusi yang tersedia dalam sekali *running*.

Dengan menggunakan parameter berikut: $m_{di} = 500$, $k_{di} = 5$, $m_{in} = 35$, $p_{in} = 0.8$, $\lambda_{in} = 1.5$, $\varepsilon_{cs} = 10^{-6}$, $\delta_s = 10^{-2}$, $k_{max} = 200$, dan daerah pencarian solusi berada pada rentang $[0 - 10]$, maka hasil simulasi numerik dari persamaan *Diophantine* dengan kondisi (3) yang ditemukan FPAC, dapat dilihat pada tabel 4.

Tabel 4. Solusi yang ditemukan pada *benchmark* persamaan *Diophantine* (bentuk eksponensial) menggunakan FPAC dalam sekali *running* (kondisi 3)

No	Persamaan Diophantine	Jumlah Variabel	Sampel Solusi Yang Ditemukan	Jumlah Solusi Yang Ditemukan
1	$1 + 5^{x_1} = 3^{x_2} + 3^{x_3}$	3	x_1 | x_2 | x_3 1 | 1 | 1 0 | 0 | 0	2
2	$13^{x_1} + 7^{x_2} = 3^{x_3} + 5^{x_4}$	4	x_1 | x_2 | x_3 | x_4 0 | 0 | 0 | 0 1 | 0 | 2 | 1 0 | 1 | 1 | 1	3
3	$5^{x_1} + 5^{x_2} = 3^{x_3} + 7^{x_4}$	4	x_1 | x_2 | x_3 | x_4 1 | 1 | 1 | 1 1 | 1 | 2 | 0 1 | 5 | 6 | 3 5 | 1 | 6 | 4 3 | 1 | 4 | 2 3 | 3 | 5 | 1 2 | 2 | 0 | 2 0 | 0 | 0 | 0 1 | 3 | 4 | 2	9
4	$1 + 2^{x_1} + 7^{x_2} = 3^{x_3} + 5^{x_4}$	4	x_1 | x_2 | x_3 | x_4 1 | 0 | 1 | 0 5 | 0 | 2 | 2 1 | 2 | 3 | 2 5 | 2 | 4 | 0 1 | 1 | 2 | 0 3 | 0 | 2 | 0 2 | 0 | 0 | 1	7
5	$5^{x_1} + 5^{x_2} + 3^{x_3} = 3^{x_4}$	4	x_1 | x_2 | x_3 | x_4 2 | 0 | 0 | 3 0 | 1 | 1 | 2 0 | 0 | 0 | 1 1 | 0 | 1 | 2 0 | 2 | 0 | 3	5

Hasil simulasi numerik yang ditunjukan oleh tabel 4 menginformasikan bahwa FPAC dapat menemukan semua solusi yang tersedia pada persamaan 1-5, meskipun masing-masing persamaan yang diujikan memiliki jumlah solusi yang berbeda. Persamaan *Diophantine* dalam bentuk eksponensial dengan jumlah variabel berbeda, tidak mempengaruhi kemampuan FPAC dalam menemukan semua solusi yang tersedia dalam sekali *running.*

3.1.2. Simulasi Numerik *Benchmark* Sistem Persamaan *Diophantine*

FPAC juga evaluasi efektifitasnya melalui permasalahan *Diophantine* dalam bentuk sistem persamaan dengan ukuran dimensi yang berbeda yaitu rendah (kasus 1) dan tinggi (kasus 2 & 3).

3.1.2.1 Kasus 1: Sistem persamaan *Diophantine* yang berdimensi $2x3$

Diberikan sebuah sistem persamaan (7) sebagai berikut:

$$x_1 + {x_2}^2 - x_3 = 115$$
$$4x_1 + 8x_2 + 7{x_1}^2 = 335$$

dimana daerah pencarian $D = \{\boldsymbol{x} \in \mathrm{Z} | 0 \leq \boldsymbol{x} \leq 30\}$.

Metode FPAC adalah gabungan antara metode FPA dengan Teknik *Clustering*. Tahapan *Global dan local pollination* pada FPA sangat sensitif terhadap parameter karena dapat mempengaruhi pergerakan individu dalam daerah pencarian [18]. Hal yang sama juga dapat terjadi pada FPAC. Pada dasarnya, semakinn rapat sebaran populasi awal pada daerah pencarian di tahap diversifikasi, semakinn besar peluang ditemukannya pusat kluster potensial, tetapi *computational cost* dari FPAC akan semakin tinggi. Oleh karena itu, pemilihan parameter FPAC pada setiap kasus dapat berbeda bergantung dengan kualitas permasalahan yang dihadapi.

Dengan menggunakan parameter berikut: $m_{di} = 500$, $k_{di} = 5$, $m_{in} = 35$, $p_{in} = 0.8$, $\lambda_{in} = 1.5$, $\varepsilon_{cs} = 10^{-6}$, $\delta_s = 10^{-2}$, $k_{max} = 100$, maka solusi dari sistem persamaan *Diophantine* (kasus 1) yang ditemukan oleh FPAC dapat dilihat pada tabel 5 berikut.

Tabel 5. Solusi yang ditemukan pada *benchmark* sistem persamaan *Diophantine* yang berdimensi rendah (2x3) menggunakan FPAC dalam sekali *running*

No	x_1	x_2	x_3	$f_1(\boldsymbol{x})$	$f_2(\boldsymbol{x})$
1	20	10	5	115	335

Hasil simulasi numerik yang ditunjukan oleh tabel 5 menginformasikan bahwa FPAC dapat menemukan semua solusi yang tersedia dalam sekali *running* pada permasalahan sistem persamaan *Diophantine* dimensi rendah (kasus 1). Sistem persamaan *Diophantine* dengan ukuran 2x3 tidak mempengaruhi kemampuan FPAC dalam menemukan semua solusi yang tersedia.

3.1.2.2 Kasus 2: Sistem persamaan *Diophantine* yang berdimensi $6x6$

Diberikan sebuah sistem persamaan (8) sebagai berikut:

$$5x_1 + 10x_2 - 5x_3 + {x_5}^3 + 8x_6 = 1772$$
$$3x_1 + 18x_3 - 5x_5 + 17x_6 = 153$$
$$6x_1 + x_3 - 99x_2 + 17{x_6}^2 = 1772$$
$$-x_1 + 5x_2 + 8x_3 - 6x_4 + 15x_5 + 10x_6 = 277$$
$$(x_1 + x_2)^2 - 7x_3 + 5x_4 + 12x_5 - 8x_6 = 150$$
$$x_2 + 5x_3 + 3x_5 - x_6 = -4$$

dimana daerah pencarian $D = \{\boldsymbol{x} \in \mathrm{Z} | 0 \leq \boldsymbol{x} \leq 30\}$.

Dengan menggunakan parameter berikut: $m_{di} = 500$, $k_{di} = 5$, $m_{in} = 35$, $p_{in} = 0.8$,

$\lambda_{in} = 1.5$, $\varepsilon_{cs} = 10^{-6}$, $\delta_s = 10^{-2}$, $k_{max} = 100$, maka solusi dari sistem persamaan *Diophantine* (kasus 2) yang ditemukan oleh FPAC dapat dilihat pada tabel 6 berikut.

Tabel 6. Solusi yang ditemukan pada *benchmark* sistem persamaan *Diophantine* yang berdimensi 6x6 menggunakan FPAC dalam sekali *running*

No	x_1	x_2	x_3	x_4	x_5	x_6	$f_1(\boldsymbol{x})$	$f_2(\boldsymbol{x})$	$f_3(\boldsymbol{x})$	$f_4(\boldsymbol{x})$	$f_5(\boldsymbol{x})$	$f_6(\boldsymbol{x})$
1	6	3	8	1	12	3	1772	153	1772	277	150	-4

Hasil simulasi numerik yang ditunjukan oleh tabel 6 menginformasikan bahwa FPAC dapat menemukan semua solusi yang tersedia dalam sekali *running* pada permasalahan sistem persamaan *Diophantine* dimensi tinggi (kasus 2). Sistem persamaan *Diophantine* dengan ukuran 6x6 tidak mempengaruhi kemampuan FPAC dalam menemukan semua solusi yang tersedia.

3.1.2.3 Kasus 3: Sistem persamaan *Diophantine* yang berdimensi $9x10$

Diberikan sebuah sistem persamaan (9) sebagai berikut:

$$
\begin{aligned}
{x_1}^2 - 2(x_2 + x_4)^3 + x_5 - 3x_6 - x_7 + 4x_9 + 15x_{10} &= -24 \\
2x_1 + (x_2 + 3x_4)^3 + 5{x_7}^2 - 6x_8 + x_9 - 9x_{10} &= 31 \\
3x_1 - 2{x_2}^2 + 10x_3 - 9x_4 + 3x_5 + x_6 - 2x_7 - 8x_8 + 12x_9 - 5x_{10} &= -25 \\
5x_1 + 2x_2 - 8x_4 - 3x_5 + 4x_6 + x_7 - x_9 &= 23 \\
x_1 - x_3 + 2x_5 - x_7 - x_9 &= -3 \\
x_2 + 2{x_4}^2 - 6x_6 - x_8 + 2x_{10} &= 8 \\
3x_1 + 2x_2 - 5x_3 - {x_4}^4 - 2x_5 + x_6 + 4x_7 - 10x_8 + 8x_9 &= -9 \\
x_1 - 3x_2 + 4x_4 + x_6 - 6x_7 + x_8 - 2x_9 &= -16 \\
(2x_1 + x_2)^2 + 3x_3 - 10x_5 - (x_6 + 3x_7)^3 - x_8 - 6x_9 &= 27
\end{aligned}
$$

dimana daerah pencarian $D = \{\boldsymbol{x} \in \mathrm{Z} | 0 \leq \boldsymbol{x} \leq 10\}$.

Kasus 3 adalah sistem persamaan *Diophantine* dengan dimensi yang cukup tinggi. Penggunaan parameter yang sama seperti pada kasus 1 dan 2 tidak akan efektif bagi metode FPAC dalam menemukan semua solusi yang tersedia. Oleh karena itu perubahan inputan parameter pada tahap diversifikasi dan intensifikasi dilakukan.

Dengan menggunakan parameter berikut: $m_{di} = 1000$, $k_{di} = 5$, $m_{in} = 35$, $p_{in} = 0.8$, $\lambda_{in} = 1.5$, $\varepsilon_{cs} = 10^{-6}$, $\delta_s = 10^{-2}$, $k_{max} = 500$, maka solusi dari sistem persamaan *Diophantine* (kasus 3) yang ditemukan oleh FPAC dapat dilihat pada tabel 7 berikut.

Tabel 7. Solusi yang ditemukan pada *benchmark* sistem persamaan *Diophantine* yang berdimensi 9x10 menggunakan FPAC dalam sekali *running*

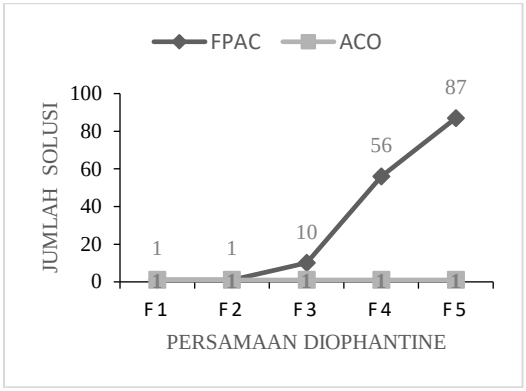
No	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	$f_1(\boldsymbol{x})$	$f_2(\boldsymbol{x})$	$f_3(\boldsymbol{x})$	$f_4(\boldsymbol{x})$	$f_5(\boldsymbol{x})$	$f_6(\boldsymbol{x})$	$f_7(\boldsymbol{x})$	$f_8(\boldsymbol{x})$	$f_9(\boldsymbol{x})$
1	3	4	5	0	1	1	1	2	2	6	-24	31	-25	23	-3	8	-9	-16	27

Hasil simulasi numerik yang ditunjukan oleh tabel 7 menginformasikan bahwa FPAC dapat menemukan semua solusi yang tersedia dalam sekali *running* pada permasalahan sistem persamaan *Diophantine* dimensi tinggi (kasus 3). Sistem persamaan *Diophantine* dengan ukuran 9x10 tidak mempengaruhi kemampuan FPAC dalam menemukan semua solusi yang tersedia.

3.2 Evaluasi

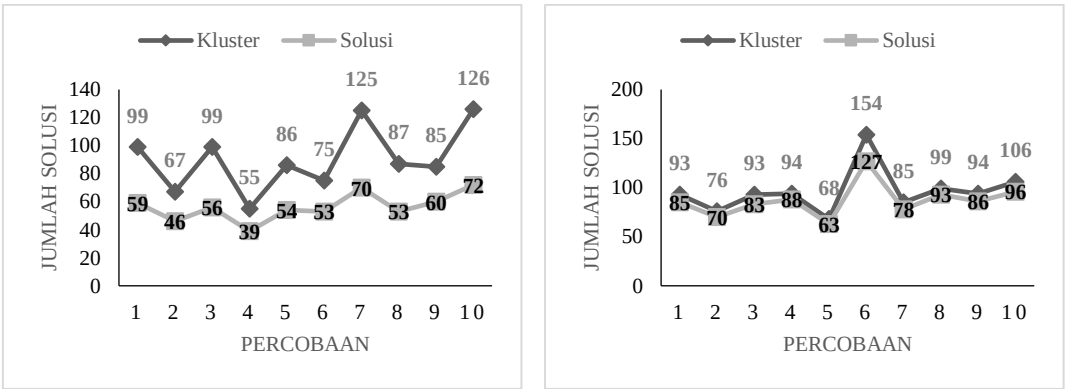
Parameter untuk mengevaluasi kemampuan FPAC dalam penelitian ini adalah jumlah

solusi yang mampu ditemukan dalam daerah pencarian. Parameter lain seperti *computational cost* tidak bisa diukur karena penelitian ini menggunakan data sekunder sebagai pembanding. Kemampuan FPAC dalam menemukan semua solusi yang tersedia pada *benchmark* persamaan *Diophantine* dibandingkan dengan hasil capaian *Ant Colony Optimization* (ACO) yang dilakukan oleh Siby Abraham dalam [7]. Sedangkan kemampuan FPAC dalam menemukan semua solusi yang tersedia pada *benchmark* sistem persamaan *Diophantine* dan persamaan *Diophantin*e dalam bentuk eksponensial dibandingkan dengan hasil capaian *Particle Swarm Optimization* (PSO) yang dilakukan oleh Oscar Perez dalam [5].



Gambar 3. Perbandingan FPAC dan ACO dalam menemukan semua solusi pada masing-masing *benchmark* persamaan *Diophantine* (kondisi 1) dalam sekali *running*

Pengujian pada *benchmark* persamaan *Diophantine* yang memenuhi kondisi (1), metode FPAC menemukan jumlah solusi yang sama dengan metode ACO khususnya pada persamaan 1 dan 2. Hal ini terjadi karena persamaan 1 dan 2 memang hanya memiliki 1 solusi. Akan tetapi pada persamaan 3, 4 dan 5 metode FPAC menemukan jumah solusi yang berbeda dengan metode ACO. Metode ACO dalam [7] hanya menemukan 1 solusi, sementara metode FPAC dapat menemukan 10 solusi pada persamaan 3 seperti yang terlihat dalam gambar 3. Kemampuan metode FPAC dalam menemukan semua solusi yang tersedia pada persamaan 3 dalam sekali *running* merupakan kontribusi dari teknik *clustering* yang bekerja pada tahap diversifikasi.



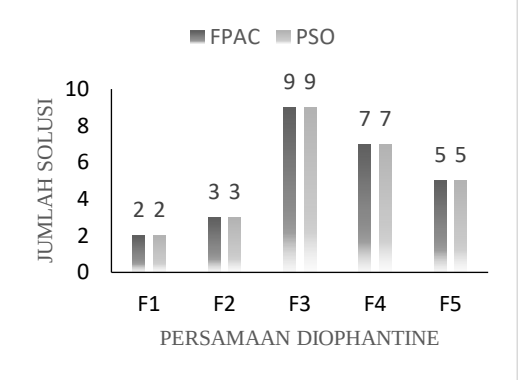
(a) (b)

Gambar 4. Jumlah solusi yang ditemukan metode FPAC pada persamaan 4 (gambar a) dan persamaan 5 (gambar b) (kondisi 1) dalam sekali *running*

Persamaan 4 dan 5 yang memenuhi kondisi (1) merupakan *benchmark* persamaan *Diophantine* dengan jumlah solusi yang belum diketahui secara pasti. Dari 10 kali percobaan pada persamaan 4 (Gambar 4a), metode FPAC menemukan jumlah solusi yang berbeda-beda pada

setiap kali percobaan. Solusi terendah terjadi pada percobaan ke-2 yaitu 46 solusi yang berasal dari 67 kluster sedangkan solusi tertinggi terjadi percobaan ke-10 yaitu 72 solusi yang berasal dari 126 kluster. Rata-rata solusi yang ditemukan metode FPAC pada persamaan 4 adalah 56 solusi dalam sekali *running.*

Sementara pada persamaan 5 (Gambar 4b), metode FPAC juga menemukan jumlah solusi yang berbeda pada setiap kali percobaan. Dari 10 kali percobaan, jumlah solusi terendah terjadi pada percobaan ke-5 yaitu 63 solusi yang berasal dari 68 kluster potensial, sedangkan jumlah solusi tertinggi terjadi pada percobaan ke-6 yaitu 127 solusi yang berasal dari 154 kluster potensial. Rata-rata solusi yang ditemukan metode FPAC pada persamaan 5 adalah 87 solusi dalam sekali *running.* Perbedaan jumlah solusi pada setiap kali percobaan disebabkan oleh karakteristik dari metode FPA yang merupakan salah satu dari algoritma metaheuristik.



Gambar 5. Perbandingan FPAC dan PSO dalam menemukan semua solusi pada masing-masing *benchmark* persamaan *Diophantine* (kondisi 3) dalam sekali *running*

Pengujian pada *benchmark* persamaan *Diophantine* yang memenuhi kondisi (2), metode FPAC menemukan jumlah solusi yang sama dengan metode ACO dalam [7]. Hal ini terjadi karena persamaan 1, 2, 3, 4 dan 5 memang hanya memiliki 1 solusi. Sementara pada pada *benchmark* persamaan *Diophantine* yang memenuhi kondisi (3), keseluruhan persamaan yang diujikan memiliki lebih dari satu solusi. Metode FPAC dapat menemukan seluruh solusi pada setiap persamaan dalam sekali *running*. Hasil tersebut sama dengan hasil yang ditemukan oleh metode PSO dalam [5] seperti yang terlihat pada gambar 5.

Kumpulan beberapa persamaan *Diophantine* yang memenuhi kondisi (1), (2) atau (3) dapat membentuk sebuah sistem persamaan *Diophantine*. Metode FPAC yang diusulkan dalam penelitian ini dapat diterapkan pada kedua hal tersebut, meskipun persamaan dan sistem persamaan *Diophantine* memiliki karakteristik yang berbeda, Pada kasus 1 yaitu sistem persamaan *Diophantine* yang berdimensi rendah, metode FPAC dapat menemukan semua solusi yang tersedia dalam sekali *running*. Sementara pada Kasus 2 dan 3, yaitu sistem persamaan *Diophantine* yang berdimensi tinggi, metode FPAC juga dapat menemukan semua solusi yang tersedia dalam sekali *running*. Hasil tersebut sama dengan hasil yang ditemukan oleh metode PSO dalam [5].

Berdasarkan evaluasi di atas dapat diketahui bahwa metode FPAC merupakan salah satu metode alternatif yang dapat menyelesaikan permasalahan persamaan *Diophantine* ataupun sistem persamaan *Diophantine* baik yang memiliki solusi tunggal maupun solusi jamak. Metode FPAC terbukti efektif dalam menemukan semua atau sebanyak mungkin solusi yang tersedia dalam daerah pencarian baik pada persamaan *Diophantine* maupun sistem persamaan *Diophantine* dalam sekali *running*.

4. KESIMPULAN DAN SARAN

Berdasarkan hasil penelitian dapat disimpulkan bahwa metode FPAC yang dikembangkan dalam penelitian ini dapat bekerja dengan baik untuk menyelesaikan permasalahan *Diophantine*. Metode FPAC dapat menemukan seluruh solusi atau akar dari persamaan *Diophantine* dalam sekali *running* tanpa dipengaruhi oleh perbedaan jumlah variabel dan pangkat ataupun struktur eksponensial yang membentuk persamaan *Diophantine*. FPAC juga dapat menemukan solusi atau akar dari sistem persamaan *Diophantine* tanpa dipengaruhi oleh jumlah dimensi yang membentuk sistem persamaan *Diophantine*.

DAFTAR PUSTAKA

[1] I. Agmour, M. Bentounsi, N. Achtaich, and Y. El Foutayeni, "Bifurcation and stability of a dynamical system with threshold prey harvesting," *Int. J. Comput. Sci. Math.*, vol. 14, no. 1, pp. 36–53, 2021.

[2] V. O. Osipyan, K. I. Litvinov, R. K. Bagdasaryan, E. P. Lukashchik, S. G. Sinitsa, and A. S. Zhuk, "Development of information security system mathematical models by the solutions of the multigrade diophantine equation systems," *ACM Int. Conf. Proceeding Ser.*, pp. 1–8, 2019, doi: 10.1145/3357613.3357624.

[3] M. R. Nasiri, S. Farhangi, and J. Rodriguez, "Model Predictive Control of a Multilevel CHB STATCOM in Wind Farm Application Using Diophantine Equations," *IEEE Trans. Ind. Electron.*, vol. 66, no. 2, pp. 1213–1223, 2019, doi: 10.1109/TIE.2018.2833055.

[4] R. Radha and G. Janaki, "Applications Of Diophantine Equations In Chemical Reactions And Cryptography," vol. 12, no. 7, pp. 3175–3178, 2021.

[5] O. Pérez, I. Amaya, and R. Correa, "Numerical solution of certain exponential and non-linear Diophantine systems of equations by using a discrete particle swarm optimization algorithm," *Appl. Math. Comput.*, vol. 225, pp. 737–746, 2013, doi: 10.1016/j.amc.2013.10.007.

[6] X.-S. Yang, *Nature-Inspired Optimization Algorithms*, Second. Academic Press, 2021.

[7] S. Abraham, S. Sanyal, and M. Sanglikar, "Finding numerical solutions of diophantine equations using ant colony optimization," *Appl. Math. Comput.*, vol. 219, no. 24, pp. 11376–11387, 2013, doi: 10.1016/j.amc.2013.05.051.

[8] D. Zaitsev, S. Tomov, and J. Dongarra, "Solving linear diophantine systems on parallel architectures," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 5, pp. 1158–1169, 2018, doi: 10.1109/TPDS.2018.2873354.

[9] Y. Man, "A Top-down Approach for Solving Linear Diophantine Equation," in *Lecture Notes in Engineering and Computer Science: Proceedings of the World Congress on Engineering*, 2019, vol. 0958, pp. 5–7.

[10] Z. A. A. Alyasseri, A. T. Khader, M. A. Al-Betar, and O. A. Alomari, "Person identification using EEG channel selection with hybrid flower pollination algorithm," *Pattern Recognit.*, vol. 105, p. 107393, 2020, doi: 10.1016/j.patcog.2020.107393.

[11] P. Singh and N. Mittal, "An efficient localization approach to locate sensor nodes in 3D wireless sensor networks using adaptive flower pollination algorithm," *Wirel. Networks*, vol. 27, no. 3, pp. 1999–2014, 2021, doi: 10.1007/s11276-021-02557-7.

[12] F. B. Ozsoydan and A. Baykasoglu, "Chaos and intensification enhanced flower pollination algorithm to solve mechanical design and unconstrained function optimization problems," *Expert Syst. Appl.*, vol. 184, no. May, p. 115496, 2021, doi: 10.1016/j.eswa.2021.115496.

[13] R. Karim, K. A. Sidarto, and S. Bantun, "Optimasi Fungsi Multimodal Menggunakan Flower Pollination Algorithm Dengan Teknik Clustering," *Techno.Com*, vol. 19, no. 2, pp. 124–134, 2020, doi: 10.33633/tc.v19i2.3216.

[14] V. Aggarwal, "Solving transcendental equations using Genetic Algorithms," pp. 1–12, 2000.

[15] S. Katoch, S. S. Chauhan, and V. Kumar, *A review on genetic algorithm: past, present, and future*, vol. 80, no. 5. Multimedia Tools and Applications, 2021.

[16] X. Li, M. G. Epitropakis, K. Deb, and A. Engelbrecht, "Seeking Multiple Solutions: An Updated Survey on Niching Methods and Their Applications," *IEEE Trans. Evol. Comput.*, vol. 21, no. 4, pp. 518–538, 2017, doi: 10.1109/TEVC.2016.2638437.

[17] K. A. Sidarto, A. Kania, and N. Sumarti, "Finding multiple solutions of multimodal optimization using spiral optimization algorithm with clustering," *Mendel*, vol. 23, no. 1, pp. 95–102, 2017, doi: 10.13164/mendel.2017.1.095.

[18] Y. Jia, S. Wang, L. Liang, Y. Wei, and Y. Wu, "A Flower Pollination Optimization Algorithm Based on Cosine," *Sensors*, vol. 23, no. 2, pp. 1–19, 2023.