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

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

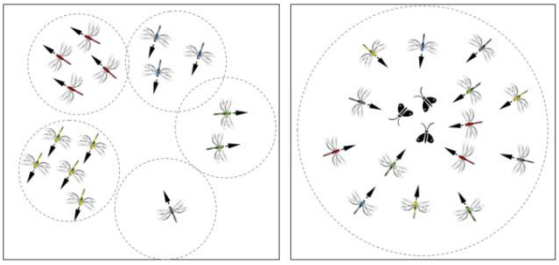
$$\sum_{j=1}^{n} v_j x_j \leq S \quad (3)$$

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

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

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)

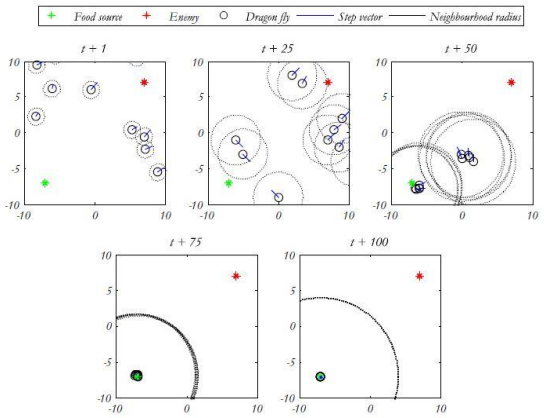
$$X_{t+1} = X_t + \Delta X_{t+1} \quad (12)$$

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.