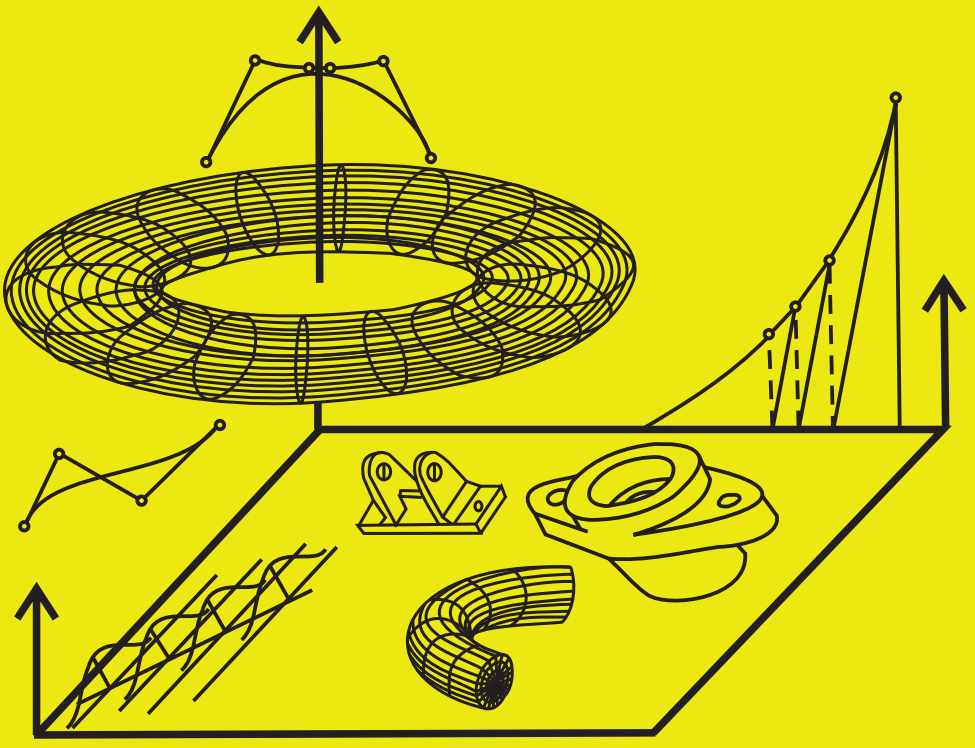
Volume 19 Nomor 1, Maret 2019 ISSN 1411-6669

MIMS

MAJALAH ILMIAH

Matematika dan Statistika



DITERBITKAN OLEH:
JURUSAN MATEMATIKA
FMIPA - UNIVERSITAS JEMBER

MAJALAH ILMIAH
Matematika dan Statistika

Editor in Chief : Kiswara Agung Santoso
Managing Editor : Kristiana Wijaya

Editorial Board:
Firdaus Ubaidillah
Agustina Pradjaningsih
Ahmad Kamsyakawuni
Dian Anggraeni

Reviewer:
Kusno, Universitas Pendidikan Mandalika Mataram
Mardjono, FMIPA, Universitas Brawijaya
Basuki Widodo, FMIPA, Institut Teknologi Sepuluh Nopember
Retantyo Wardoyo, FMIPA, Universitas Gadjah Mada
Slamin, FASILKOM, Universitas Jember
Herry Suprajitno, FMIPA, Universitas Airlangga

Layout and Editor:
Ikhsanul Halikin

Desain Grafis:
Yoyok Yulianto

Alamat Redaksi:
Jurusan Matematika FMIPA – Universitas Jember
Jalan Kalimantan No 37 Kampus Tegalboto Jember 68121
Telp. : (0331) 334293
E-mail: mims.fmipa@unej.ac.id
Website: https://jurnal.unej.ac.id/index.php/MIMS/index

Diterbitkan oleh : Jurusan Matematika – FMIPA Universitas Jember.
Tahun pertama terbit : Oktober 2000
Jumlah terbit : Dua kali setahun pada bulan Maret dan September
Gambar cover depan : rancang bangun geometri, iterasi dan regresi

Majalah Ilmiah Matematika dan Statistika	Volume 19 Nomor 1	Halaman: 1 – 52	Jember, Maret 2019	ISSN 1411-6669

MAJALAH ILMIAH
Matematika dan Statistika

Volume 19 Nomor 1, Maret 2019 **ISSN 1411-6669**

Daftar Isi

Majalah Ilmiah Matematika dan Statistika Volume 19 Nomor 1, 2019, 17 – 28
https://jurnal.unej.ac.id/index.php/MIMS/index ISSN 1411-6669

PENYANDIAN CITRA MENGGUNAKAN ALGORITMA *4D PLAYFAIR CIPHER* DENGAN PEMBANGKITAN KUNCI MODIFIKASI *LINEAR FEEDBACK SHIFT REGISTER*

(***Image Encoding Used 4D Playfair Cipher Algorithm with Key Generation Modification of Linear Feedback Shift Register***)

Rivi Tri Rahayu, **Abduh Riski**, **Ahmad Kamsyakawuni**

Jurusan Matematika, Fakultas MIPA, Universitas Jember
Jl. Kalimantran 37 Jember 68121, Indonesia
E-mail: rivirahayu16996@gmail.com, {riski, kamsyakawuni}.fmipa@unej.ac.id

Abstract. The fast development of sophisticated technology make it easier for someone to send a message to other but can also make it easier for third parties to sabotage the content of the message, so a technique called cryptography is needed to secure the message. Image encoding is one of the techniques for securing messages in cryptography. In enhancing security in image encoding, this study discusses about Playfair Cipher, 3D Playfair Cipher and 4D Playfair Cipher with key generation using LFSR Modification. The encryption process using 4D Playfair Cipher with key generation using LFSR Modification visually produces cipher image that is different from the original image compared to using Playfair Cipher and 3D Playfair Cipher. In the decryption process using Playfair Cipher, 3D Playfair Cipher and 4D Playfair Cipher-Modification LFSR can return cipher image to its original image. The result of the study shows that the proposed method can be used to secure the message.

Keywords: Playfair Cipher, 3D Playfair Cipher, 4D Playfair Cipher, LFSR
MSC 2010: 94A60

1. Pendahuluan

Perkembangan teknologi cepat membuat semakin mudahnya seseorang mengirimkan suatu pesan kepada orang lain tetapi juga dapat membuat semakin mudahnya pihak ketiga menyabotase isi pesan tersebut maka dibutuhkan suatu teknik yang dinamakan kriptografi untuk mengamankan isi pesan. Kriptografi merupakan suatu ilmu untuk melindungi atau menyembunyikan pesan agar tidak diketahui oleh pihak ketiga dengan cara mengubah isi pesan asli menjadi kode – kode yang sulit dimengerti maknanya.Terdapat beberapa istilah dalam kriptografi seperti enkripsi dan dekripsi. Enkripsi merupakan sebuah proses merubah pesan yang dapat dimengerti (*plaintext*) menjadi sebuah kode yang tidak dapat dimengerti (*ciphertext*) dan dekripsi merupakan sebuah proses kebalikan dari enkripsi [3]. Jenis kriptografi dibedakan menjadi dua macam yaitukriptografi klasik dan kriptografi modern. Salah satu perbedaan algoritma kriptografi klasik dan algoritma kriptografi

modern adalah algoritma kriptografi klasik menggunakan metode substitusi dan transposisi sedangkan algoritma kriptografi modern memanfaatkan operasi komputer digital dalam proses enkripsi / dekripsi [1].

Pada penelitian ini membahas tentang peningkatan keamanan pada penyandian citra menggunakan algoritma *4D Playfair Cipher* dan pembangkitan matriks kuncinya menggunakan konsep algoritma *Linear Feedback Shift Register* (LFSR) yang telah dimodifikasi. Konsep algoritma *Linear Feedback Shift Register* (LFSR) yang digunakan yaitu dengan menambahkan langkah *Shift*, Logika XOR dan XNOR. Sedangkan dua langkah modifikasi yang dilakukan pada LFSR yaitu dengan penambahan bilangan dan rotasi.

Citra

Citra merupakan gambar pada bidang dua dimensi. Citra terbagi menjadi dua antara lain, citra bersifat analog dan citra yang bersifat digital. Kualitas sebuah citra selalu dikaitkan dengan resolusi dalam intensitas warna. Resolusi citra menyatakan ukuran panjang kali lebar sebuah citra yang dinyatakan dalam satuan piksel. Semakin tinggi resolusi sebuah citra berarti semakin banyak jumlah piksel dan semakin tinggi kedalam intensitas berarti semakin banyak pula jumlah pikselnya. Citra dapat diolah menggunakan komputer, citra dapat dikatakan citra digital apabila suatu citra tersebut direpresentasikan secara numerik atau nilai-nilai diskrit melalui proses digitalisasi. Digitalisasi merupakan representasi citra dari fungsi kontinu menjadi fungsi diskrit. Secara umum, citra digital berbentuk persegi panjang dengan ukuran dimensi dapat dinyatakan sebagai tinggi × lebar atau lebar × panjang. Citra digital yang berukuran $N \times M$ dinyatakan dengan matriks berukuran N baris dan M kolom sebagai berikut:

$$f(x,y) = \begin{bmatrix} f(1,1) & \cdots & f(1,M) \\ \vdots & \ddots & \vdots \\ f(N,1) & \cdots & f(N,M) \end{bmatrix}$$

Indeks baris (x) dan indeks kolom (y) menyatakan koordinat suatu titik pada citra sedangkan $f(x,y)$ merupakan intensitas (derajat keabuan) pada titik (x,y). Masing-masing elemen pada citra digital (elemen pada matriks) disebut piksel [2].

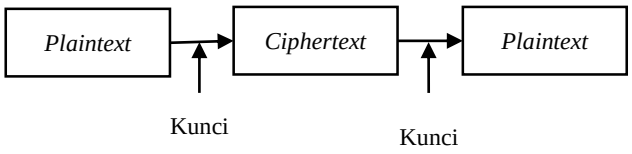
Kode ASCII

Kode ASCII (*American Standard Code for Information Interchange*) merupakan suatu standard internasional dalam kode huruf dan simbol seperti Hex dan Unicode tetapi ASCII lebih bersifat universal. Contohnya, 47 untuk karakter "/". Kode ASCII pada dasarnya memiliki komposisi bilangan biner sebanyak 8 bit. Dimulai dari 0000 0000 hingga 1111 1111. Total kombinasi yang dihasilkan pada Kode ASCII adalah 256, dimulai dari kode 0 hingga 255 dalam sistem bilangan Desimal.

Kriptografi

Kriptografi berasal dari Bahasa Yunani yaitu *cryptos* yang berarti rahasia dan *graphein*

yang berarti menulis, apabila kedua istilah digabungkan akan bermakna menulis rahasia. Kriptografi sendiri merupakan ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi, seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data, namun tidak semua aspek keamanan informasi dapat diselesaikan dengan kriptografi. Terdapat beberapa istilah dalam kriptografi seperti enkripsi dan dekripsi. Enkripsi merupakan sebuah proses merubah pesan yang dapat dimengerti (*plaintext*) menjadi sebuah kode yang tidak dapat dimengerti (*ciphertext*) dan dekripsi merupakan sebuah proses kebalikan dari enkripsi.



Gambar 1. Proses enkripsi dan dekripsi

Algoritma kriptografi dibedakan menjadi dua, yaitu yaitu algoritma kriptografi klasik dan algoritma kriptografi modern. Algoritmakriptografi klasik menggunakan metode substitusi dan transposisi sedangkan algoritma kriptografi modern memanfaatkan operasi komputer digital dalam proses enkripsi dan dekripsi [3].

Playfair Cipher

Playfair menggunakan kunci dalam matriks 5×5 yang berisi 25 huruf alfabet dan mengganti huruf J menjadi huruf I yang ada didalam alfabet. *Playfair Cipher* merupakan salah satu kriptografi klasik yang penyandiannya menggunakan subtitusi. Pada algoritma ini dibutuhkan dua huruf yang berpasangan (digram) dalam mengenkripsi dan mendekripsi pesan [6].

Beberapa aturan dalam membuat sandi pada *playfair cipher* [6], yaitu:

a. Jika dua huruf *plaintext* berada pada satu baris kunci yang sama maka setiap huruf diganti dengan huruf yang berada disebelah kanannya.
b. Jika dua huruf *plaintext* berada pada satu kolom kunci yang sama maka setiap huruf diganti dengan huruf yang berada dibawahnya.
c. Jika huruf *plaintext* berada terbalik dengan tabel maka sandi yang dihasilkan akan dibaca terbalik, dengan kata lain yang semula dibacanya kiri ke kanan menjadi kanan ke kiri.
d. Jika dua huruf tidak pada baris yang sama atau kolom yang sama, maka huruf pertama diganti dengan huruf pada perpotongan baris huruf pertama dengan kolom huruf kedua. Huruf kedua diganti dengan huruf pada titik sudut keempat dari persegi panjang yang dibentuk dari tiga huruf yang digunakan.
e. Jika terdapat huruf yang ganda pada *plaintext* harus disisipkan huruf X atau Z dan apabila *plaintext* memiliki jumlah huruf yang ganjil maka harus disisipkan juga huruf X atau Z pada akhir *plaintext*.

3D Playfair Cipher

3D Playfair Cipher adalah pengembangan algoritma kriptografi klasik yang membutuhkan tiga huruf berpasangan (trigram) selama proses enkripsi dan dekripsi. Pada algoritma ini menggunakan formasi matriks kunci berukuran 4 × 4 × 4 yang berisi 26 huruf alfabet, 10 angka, dan beberapa simbol khusus. Pada algortima ini memiliki matriks kunci terdiri dari empat baris, empat kolom, dan empat tingkat (lihat Tabel 1). Kunci yang digunakan pada algoritma ini tidak boleh berulang [9].

Tabel 1. Matriks kunci *3D playfair cipher*

TINGKAT 1			
0	1	2	3
4	5	6	7
8	9	A	B
C	D	E	F

TINGKAT 2			
G	H	I	J
K	L	M	N
O	P	Q	R
S	T	U	V

TINGKAT 3			
W	X	Y	Z
!	“	=	$
%	&	‘	(
)	*	+	,

TINGKAT 4			
-	.	/	:
;	<	=	>
?	@	[	\
]	^	_	|

Selama proses enkripsi, pesan akan dipecah menjadi trigram (pasangan yang tediri dari tiga huruf). Huruf tambahan X dan Z digunakan untuk memenuhi trigram, X atau Z ditambahkan jika tersisa satu tempat kosong pada pesan, X dan Z ditambahkan jika terdapat dua tempat kosong. Contohnya, NEGARAKU akan diubah menjadi {NEG}, {ARA} dan {KUX}. Penggantian huruf dalam trigram akan diganti oleh pesan yang sehubungan dengan posisi huruf dalam trigram di baris, kolom, tingkat dengan cara melingkar (lihat Tabel 2). Proses enkripsi menggunakan model *circular* dapat dilihat pada [9].

Tabel 2. Proses enkripsi

Trigram *Plaintext*	Trigram *Plaintext*			Trigram *Ciphertext*
	Kar-1	Kar-2	Kar-3	
Kar-1	Baris	Kolom	Tingkat	Kar-1
Kar-2	Tingkat	Baris	Kolom	Kar-2
Kar-3	Kolom	Tingkat	Baris	Kar-3

Penggantian huruf untuk tujuan dekripsi mengikuti mode *circular*, hanya urutannya berubah yaitu baris, tingkat, kolom huruf dalam trigram *plaintext* [9], seperti ditunjukkan pada Tabel 3.

Tabel 3. Proses dekripsi

Trigram	Trigram Ciphertext			Trigram
Ciphertext	Kar-1	Kar-2	Kar-3	*Plaintext*
Kar-1	Baris	Tingkat	Kolom	Kar-1
Kar-2	Kolom	Baris	Tingkat	Kar-2
Kar-3	Tingkat	Kolom	Baris	Kar-3

4D Playfair Cipher

4D Playfair Cipher adalah pengembangan algoritma kriptografi klasik yang membutuhkan empat huruf berpasangan (*quartets*) selama proses enkripsi dan dekripsi. Pada algoritma ini menggunakan formasi matriks kunci berukuran $2 \times 2 \times 13 \times 5$ yang berisi 260 bilangan dari 0 sampai 259. Pada algoritma ini nilai 0 sampai 255 menampilkan bilangan yang dapat menyimpan seluruh nilai pada kode ASCII. Bilangan 256 sampai 258 digunakan atau disisipkan apabila jumlah huruf *plaintext* bukan kelipatan empat. Nilai 259 digunakan hanya selama proses substitusi [1].

4DPlayfair Cipher memiliki empat langkah utama, antara lain:

a. Buatlah kunci rahasia dengan barisan bilangan antara 0 sampai 259. Contohnya : (10 20 30 40 50 60 70 80).
b. Bilangan yang terdapat di dalam kunci rahasia tidak boleh berulang atau terdapat bilangan yang sama.
c. Masukkan kunci pada langkah sebelumnya kedalam matriks kunci berukuran $2 \times 2 \times 13 \times 5$ yang disediakan oleh *4DPlayfair Cipher* dengan mengisi berurutan dimulai dari D1_P1, D1_P2, D2_P1 sampai D2_P2.
d. Sisipkan pada tempat yang tersisa dalam matriks dengan bilangan yang tidak terdapat didalam kunci dimulai dari bilangan 0 sampai 259 mengkuti aturan yang diberika pada langkah tiga [1].
   Matriks kunci pada *4DPlayfair Cipher* dapat dilihat pada Tabel 4.

Proses enkripsi menggunakan metode *circular*, seperti pada Tabel 5. Pergantian huruf dalam *quartets* akan diganti oleh pesan sandi yang sehubungan dengan posisi yang terdapat dalam *quartets* pada baris (*row* / R), kolom (*coloumn* / C), arah (*direction* / D), dan kerangka (*plane* / P*)* dengan metode melingkar [1].

Tabel 4. Matriks *4Dplayfair cipher*

D1_P1

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24
25	26	27	28	29
30	31	32	33	34
35	36	37	38	39
40	41	42	43	44
45	46	47	48	49
50	51	52	53	54
55	56	57	58	59
60	61	62	63	64

D1_P2

65	66	67	68	69
70	71	72	73	74
75	76	77	78	79
80	81	82	83	84
85	86	87	88	89
90	91	92	93	94
95	96	97	98	99
100	101	102	103	104
105	106	107	108	109
110	111	112	113	114
115	116	117	118	119
120	121	122	123	124
125	126	127	128	129

D2_P1

130	131	132	133	134
135	136	137	138	139
140	141	142	143	144
145	146	147	148	149
150	151	152	153	154
155	156	157	158	159
160	161	162	163	164
165	166	167	168	169
170	171	172	173	174
175	176	177	178	179
180	181	182	183	184
185	186	187	188	189
190	191	192	193	194

D2_P2

195	196	197	198	199
200	201	202	203	204
205	206	207	208	209
210	211	212	213	214
215	216	217	218	219
220	221	222	223	224
225	226	227	228	229
230	231	232	233	234
235	236	237	238	239
240	241	242	243	244
245	246	247	248	249
250	251	252	253	254
255	256	257	258	259

Tabel 5. Proses enkripsi *4Dplayfair cipher*

Plaintext Quartet	*Plaintext Quartet*				*Ciphertext Quartet*
	Kar-1	Kar-2	Kar-3	Kar-4	
Kar-1	R	C	D	P	Kar-1
Kar-2	P	R	C	D	Kar-2
Kar-3	D	P	R	C	Kar-3
Kar-4	C	D	P	R	Kar-4

Penggantian huruf untuk tujuan dekripsi mengikuti mode *circular* seperti pada proses enkripsi, hanya urutannya berubah baris (*row* / R), kerangka (*plane* / P), arah (*direction* / D) dan kolom (*coloumn* / C) dengan metode melingkar (circular) [1].

Tabel 6. Proses enkripsi *4DPlayfair Cipher*

Ciphertext Quartet	*Ciphertext Quartet*				*Plaintext Quartet*
	Kar-1	Kar-2	Kar-3	Kar-4	
Kar-1	R	P	D	C	Kar-1
Kar-2	C	R	P	D	Kar-2
Kar-3	D	C	R	P	Kar-3
Kar-4	P	D	C	R	Kar-4

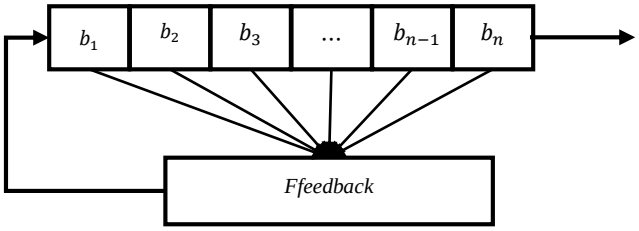
Linear Feedback Shift Register (LFSR)

Linear Feedback Shift Register (LFSR) merupakan salah satu algoritma yang dapat digunakan untuk membangkitkan deretan bilangan biner secara acak dalam pembuatan kunci pada kriptografi. LFSR membangkitkan deretan bilangan dengan menggunakan operasi XOR dan XNOR. Pada proses XOR nilai awal bit yang ada pada register dengan panjang tertentu tergantung dari derajat polinomial yang digunakan. LFSR adalah *shift register* yang bit masukannya merupakan fungsi umpan-balik dari bentuk sebelumnya. Periode maksimum LFSR dengan n-bit memiliki rumus $2^n - 1$ [7].

Bentuk umum dari *Linear Feedback Shift Register* (LFSR) dapat didefinisikan, sebagai berikut:

$$r_{i+1}(b_0, b_1, b_2, \dots, b_{n-1}) = f(b_0, b_1, b_2, \dots, b_{n-1})$$

dimana f adalah fungsi *Linear Feedback Shift Register* (LFSR).
Pembangkitan fungsi umpan balik LFSR dapat dibangkitkan seperti Gambar 2.



Gambar 2. LFSR dengan n-bit

Modifikasi *Linear Feedback Shift Register* (LFSR) merupakan pengembangan algoritma *Linear Feedback Shift Register* (LFSR) dimana pada modifikasinya urutan bilangan acak yang dihasilkan dari perhitungan *Linear Feedback Shift Register* (LFSR) selanjutnya dirotasi sejauh lima ke kiri.

Logika XOR dan XNOR

XOR merupakan singkatan dari *Exclusive*-OR. Logika operasi XOR akan menghasilkan keluaran bernilai Benar jika dan hanya jika salah satu dari nilai input bernilai Benar. Jika kedua input bernilai Benar maka keluaran akan bernilai Salah [8]. Pada operasi logika 0 merepresentasikan logika bernilai Salah dan 1 merepresentasikan logika bernilai Benar. Tabel 7 menunjukkan tabel kebenaran dari operasi XOR.

Tabel 7. Tabel kebenaran operasi XOR

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

XNOR merupakan *invers* dari XOR. Jika pada XOR nilai Benar didapat jika dan hanya jika salah satu input saja yang bernilai Benar. Pada XNOR hasil Benarhanya didapat jika kedua nilai ini menunjukkan nilai yang sama (Benar-Benar atau Salah-Salah). Jika salah satu input saja yang bernilai Benar maka keluarannya akan bernilai Salah [8]. Tabel 8 menunjukkan tabel kebenaran dari logika XNOR.

Tabel 8. Tabel kebenaran operasi XNOR

A	B	A XNOR B
0	0	1
0	1	0
1	0	0
1	1	1

Analisis Histogram

Analisis histogram merupakan salah satu analisis yang dapat digunakan untuk melihat apakah hasil dari proses enkripsi aman dan tahan terhadap serangan-serangan kriptanalisis. Di dalam bidang pengolahan citra histogram memperlihatkan distribusi nilai *pixel* di dalam sebuah citra. Histogram digunakan penyerang untuk melakukan kriptanalisis dengan memanfaatkan frekuensi kemunculan *pixel* di dalam histogram. Penyerang berharap nilai *pixel* yang sering muncul di dalam *plainimage* berkorelasi dengan nilai *pixel* yang sering muncul di dalam *cipherimage*.Histogram *cipherimage* dan *plainimage* seharusnya berbeda secara signifikan atau secara statistik tidak memiliki kemiripan [5].

Analisis histogram yang digunakan adalah pengujian X^2 dari gambar yang telah terenkripsi. Persamaan X^2 dari gambar terenkripsi dari dimensi $m \times n$ sebagai berikut.

$$X^2 = \sum_{i=0}^{255} \frac{(v_i - v_0)^2}{v_0}$$

dimana v_i adalah frekuensi yang diamati dari nilai piksel $i(0 \leq i \leq 255)$ dan v_0 merupakan frekuensi yang diharapkan dari nilai piksel i, jadi $v_0 = \frac{m \times n}{256}$. Sehingga semakin kecil hasil pengujian X^2 maka tingkat keseragaman dalam histogram semakin merata dan hasil dari enkripsi semakin aman dan berlaku sebaliknya [2].

Analisis Diferensial

Analisis Diferensial digunakan untuk menguji pengaruh perubahan setiap *pixel* pada citra yang terenkripsi. Terdapat dua indikator pengukuran umum yang digunakanpada analisis ini yaitu *Number of Pixel Change Rate* (NPCR) dan *Unified Avarage Changing Intensity* (UACI). Adapun perhitungan NPCR didefinisikan sebagai berikut:

$$NPCR = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{o} \frac{d_{i,j,k}}{T} \right) \times 100\%$$

dimana m, n dan o adalah lebar, tinggi, dan kedalaman citra dan digunakan untuk menghitung T yang merupakan jumlah total *pixel* pada *cipherimage* sedangkan $d_{i,j,k}$ melambangkan derajat keabuan ditentukan sebagai berikut:

$$d_{i,j} = \begin{cases} 0, & jika\ c_{i,j,k}^{(1)} = c_{i,j,k}^{(2)} \\ 1, & jika\ c_{i,j,k}^{(1)} \neq c_{i,j,k}^{(2)} \end{cases}$$

dimana $c_{i,j,k}^{(1)}$ dan $c_{i,j,k}^{(2)}$ merupakan nilai derajat keabuandari baris i, kolom j dan kanal k dari citra $c^{(1)}$ dan citra $c^{(2)}$.

Sedangkan perhitungan UACI didefinisikan sebagai berikut :

$$UACI = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{o} \frac{\left| c_{i,j,k}^{(1)} - c_{i,j,k}^{(2)} \right|}{F \cdot T} \right) \times 100\%$$

dimana F menunjukkan nilai *pixel* terbesar yang kompatibel dengan *chiperimage* [2].

2. Metodologi

Data yang digunakan pada penilitian ini adalah citra yang disebut sebagai *plainimage*. Data yang digunakan untuk pengujian pada penelitian ini sebanyak 4 citra. Langkah-langkah penelitian yang dilakukan adalah pertama mengumpulkan literatur yang berkaitan dengan *Playfair Cipher*, *3D Playfair Cipher* dan *4D Playfair Cipher*-Modifikasi LFSR. Kedua melakukan percobaan enkripsi dan dekripsi menggunakan *Playfair Cipher*, *3D Playfair Cipher* dan *4D Playfair Cipher*-Modifikasi LFSR. Kemudian pembuatan program enkripsi dan dekripsi pada citra menggunakan *software* MATLAB R2015b dan disimulasikan program enkripsi dan dekripsi yang telah dibuat pada *software* MATLAB R2015b.Selanjutnya menganalisis hasil keamanan citra dengan membandingkan hasil perhitungan dari histogram, NPCR dan UACI. Sehingga dapat dianalisis pengaruh pertambahan pembangkitan kunci algoritma Modifikasi LFSR terhadap peningkatan keamanan *cipher image* yang dihasilkan.

3. Hasil dan Pembahasan

Simulasi dilakukan dengan mengenkripsi 2 citra *Grayscale* dan 2 citra RGB. Misal kunci yang digunakan pada saat enkripsi dan dekripsi citra menggunakan *Playfair Cipher*, *3D Playfair Cipher* dan *4D Playfair Cipher*-Modifikasi LFSR adalah “MTK2015a”.

a. Hasil Perhitungan X^2 Menggunakan *Playfair Cipher*, *3D Playfair Cipher* dan *4D Playfair Cipher*-Modifikasi LFSR.

Tabel 9. Hasil perhitungan X^2

No	Data Penelitian	X^2
1	Citra 1	*Playfair Cipher* =*12214,97* *3D Playfair Cipher* =10930,61 *4D Playfair Cipher*-Modifikasi LFSR=2989,12
2	Citra 2	*Playfair Cipher* =9194,32 *3D Playfair Cipher* =8583,62 *4D Playfair Cipher*-Modifikasi LFSR=2792,72
3	Citra 3	*Playfair Cipher* =*5662,30* *3D Playfair Cipher* =5473,66 *4D Playfair Cipher*-Modifikasi LFSR =4137,20
4	Citra 4	*Playfair Cipher* =172950,88 *3D Playfair Cipher* =80162,79 *4D Playfair Cipher*-Modifikasi LFSR=*7249,30*

b. Hasil Analisis Diferensial Menggunakan *Playfair Cipher*, *3D Playfair Cipher* dan *4D Playfair Cipher*-Modifikasi LFSR

Tabel 10. Hasil analisis diferensial NPCR

No	Data Penelitian	NPCR (%)		
		Playfair Cipher	*3D Playfair Cipher*	*4D Playfair Cipher*-Modifikasi LFSR
1	Citra 1	99,24	99,34	87,32
2	Citra 2	99,34	99,32	87,77
3	Citra 3	99,40	99,35	92,83
4	Citra 4	98,95	99,14	81,35

Tabel 11. Hasil analisis diferensial UACI

No	Data Penelitian	UACI (%)		
		Playfair Cipher	*3D Playfair Cipher*	*4D Playfair Cipher*-Modifikasi LFSR
1	Citra 1	44,67	44,93	31,52
2	Citra 2	42,08	42,04	31,34
3	Citra 3	21,76	22,13	26,87
4	Citra 4	25,62	24,66	22,84

Hasil penelitian menunjukkan bahwa proses enkripsi citra menggunakan *Playfair* Cipher terlihat acak. Pada proses enkripsi menggunakan *3D Playfair* Cipher juga terlihat acak. Sedangkan proses enkripsi menggunakan *4D Playfair* Cipher terlihat acak (tidak berpola) sehingga sulit untuk menduga citra aslinya. Proses enkripsi citra menggunakan *Playfair Cipher*, *3D Playfair Cipher* dan *4D Playfair Cipher*-Modifikasi LFSR juga diterapkan melalui program MATLAB R2015b berdasarkan metode yang diajukan oleh penulis.

Proses dekripsi merupakan kebalikan dari proses enkripsi. Pada proses dekripsi dilakukan juga tiga perlakuan yaitu *Playfair Cipher*, *3D Playfair Cipher* dan *4D Playfair Cipher*-Modifikasi LFSR. Hasil yang diperoleh dari proses dekripsi menggunakan ketiga perlakuan diatas dapat mengembalikan *cipherimage* menjadi citra asli (*plainimage*). Proses dekripsi citra menggunakan *Playfair Cipher*, *3D Playfair Cipher* dan *4D Playfair Cipher*-Modifikasi LFSR juga diterapkan melalui program MATLAB R2015b berdasarkan metode yang diajukan oleh penulis.

Hasil dari analisis histogram pada ketiga perlakuan menunjukkan bahwa *4D Playfair Cipher*-Modifikasi LFSR menghasilkan histogram yang lebih merata dengan hasil perhitungan X^2 lebih kecil dibandingkan hasil histogram menggunakan *Playfair Cipher* dan *3D Playfair Cipher*. Artinya, hasil dari proses enkripsi dengan menggunakan *4D Playfair Cipher*-Modifikasi LFSR akan lebih kuat terhadap serangan analisis tipe statistik dibandingkan dengan menggunakan *Playfair Cipher* dan *3D Playfair Cipher*.

Hasil perhitungan NPCR yang diperoleh menggunakan *4D Playfair Cipher*-Modifikasi LFSR adalah sebesar 81,35% hingga 92,83% sedangkan hasil perhitungan NPCR yang diperoleh menggunakan *Playfair Cipher* adalah sebesar 98,95% hingga99,40% dan hasil perhitungan NPCR yang diperoleh menggunakan *3D Playfair Cipher* adalah sebesar 99,14% hingga 99,42%. Tabel 11 dijelaskan bahwa hasil perhitungan UACI yang diperoleh menggunakan *4D Playfair Cipher*-Modifikasi LFSR adalah sebesar 22,84% hingga 31,52% sedangkan hasil perhitungan UACI yang diperoleh menggunakan *Playfair Cipher* adalah sebesar 21,76% hingga 44,67% dan hasil perhitungan UACI yang diperoleh menggunakan *3D Playfair Cipher* adalah sebesar 22,13% hingga 44,93%. Berdasarkan hasil yang telah diperoleh, semakin besar suatu hasil perhitungan NPCR dan UACI maka semakin kuat suatu citra hasil enkripsi terhadap serangan diferensial. Citra yang menghasilkan nilai NPCR dan UACI di bawah minimum masih kuat terhadap serangan diferensial, dibuktikan secara visual bahwa citra terlihat tidak berpola. Namun citra yang menghasilkan nilai NPCR dan UACI di bawah minimum rentan terhadap serangan diferensial apabila citra yang dihasilkan dari proses enkripsi masih terlihat jelas polanya.

4. Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, didapat beberapa kesimpulan sebagai berikut:

a. Pembangkitan kunci menggunakan Modifikasi *Linear Feedback Shift Register* (LFSR) dapat menghasilkan deretan bilangan secara acak tanpa terdapat perulangan kunci pada deretan bilangan.
b. Proses enkripsi menggunakan algoritma *4D Playfair Cipher* dan Modifikasi *Linear Feedback Shift Register* (LFSR) menghasilkan *cipherimage* berbeda dari citra asli

secara visual dan proses dekripsi dapat mengembalikan *cipherimage* kedalam citra aslinya. Proses enkripsi menggunakan algoritma *Playfair Cipher* maupun *3D Playfair Cipher* menghasilkan *cipherimage* berbeda dengan citra asli secara visual dan proses dekripsi dapat mengembalikan *cipherimage* kedalam citra aslinya.

c. Berdasarkan perbandingan antara hasil perhitungan dari histogram, NPCR, UACI, dan *cipherimage* yang dihasilkan. Tingkat keamanan hasil penyandian citra menggunakan *4D Playfair Cipher* dengan pembangkitan kunci Algoritma Modifikasi LFSR lebih kuat dibandingkan dengan hasil penyandian citra menggunakan *Playfair Cipher* dan *3D Playfair Cipher*, dapat dilihat pada hasil histogram yang seragam, hasil perhitungan X^2 lebih kecil, serta *cipherimage* yang dihasilkan acak dan merata daripada menggunakan *Playfair Cipher* dan *3D Playfair Cipher.*

Daftar Pustaka

[1] Bhat, K., D. Mahto., dan D. K. Yadav. (2017). A Novel Approach to Information Four Dimensional (4D) Playfair Cipher Fused With Linear Feedback Shift Register. *Indian Journal of Computer Science and Engineering (IJCSE).* 8 (1):15-32.

[2] Boriga, R. E., A.C. Dascalescu, dan A.V. Diaconu. (2014). A New Fast Image Encryption Scheme Based on 2D Chaotic Maps. *IAENG International Journal of Computer* (*IJSC*). 41(4):1-10.

[3] Donnaro, D. P. (2018). Implementasi Teknik Playfair Cipher untuk Penyembunyian Teks Terenkripsi pada Citra dengan Metode End of File. *Skripsi*. Jember: Universitas Jember.

[4] Munir, R. (2002). *Diktat Kuliah Pengolahan Citra*. Bandung: Departemen Teknik Informatika ITB.

[5] Munir, R. (2012). Analisis Keamanan Algoritma Enkripsi Citra Digital Menggunakan Kombinasi Dua Chaos Map dan Penerapan Teknik Selektif. *Juti*. 10(2):89–95.

[6] Nurkifli, E. H. (2014). Modifikasi Algoritma Playfair dan Menggabungkan dengan Linear Feedback Shift Register (LFSR). *SENTIKA.* ISSN: 2089-9813. 366-271.

[7] Pramudianto, A. D. dan Rino. (2012). Penggunaan Polinomial untuk Stream Key Generator pada Algoritma Stream Ciphers Berbasis Feedback Shift Register. *Seminar Nasional Matematika dan Pendidikan Matematika*. ISBN : 978-979-16353-8-7. Yogyakarta: Universitas Negeri Yogyakarta.

[8] Putra, D. (2010). *Pengolahan Citra Digital*. Yogyakarta:Andi Offset.

[9] Singh, S., Jain, R., Deep, P. dan Agarwal, S. (2015). Developing Mobile Messege Security Application Using 3D Playfair Cipher Algorithm. *International Coference on Advances in Computer Engineering and Applications (ICACEA).* 838 – 841.

Majalah Ilmiah Matematika dan Statistika Volume 19 Nomor 1, 2019, 39 – 52
https://jurnal.unej.ac.id/index.php/MIMS/index **ISSN**1411-6669

PENERAPAN *DRAGONFLY OPTIMIZATION ALGORITHM* (DOA) PADA PERMASALAHAN *MULTIPLE CONSTRAINTS BOUNDED KNAPSACK*

(Studi Kasus: Kerajinan Bambu Hitam Desa Pujerbaru Kecamatan Maesan Kabupaten Bondowoso)

(Application of Dragonfly Optimization Algorithm (DOA) in Multiple Constraints Bounded Knapsack Problems (Case Study: Black Bamboo Crafts Pujerbaru Village Maesan District Bondowoso Regency))

Laylatul Febriana Nilasari, Kiswara Agung Santoso, Abduh Riski

Jurusan Matematika, Fakultas MIPA, Universitas Jember
Jl. Kalimantan 37 Jember 68121, Indonesia
E-mail: laylatulfebriananilasari@gmail.com, {kiswara, riski}.fmipa@unej.ac.id

Abstract. Optimization is very useful in almost all fields in running a business effectively and efficiently to achieve the desired results. This study solves the problem of multiple constraints bounded knapsack by implementing DOA. The problem of multiple constraints bounded knapsack has more than ones constraint with objects that are entered into the storage media, the dimensions can be partially or completely included, but the number of objects is limited. The purpose of this study is to determine the results of using DOA to solve multiple constraits bounded knapsack and the effectiveness of DOA compared to the results of the Simplex method. The data used in this study are primary data. There are ten parameters to be tested, namely population parameters, maximum iteration, s, a, c, f, e and range. The trial results of the ten parameters show that the best value of the parameters is neither too large nor too small. If the best value is too large then the position of the dragonfly will be randomized so that it is not clear the position of the dragonfly and if it is too small the best value then the change is not visible. In addition, based on the results of the final experiment it can be seen that DOA is less effective in solving multiple constraints bounded knapsack problems, because of many experiments there is no solution similar to Simplex. DOA approach to optimal, seen from a small deviation.

Keywords: DOA, Knapsack, Multiple constraints bounded knapsack problem.
MSC2010: 9004

1. Pendahuluan

Optimasi adalah salah satu disiplin ilmu dalam matematika yang fokus untuk mendapatkan nilai minimum atau maksimum secara sistematis dari suatu fungsi, peluang maupun pencarian nilai lainnya dalam berbagai kasus. Optimasi sangat berguna di hampir segala bidang dalam menjalankan usaha secara efektif dan efisien untuk mencapai target hasil yang ingin dicapai, sehingga optimasi sangat penting dalam persaingan di dunia industri yang sudah sangat ketat di segala bidang yang ada.

Masalah *Knapsack* merupakan suatu permasalahan yang berhubungan dengan penyimpanan objek ke dalam media penyimpanan yang terbatas. *Knapsack* merupakan suatu kantong atau tempat yang digunakan untuk memuat suatu objek. Kantong atau tempat tersebut hanya dapat menyimpan beberapa objek saja dengan ketentuan total ukuran objek tersebut lebih kecil atau sama dengan ukuran kapasitasnya. Permasalahan *Knapsack* dibagi menjadi tiga jenis berdasarkan persoalannya, yaitu 0-1 *knapsack problem, bounded knapsack problem* dan *unbounded knapsack problem*. Pengelompokan tersebut didasarkan pada pola penyimpanan barang dengan bobot dan nilai yang bervariasi. Permasalahan ini bertambah kompleks, ketika tiap-tiap pilihan yang ada masing-masing memiliki lebih dari satu dimensi batasan yang lebih dikenal *Multiple Constraints Knapsack Problem*. Pada bidang bisnis ekonomi, tiap-tiap objek memiliki lebih dari satu dimensi batasan sebagai bahan pertimbangan.

Penelitian sebelumnya mengenai *knapsack*, dilakukan oleh Hadi [2] membahas tentang *bounded knapsack problem* diselesaikan dengan menggunakan *hybrid genetic algorithm*, dalam penelitian tersebut disimpulkan bahwa penyelesaian *bounded knapsack problem* memiliki hasil yang lebih baik apabila diselesaikan dengan metode *hybrid genetic algorithm* daripada algoritma genetik biasa. Kemudian Hayyu [3] membahas tentang *bounded knapsack problem* diselesaikan dengan menggunakan Algoritma *Artificial Bee Colony*, dalam penelitian tersebut disimpulkan bahwa penyelesaian *bounded knapsack problem* juga memiliki hasil yang lebih baik apabila diselesaikan dengan Algoritma *Artificial Bee Colony* daripada algoritma genetik biasa, dan masih banyak algoritma yang dapat diterapkan dalam kasus optimasi salah satunya *Dragonfly Optimization Algorithm* (DOA).

DOA adalah salah satu algoritma optimasi yang dapat digunakan untuk pengambilan keputusan. Algoritma ini terinspirasi dari spesies *Capung* dalam menangkap mangsa. Algoritma ini memiliki dua fase, yaitu fase *Eksplorasi* dan fase *Eksploitasi*. *Capung* menciptakan sub kawanan dan terbang di atas wilayah yang berbeda dalam kawanan dinamis, yang merupakan tujuan utama dari fase *Eksplorasi*. Namun, dalam kelompok statis, capung terbang dalam kawanan yang lebih besar dan sepanjang satu arah, yang menguntungkan dalam fase *Eksploitasi*. Perlu diketahui bahwa terdapat lima faktor yang digunakan dalam algoritma ini, antara lain: Pertama Faktor Pemisahan yaitu faktor yang menentukan seekor capung akan berpisah dari kelompoknya, kedua Faktor Penggabungan yaitu Faktor yang menentukan seekor capung bergabung dengan kelompok baru, ketiga Faktor Penyesuaian yaitu Faktor yang menentukan seekor capung menyesuaikan arah terbang pada kelompok yang baru, keempat Faktor Sumber Makanan yaitu Faktor yang menentukan capung dalam kelompok secara bersama-sama menuju sumber makanan, dan kelima Faktor Predator yaitu Faktor yang menentukan capung dalam kelompok secara bersama-sama berpencar untuk menghindari predator.

Berdasarkan uraian diatas, penulis tertarik untuk meneliti lebih lanjut permasalahan *Multiple Constraints Bounded Knapsack* dengan menerapkan DOA. Diharapkan dari

penelitian ini DOA dapat menghasilkan solusi yang optimal dengan waktu komputasi lebih efisien.

Multiple Constraints Bounded Knapsack

Multiple constraints knapsack problem (MCKP) adalah suatu permasalahan *knapsack* yang sering disebut juga dengan *multidimensional knapsack problem* (MKP) dimana MCKP merupakan permasalahan optimasi kombinatorial NP-*hard* yang terdapat pada beragam aplikasi. Pada MCKP, tiap-tiap objek/barang memiliki batasan lebih dari satu dimensi. Batasan-batasan bisa berupa waktu, biaya dan pekerja. Tujuan dari masalah ini yaitu memperoleh solusi optimal dengan mengambil kombinasi barang sedemikian hingga semua batasan juga tidak melewati kapasitas yang tersedia [4].

Secara matematik, *multiple constraints bounded knapsack* dapat dirumuskan sebagai formulasi MCBK berikut :

Fungsi tujuan maksimum

$$Z = \sum_{j=1}^{n} p_j x_j \quad (1)$$

Kendala

$$\sum_{j=1}^{n} w_j x_j \leq C \quad (2)$$

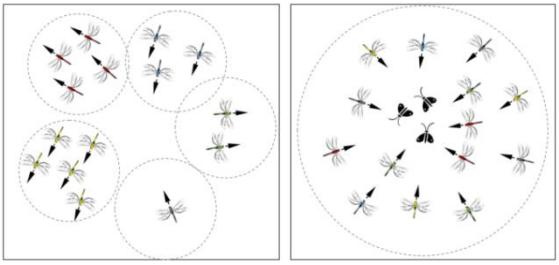
$$\sum_{j=1}^{n} v_j x_j \leq S \quad (3)$$

$$\sum_{j=1}^{n} b_j x_j \leq M \quad (4)$$

$$x_j \in \{1,2,\dots,m_j\}, \quad j = 1,2,\dots,n \quad (5)$$

Dragonfly Optimization Algorithm (DOA)

Inspirasi utama dari *Dragonfly Optimization Algorithm* (DOA) berasal dari perilaku sibuk statis dan dinamis. Kedua perilaku ini sangat mirip dengan dua fase utama optimasi menggunakan metaheuristic yaitu fase eksplorasi dan fase eksploitasi. Capung menciptakan sub kawanan dan terbang di atas wilayah yang berbeda dalam kawanan dinamis, yang merupakan tujuan utama dari fase eksplorasi. Namun, dalam kelompok statis, capung terbang dalam kawanan yang lebih besar dan sepanjang satu arah, yang menguntungkan dalam fase eksploitasi. Sebuah model konseptual dari kawanan dinamis dan statis diilustrasikan dalam Gambar 1.



Gambar 1. Model konseptual dinamis dan statis

Capung hanya menunjukkan dua jenis kawanan: statis dan dinamis seperti yang ditunjukkan pada Gambar 1. Seperti yang dilihat pada gambar diatas, capung cenderung menyelaraskan terbangnya sambil mempertahankan pemisahan dan kohesi yang tepat dalam kerumunan yang dinamis. Namun, dalamgerombolan statis, keberpihakan sangat rendah sementara kohesi tinggi untuk menyerang mangsa. Oleh karena itu, kami menetapkan capung dengan penjajaran tinggi dan bobot kohesi rendah ketika menjelajahi ruang pencarian dan penyelarasan rendah dan kohesi tinggi ketika mengeksploitasi ruang pencarian. Untuk transisi antara eksplorasi dan eksploitasi, jari-jari lingkungan meningkat sebanding dengan jumlah iterasi [5].

Simulasi perilaku capung ada limafaktor penting. Setiap perilaku partikel yang diekspresikan dirumuskan menggunakan beberapa model. Model-model ini dibahas dalam bagian berikut (Amini, *et al.*, 2018) :

1. *Separation* atau pemisahan mengacu pada mekanisme yang diikuti capung untuk menghindari tabrakan dengan capung lainnya. Perilaku ini dirumuskan seperti pada Persamaan (6)

$$S_i = \sum_{j=1}^{n} X - X_j \quad (6)$$

2. *Alignment* atau keselarasan menunjukkan kecocokan kecepatan capung menurut capung terdekat lainnya. Perilaku ini dirumuskan seperti pada Persamaan (7)

$$A_i = \frac{\sum_{j=1}^{n} V_j}{n} \quad (7)$$

3. Kohesi mengacu pada kecenderungan capung pusat massa di lingkungan itu. Perilaku ini dirumuskan seperti pada Persamaan (8)

$$C_i = \frac{\sum_{j=1}^{n} X_j}{n} - X \quad (8)$$

   Daya tarik terhadap sumber makanan dan melarikan diri dari musuh adalah dua perilaku kunci lainnya yang masing-masing capung berperilaku untuk bertahan hidup.

4. Capung yang melompat menuju sumber makanan dirumuskan menggunakan Persamaan (9)

$$F_i = X^+ - X \quad (9)$$

5. Melarikan diri dari musuh dirumuskan menggunakan Persamaan (10)

$$E_i = X^- - X \quad (10)$$

DOA menggunakan dua vektor untuk menyelesaikan masalah optimasi yaitu vektor langkah dan vektor posisi. Vektor langkah didefinisikan seperti pada Persamaan (2.14)

$$\Delta X_{t+1} = (sS_i + aA_i + cC_i + fF_i + eE_i) + w\Delta X_t \quad (11)$$

Setelah menghitung vektor langkah, vektor posisi dirumuskan seperti pada Persamaan (12)

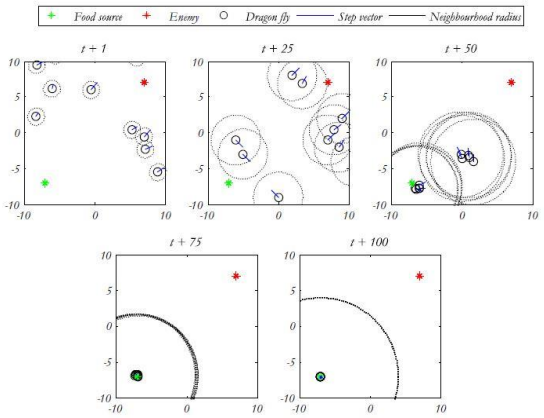
$$X_{t+1} = X_t + \Delta X_{t+1} \quad (12)$$

dengan *t* menunjukkan lokasi saat ini [1].

Vektor langkah dan vektor posisi digunakan ketika X_i mempunyai tetangga, tetapi jika X_i tidak mempunyai tetangga maka X_i berpindah secara acak, dirumus seperti pada Persamaan (13)

$$X_{t+1} = X_t + rand(-1,1)(X^+ - X_t) \quad (13)$$

Dengan pemisahan, keselarasan, kohesi, faktor makanan, dan faktor musuh (s, a, c, f, dan e), berbagai perilaku eksploratif dan eksploitatif dapat dicapai selama optimisasi. Tetangga capung sangat penting, sehingga suatu lingkungan (lingkaran dalam 2D, bola dalam ruang 3D, atau hypersphere dalam ruang *n*D) dengan radius tertentu diasumsikan di sekitar masing-masing capung buatan. Contoh perilaku berkerumun capung dengan peningkatan radius lingkungan menggunakan model matematika yang diusulkan diilustrasikan pada Gambar 2.



Gambar 2. Model perilaku berkerumun capung

Pada Gambar 2 model menggambarkan bagaimana perilaku berkurumun capung dimana terdapat beberapa tahapan berkerumun capung untuk mendapatkan mangsa, dijelaskan bahwa pertama capung akan berpencar untuk mencari atau menhindari musuh jika salah satu capung menemukan mangsa maka capung yang lain juga akan mengikuti mangsa yang telah ditemukan tersebut dan berkerumun pada mangsa untuk memakan mangsa tersebut. Jika banyak capung yang berkerumun pada satu titik yang sama maka radius linkungannya juga akan meningkat.

2. Metodologi

Data yang akan digunakan dalam penelitian ini adalah data primer berupa data penjulan dari rumah produksi kerajinan Bambu Hitam di Desa Pujerbaru, Kecamatan Maesan, Kabupaten Bondowoso. Kerajinan Bambu Hitam memproduksi berbagai macam jenis kerajinan yang berasal dari bambu hitam yang kemudian dibentuk dan diukir sesuai yang di inginkan. Data yang diambil berupa nama barang/benda, volume barang/benda,

berat barang/benda, jumlah barang/benda, biaya produksi, harga jual dan keuntungan.

Menerapkan *Dragonfly Optimization Algorithm* (DOA) pada data yang telah diidentifikasi dengan langkah-langkah berikut :

1) Input parameter dan data
2) Inisialisasi populasi awal
   a) Kandidat solusi awal

$$X = \begin{bmatrix} X_{1,1} & X_{1,2} & \cdots & X_{1,d} \\ X_{2,1} & X_{2,2} & \cdots & X_{2,d} \\ \vdots & \vdots & & \vdots \\ X_{n,1} & X_{n,2} & \cdots & X_{n,d} \end{bmatrix}$$

   Nilai X dipilih dari nilai [0,1]

$$Y = \begin{bmatrix} Y_{1,1} & Y_{1,2} & \cdots & Y_{1,d} \\ Y_{2,1} & Y_{2,2} & \cdots & Y_{2,d} \\ \vdots & \vdots & & \vdots \\ Y_{n,1} & Y_{n,2} & \cdots & Y_{n,d} \end{bmatrix}$$

   dengan $Y_{i,j} = X_{i,j} \times m_j$ kemudian dibulatkan ke pembulatan terdekat

   b) Pengecekkan kendala
   Setiap solusi harus memenuhi Persamaan (2)-(4)
   Jika tidak memenuhi , maka perlu dipinalty dengan menggunakan rumus

$$x_{i,j} = x_{i,j} - \frac{1}{m_j} \qquad (17)$$

   c) Evaluasi fungsi
   Hitung total profit dari setiap kandidat solusi. Total profit dihitung menggunakan Persamaan (18)

$$Z = \sum_{j=1}^{n} p_j y_j \qquad (18)$$

3) Inisialisasi langkah Δx
   Karena capung diawali dari posisi diam maka :

$$\Delta X = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

4) Memperbarui sumber makanan dan musuh
   X^+ untuk sumber makanan diambil dari posisi terbaik
   X^- untuk musuh diambil dari posisi terburuk
5) Memperbarui *w*, *s*, *a*, *c*, *f*, dan *e*
6) Menghitung *S*, *A*, *C*, *F*, dan *E*
   Untuk menghitung *S*, *A*, *C*, *F*, dan *E* didasarkan pada Persamaan (6)-(10).
7) Perbarui Kecepatan dan posisi
   Untuk memperbarui kecepatan dan posisi didasarkan pada Persamaan (11)-(12)

8) Memastikan X_i berada pada ruang pencarian [0,1]
   a) Jika $X_i > 0$ maka menggunakan rumus pada persamaan (19)

$$X_i = \frac{X_i - \min(X_i)}{\max(X_i) - \min(X_i)} \quad (19)$$

   b) Jika $X_i < 0$ maka menggunakan rumus pada persamaan (20)

$$X_i = \frac{X_i}{\max(X_i)} \quad (20)$$

3. Hasil dan Pembahasan

Penelitian ini menggunakan implementasi dari *Dragonfly Optimization Algorithm* (DOA) untuk menyelesaikan permasalahan dari *multiple constraints bounded knapsack* dengan menggunakan *software* MATLAB 2015. Program ini digunakan untuk mempermudah perhitungan yang dilakukan agar jika data yang diuji merupakan data dengan jumlah yang besar, maka tidak perlu dilakukan perhitungan manual. Program yang dibuat ini juga berguna untuk melihat hasil yang didapat dari DOA dalam menyelesaikan permasalahan *multiple constraints bounded knapsack*. Hasil dari percobaan akhir DOA akan dibandingkan dengan hasil metode *Simplex* dengan menggunakan presentase deviasi menggunakan Persamaan (21).

$$\text{Presentase deviasi} = \frac{simplex - z_i}{simplex} \times 100\% \quad (21)$$

Adapun hasil dari pengujian parameternya adalah sebagai berikut:

a. Uji pengaruh parameter populasi

Uji parameter populasi (*Npop*) dilakukan dengan menggunakan empat nilai, yaitu 25, 50, 100 dan 200. Populasi merepresentasikan banyaknya kandidat solusi (capung). Sementara nilai parameter yang digunakan untuk pengujian populasi ini adalah *Maxgen* = 500; *wmax* = 0,9; *wmin* = 0,2; *s* = 0,1; *a* = 0,1; *c* = 0,1; *e* = 0,01; *f* = 0,1 dan *R* = 2. Setiap kombinasi nilai parameter dilakukan 10 kali *running* program untuk kemudian dihitung rata-rata dari profit, rata-rata iterasi konvergen, dan rata-rata waktu komputasi.

Tabel 1. Hasil uji parameter populasi

Pop	Profit Rata-rata	Rata-rata Deviasi (%)	Rata-rata Iterasi Konvergen	Rata-rata Waktu Komputasi
25	11760650	0,0611	326,4	31,62333
50	11834850	0,0552	210,7	46,58957
100	11911900	0,0491	276,6	106,43653
200	**11972400**	**0,0442**	**272,7**	**340,98917**

b. Uji pengaruh parameter maksimal iterasi

Uji parameter maksimal iterasi (*maxgen*) dilakukan dengan menggunakan empat nilai, yaitu 100, 200, 500 dan 1000. Nilai parameter yang digunakan untuk pengujian populasi ini adalah *Npop* = 100; *wmax* = 0,9; *wmin* = 0,2; *s* = 0,1; *a* = 0,1; *c* = 0,1; *e* = 0,01; *f* =

0,1 dan R = 2. Setiap kombinasi nilai parameter dilakukan 10 kali *running* program sama halnya dengan pengujian terhadap parameter populasi. Langkah selanjutnya yaitu menghitung rata-rata profit, rata-rata iterasi konvergen, dan rata-rata waktu komputasi.

Tabel 2. Hasil uji parameter maximal iterasi

Maxgen	Profit Rata-rata	Rata-rata Deviasi (%)	Rata-rata Iterasi Konvergen	Rata-rata Waktu Komputasi
100	11629500	0,0716	61,4	21,3637
200	11791850	0,0586	137,9	42,37385
500	11815600	0,0568	342,2	105,303
1000	**11899050**	**0,0501**	**608,8**	**211,88449**

c. Uji pengaruh *wmax*

Pengujian parameter yang digunakan bernilai 0,3; 0,5; 0,7 dan 0,9. Nilai parameter yang digunakan untuk pengujian populasi ini adalah *Npop* = 100; *Maxgen* = 500; *wmin* = 0,2; s = 0,1; a = 0,1; c = 0,1; e = 0,01; f = 0,1 dan R = 2. Setiap nilai parameter dilakukan *running* sebanyak 10 kali. Langkah selanjutnya yaitu menghitung rata-rata profit, rata-rata iterasi konvergen, dan rata-rata waktu komputasi.

Tabel 3. Hasil uji parameter *wmax*

Wmax	Profit Rata-rata	Rata-rata Deviasi (%)	Rata-rata Iterasi Konvergen	Rata-rata Waktu Komputasi
0,3	11871050	0,0523	290,4	107,00715
0,5	**11985850**	**0,0432**	**359,6**	**105,79297**
0,7	11917050	0,0487	247,4	105,82807
0,9	11917950	0,0486	316,4	105,40362

d. Uji pengaruh *wmin*

Pengujian parameter yang digunakan bernilai 0,1; 0,15; 0,2 dan 0,25. Nilai parameter yang digunakan untuk pengujian populasi ini adalah *Npop* = 100; *Maxgen* = 500; *wmax* = 0,9; s = 0,1; a = 0,1; c = 0,1; e = 0,01; f = 0,1 dan R = 2. Setiap nilai parameter dilakukan *running* sebanyak 10 kali. Langkah selanjutnya yaitu menghitung rata-rata profit, rata-rata iterasi konvergen, dan rata-rata waktu komputasi.

Tabel 4. Hasil uji parameter *wmin*

Wmin	Profit Rata-rata	Rata-rata Deviasi (%)	Rata-rata Iterasi Konvergen	Rata-rata Waktu Komputasi
0,1	11834550	0,0552	295,6	109,63559
0,15	**11883000**	**0,0514**	**307,1**	**107,18781**
0,2	11862850	0,0530	339,4	108,24263
0,25	11874150	0,0521	291,9	107,39337

e. Uji pengaruh *s*

Pengujian parameter yang digunakan bernilai 0,01; 0,1; 0,2; 0,5 dan 1. Nilai parameter yang digunakan untuk pengujian populasi ini adalah *Npop* = 100; *Maxgen* = 500; *wmax* = 0,9; *wmin* = 0,2; *a* = 0,1; *c* = 0,1; *e* = 0,01; *f* = 0,1 dan *R* = 2. Setiap nilai parameter dilakukan *running* sebanyak 10 kali. Langkah selanjutnya yaitu menghitung rata-rata profit, rata-rata iterasi konvergen, dan rata-rata waktu komputasi.

Tabel 5. Hasil uji parameter berat pemisahan

S	Profit Rata-rata	Rata-rata Deviasi (%)	Rata-rata Iterasi Konvergen	Rata-rata Waktu Komputasi
0,01	11834600	0,0552	321,4	111,28149
0,1	11865950	0,0527	269,9	107,73187
0,2	11872300	0,0522	319,4	107,87031
0,5	**11957000**	**0,0455**	**289,4**	**107,37205**
1	11850000	0,0540	338,9	107,84718

f. Uji pengaruh *a*

Pengujian parameter yang digunakan bernilai 0,01; 0,1; 0,2; 0,5 dan 1. Nilai parameter yang digunakan untuk pengujian populasi ini adalah *Npop* = 100; *Maxgen* = 500; *wmax* = 0,9; *wmin* = 0,2; *s* = 0,1; *c* = 0,1; *e* = 0,01; *f* = 0,1 dan *R* = 2. Setiap nilai parameter dilakukan *running* sebanyak 10 kali. Langkah selanjutnya yaitu menghitung rata-rata profit, rata-rata iterasi konvergen, dan rata-rata waktu komputasi.

Tabel 6. Hasil uji parameter berat keselarasan

a	Profit Rata-rata	Rata-rata Deviasi (%)	Rata-rata Iterasi Konvergen	Rata-rata Waktu Komputasi
0,01	**11935000**	**0,0472**	**240**	**107,87077**
0,1	11897450	0,0502	220,1	107,76962
0,2	11882850	0,0514	301	107,49048
0,5	11872600	0,0522	274,1	107,60776
1	11860800	0,0531	326,9	108,49402

g. Uji pengaruh *c*

Pengujian parameter yang digunakan bernilai 0,01; 0,1; 0,2; 0,5 dan 1. Nilai parameter yang digunakan untuk pengujian populasi ini adalah *Npop* = 100; *Maxgen* = 500; *wmax* = 0,9; *wmin* = 0,2; *s* = 0,1; *a* = 0,1; *e* = 0,01; *f* = 0,1 dan *R* = 2. Setiap nilai parameter dilakukan *running* sebanyak 10 kali. Langkah selanjutnya yaitu menghitung rata-rata profit, rata-rata iterasi konvergen, dan rata-rata waktu komputasi.

Tabel 7. Hasil uji parameter berat kohesi

C	Profit Rata-rata	Rata-rata Deviasi (%)	Rata-rata Iterasi Konvergen	Rata-rata Waktu Komputasi
0,01	11867000	0,05265	236,8	108,10581
0,1	11868050	0,05256	255,6	107,81732
0,2	**11921150**	**0,0483**	**322**	**108,40331**
0,5	11867050	0,0526	271	108,01463
1	11888800	0,0509	297,9	108,17388

h. Uji pengaruh *f*

Pengujian parameter yang digunakan bernilai 0,01; 0,1; 0,2; 0,5 dan 1. Nilai parameter yang digunakan untuk pengujian populasi ini adalah *Npop* = 100; *Maxgen* = 500; *wmax* = 0,9; *wmin* = 0,2; *s* = 0,1; *a* = 0,1; *c* = 0,1; *e* = 0,01 dan *R* = 2. Setiap nilai parameter dilakukan *running* sebanyak 10 kali. Langkah selanjutnya yaitu menghitung rata-rata profit, rata-rata iterasi konvergen, dan rata-rata waktu komputasi.

Tabel 8. Hasil uji parameter faktor makanan

f	Profit Rata-rata	Rata-rata Deviasi (%)	Rata-rata Iterasi Konvergen	Rata-rata Waktu Komputasi
0,01	11868100	0,0526	308,5	107,37811
0,1	11885600	0,0512	314	107,61362
0,2	11887050	0,0510	320,5	107,956
0,5	**11896450**	**0,0503**	**277,2**	**111,93583**
1	11880850	0,0515	323,5	114,18152

i. Uji pengaruh *e*

Pengujian parameter yang digunakan bernilai 0,01; 0,1; 0,2; 0,5 dan 1. Nilai parameter yang digunakan untuk pengujian populasi ini adalah *Npop* = 100; *Maxgen* = 500; *wmax* = 0,9; *wmin* = 0,2; *s* = 0,1; *a* = 0,1; *c* = 0,1; *f* = 0,1 dan *R* = 2. Setiap nilai parameter dilakukan *running* sebanyak 10 kali. Langkah selanjutnya yaitu menghitung rata-rata profit, rata-rata iterasi konvergen, dan rata-rata waktu komputasi.

Tabel 9. Hasil uji parameter faktor musuh

e	Profit Rata-rata	Rata-rata Deviasi (%)	Rata-rata Iterasi Konvergen	Rata-rata Waktu Komputasi
0,01	11908050	0,0494	307,7	112,22814
0,1	**11917450**	**0,0486**	**300,3**	**106,74103**
0,2	11868800	0,0525	290,4	107,88859
0,5	11841100	0,0547	291,2	108,27194
1	11724800	0,0640	336,5	108,89504

j. Uji pengaruh *Range(R)*

Pengujian parameter yang digunakan bernilai 0,5; 1,25; 2 dan 2,75. Nilai parameter yang digunakan untuk pengujian populasi ini adalah *Npop* = 100; *Maxgen* = 500; *wmax* = 0,9; *wmin* = 0,2; *s* = 0,1; *a* = 0,1; *c* = 0,1; *e* =0,01 dan *f* = 0,1. Setiap nilai parameter dilakukan *running* sebanyak 10 kali. Langkah selanjutnya yaitu menghitung rata-rata profit, rata-rata iterasi konvergen, dan rata-rata waktu komputasi.

Tabel 10. Hasil uji parameter range

R	Profit Rata-rata	Rata-rata Deviasi (%)	Rata-rata Iterasi Konvergen	Rata-rata Waktu Komputasi
0,5	11831100	0,0555	317,1	60,1088
1,25	**12073350**	**0,0362**	**331,6**	**71,76167**
2	11898300	0,0501	332,8	112,32012
2,75	11867900	0,0526	267,1	116,15045

k. Simulasi akhir

Pada simulasi akhir, digunakan nilai parameter terbaik untuk menguji DOA dalam permasalahan *multiple constraints bounded knapsack*. Simulasi data dilakukan dengan 10 kali *running* program. Hasil yang diperoleh dari program dapat dilihat pada Tabel 11.

Tabel 11. Hasil akhir simulasi data dengan presentase deviasi

No	Biaya Produksi	Profit	iterasi konvergen	Waktu Komputasi (Detik)	Deviasi (%)
1	19815000	12255000	745	378,5599	0,0217
2	19872000	12253000	433	381,8868	0,0218
3	19800500	12234500	247	368,2501	0,0233
4	19752500	12337500	787	375,1007	0,0151
5	**19951500**	**12388500**	**786**	**370,8726**	**0,0110**
6	19816000	12254000	690	384,827	0,0218
7	19855000	12165000	449	383,6232	0,0289
8	19956000	12264000	841	371,1549	0,0210
9	19935000	12320000	744	371,2244	0,0165
10	19963500	12251500	875	369,0311	0,0220
Rata-rata		12272300	659,7	375,45307	0,0203

Berdasarkan hasil uji parameter populasi dan maksimal iterasi pada Tabel 1 dan 2, diketahui bahwa semakin besar nilai populasi dan semakin besar nilai maksimal iterasi, maka rata-rata profit semakin besar dan rata-rata presentasi deviasi semakin kecil. Hal ini dikarenakan kandidat solusi semakin banyak. Nilai terbaik populasi yang diujikan adalah sebesar 200 dan nilai terbaik maksimal iterasi yang diujikan adalah sebesar 1000.

Dari hasil uji parameter *wmax* dan *wmin* pada Tabel 3 dan 4, untuk nilai terbaik dari *wmax* dan *wmin* masing-masing adalah 0,5 dan 0,15, sehingga diketahui .bahwa nilai terbaiknya berada di tengah yang artinya nilainya tidak terlalu besar atau tidak terlalu kecil.

Berdasarkan hasil uji parameter s, a dan c pada Tabel 5, 6 dan 7, untuk parameter s didapat solusi terbaiknya yaitu 0,5, sehingga diketahui bahwa nilai terbaiknya berada di tengah yang artinya nilainya tidak terlalu besar atau tidak terlalu kecil. Untuk parameter a didapatkan solusi terbaiknya 0,01, sehingga diketahui bahwa nilai terbaiknya berada pada nilai terkecil tetapi tidak terlalu kecil. Untuk parameter c didapatkan solusi terbaiknya ialah 0,2, sehingga diketahui bahwa nilai terbaiknya berada di tengah yang artinya nilainya tidak terlalu besar atau tidak terlalu kecil.

Berdasarkan hasil uji parameter f, e dan Rpada Tabel 8, 9 dan 10, untuk masing-masing nilai terbaik dari parameter f, e dan R adalah 0,5, 0,1 dan 1,25, sehingga diketahui bahwa nilai terbaiknya berada di tengah yang artinya nilainya tidak terlalu besar atau tidak terlalu kecil.

Berdasarkan keseluruhan hasil uji parameter, dapat diketahui bahwa nilai terbaik dari parameter-parameternya tidak terlalu besar dan tidak terlalu kecil. Jika nilai terbaiknya terlalu besar maka posisi capung akan teracak sehingga tidak jelas posisi dari capung dan jika terlalu kecil nilai terbaiknya maka perubahannya tidak terlihat.

Berdasarkan hasil simulasi akhir data dengan persentase deviasi yang menggunakan *Pop* = 200;*Maxgen* = 1000; *wmax* = 0,5; *wmin* = 0,15; *s* = 0,5; *a* = 0,01; *c* = 0,2; *e* = 0,1; *f* = 0,5 dan *R* = 1,25 yang diujikan dengan jumlah *running* program sebanyak sepuluh, DOA belum mampu mencapai nilai optimum *Simplex*, namun presentase deviasinya sangat kecil (lihat Tabel 11), artinya hasil DOA mendekati optimal. Profit metode *Simplex* pada data kerajinan adalah sebesar Rp 12.526.500,-. Profit terkecil yang mampu dihasilkan oleh DOA sebesar Rp 12.030.500,- dengan presentase deviasi sebesar 0,0396%, sedangkan profit terbesar yang didapat yaitu sebesar Rp 12.141.500,- dengan presentase deviasi sebesar 0,0307%. Adapun rata-rata profit dari hasil simulasi akhir data yaitu sebesar Rp 12.095.750,-, rata-rata persentase deviasi sebesar 0,0344%, dan rata-rata iterasi konvergen yaitu 592.

Berdasarkan uraian di atas, dapat dikatakan bahwa DOA kurang efektif untuk menyelesaikan permasalahan *multiple constraints bounded knapsack*, karena dari banyak percobaan tidak ada solusi yang mendekati *Simplex*.Solusi mendekati optimal yang didapatkan dari data kerajianan memiliki keuntungan sebesar Rp 12.141.500,- dengan total berat 381,75 kg, total volume 8.786.031 cm^3, biaya produksi sebesar Rp 19.948.500,- dan waktu komputasi 461,1954 detik. Barang yang diangkut diantaranya dapat dilihat pada Tabel 12.

Tabel 12. Data kerajinan yang dapat diangkut

No	Nama Barang	Jumlah Barang (Satuan)	Total Berat (Kg)	Total Volume (Cm^3)
1	Meja Biasa	6	48	766536
2	Meja Sudut	4	40	1481040
3	Nampan 1	17	8,5	27744
4	Nampan 2	16	16	70400
5	Nampan 3	15	15	99000
6	Nampan 4	13	13	136500
7	Peti Harta Besar	20	40	369600
8	Peti Harta Kecil	20	20	84000
9	Vas Bunga Duduk	17	8,5	969408
10	Vas Bunga Lantai	18	13,5	1742400
11	Lampu Gantung	5	17,5	1125000
13	Lampu Dinding 1	7	17,5	361998
14	Lampu Dinding 2	8	20	730080
15	Hiasan Dinding	18	9	549900
16	Pigura 20R	10	10	30000
17	Pigura 85x65	12	18	99450
18	Kere Gambar 1	13	19,5	31200
19	Kere Gambar 2	9	15,75	26775
20	Kere Gambar 3	7	14	21000
21	Kere Gambar 4	8	18	64000
	Total	243	381,75	8786031

4. Kesimpulan

a. Solusi terbaik yang dihasilkan oleh *Dragonfly Optimization Algorithm* dalam penyelesaian masalah *multiple constraint bounded knapsack* adalah sebesar Rp 12.141.500,-. Solusi terbaik tersebut diperoleh dari profit paling maksimal hasil akhir simulasi data dengan nilai terbaik dari setiap parameter dan iterasi konvergen 613. Parameter-parameter tersebut memiliki pengaruh yang sama yaitu untuk mendapatkan hasil yang optimal, dimana semakin besar nilai dari parameter tersebut maka hasil yang didapatkan juga semakin mendekati nilai optimal.

b. Hasil dari *Dragonfly Optimization Algorithm* yang memiliki keuntungan Rp 12.141.500,- dibandingkan dengan hasil *simplex* yang memiliki keuntungan Rp 12.526.500,-, yang artinya *Dragonfly Optimization Algorithm* kurang efektif untuk menyelesaikan permasalahan *multiple constraint bounded knapsack.Dragonfly Optimization Algorithm* mendekati optimal, dilihat dari deviasi yang cukup kecil.

Daftar Pustaka

[1] Amini, Z., Maeen, M., dan Jahangir, M.R. (2018). Providing A Load Balancing Method Based on Dragonfly Optimization Algorithm for Resource Allocation in Cloud Computing. *International Journal of Networked and Distributed Computing*. **6** (1), 75-85.

[2] Hadi, I.S. (2015). Penerapan Algoritma Genetika *Hybrid* pada Permasalahan *Bounded Knapsack*. *Skripsi*. Jember: Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

[3] Hayyu, A. N. (2016). Penerapan Algoritma *Artificial Bee Colony* pada Permasalahan *Multiple Constraints Bounded Knapsack*. *Skripsi*. Jember: Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

[4] Lamine, A., Khemakhem, M., dan Chabchoud, H. (2012). Knapsack Problem Involving Dimensions, Demands and Multiple Choice Constraints. *International Journal of Advanced Science and Technology*. **46** (4), 55-61.

[5] Mirjalili, S. (2016). Dragonfly Algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Comput & Appl*, **27**, 1053-1073.