Majalah Ilmiah Matematika dan Statistika Volume 18 Nomor 1, 2018, 13 – 22
https://jurnal.unej.ac.id/index.php/MIMS/index ISSN 1411-6669

Algoritma *Elephant Herding Optimization*: Permasalahan *Multiple Constraints Knapsack* 0-1

(*Elephant Herding Optimization Algorithm: Multiple Constraints Knapsack* 0-1 *Problems*)

Yulia Dewi Regita, Kiswara Agung Santoso, Ahmad Kamsyakawuni

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

Abstract. Optimization problems are often found in everyday life, such as when determining goods to be a limited storage media. This causes the need for the selection of goods in order to obtain profits with the requirements met. This problem in mathematics is usually called a knapsack. Knapsack problem itself has several variations, in this study knapsack type used is multiple constraints knapsack 0-1 which is solved using the Elephant Herding Optimization (EHO) algorithm. The aim of this study is to obtain an optimal solution and study the effectiveness of the algorithm comparing it to the Simplex method in Microsoft Excel. This study uses two data, consisting of primary and secondary data. Based on the results of parameter testing, the proven parameters are nClan, nCi,α,β and MaxGen have a significant effect. The final simulation results have also shown a comparison of the EHO algorithm with the Simplex method having a very small percentage deviation. This shows that the EHO algorithm is effective for completing optimization multiple constraints knapsack 0-1.

Keywords: EHO *Algorithm, Multiple Constraints Knapsack* 0-1 *Problem.*
MSC 2010: 9008

1. Pendahuluan

Masalah optimasi merupakan hal yang sering dijumpai dalam pekerjaan sehari-hari, seperti penentuan pemilihan barang pada permasalahan *knapsack*. Permasalahan *knapsack* merupakan permasalahan yang berhubungan dengan penyimpanan objek ke dalam media penyimpanan yang bersifat terbatas. *Knapsack* merupakan suatu kantong atau tempat yang digunakan untuk memuat sesuatu objek. Kantong atau tempat tersebut hanya dapat menyimpan beberapa objek saja dengan ketentuan total ukuran objek tersebut lebih kecil atau sama dengan ukuran kapasitasnya.

Knapsack terdiri dari beberapa persoalan antara lain *Knapsack* 0-1 (*Binary Knapsack*), *Knapsack* terbatas (*Bounded Knapsack*) dan *Knapsack* tak berbatas (*Unbounded*

Knapsack) [3]. Permasalahan *knapsack* tersebut terdiri dari beberapa variasi yaitu *Multi Objective Knapsack Problem, Multi Dimensional* atau *Multiple Constraints Knapsack Problem, Multi Knapsack Problem* dan *Quadratic Knapsack Problem* [2]. Penelitian ini akan membahas permasalahan *Multi Dimensional* atau *Multiple Constraints Knapsack* 0-1. Permasalahan *Multi Dimensional* atau *Multiple Constraints Knapsack* 0-1 merupakan salah satu variasi dari permasalahan *knapsack,* dimana terdapat beberapa barang yang harus dipilih dengan kendala lebih dari 1. Tujuan dari *multiple constraints knapsack* 0-1 adalah untuk memaksimalkan keuntungan yang diperoleh. Permasalahan *multiple constraints knapsack* 0-1 pernah diteliti Hilviah [1] dengan menerapkan algoritma *Dynamic Programming* dan algoritma *Backtracking*. Pada penelitiannya disimpulkan bahwa algoritma *Dynamic Programming* lebih efektif dan efisien dibandingkan dengan algoritma *Backtracking*. Hal ini berdasarkan proses *running time* algoritma *Dynamic Programming* membutuhkan waktu yang lebih singkat dibandingkan dengan algoritma *Backtracking*. Mengacu pada penelitian tersebut penulis tertarik untuk melakukan penelitian lebih lanjut tentang permasalahan *multiple constraints knapsack* 0-1 dengan menggunakan algoritma metaheuristik lain.

Algoritma *Elephant Herding Optimization* (EHO) merupakan salah satu algoritma metaheuristik yang dikenalkan oleh Wang dkk. [6] diusulkan untuk menyelesaikan tugas optimasi. Untuk menunjukkan keefektifannya, EHO diterapkan untuk menyelesaikan optimasi fungsi. Hasil dari EHO dibandingkan dengan BBO (*Biogeography-based Optimization*), DE (*Differential Evolution*) dan GA (*Genetic Algorithm*). Berdasarkan perbandingan menunjukkan bahwa EHO dapat menemukan solusi yang jauh lebih baik dibandingkan ketiga algoritma metaheuristik tersebut, EHO memiliki konvergensi lebih tinggi.

Berdasarkan uraian di atas, penulis tertarik untuk menerapkan algoritma EHO pada penyelesaian permasalahan *multiple constraints knapsack* 0-1. Untuk mengetahui keefektifan EHO maka perlu adanya pembanding. Metode simpleks merupakan metode dasar yang digunakan untuk menyelesaikan permasalahan program linier termasuk *multiple constraints knapsack* 0-1. Dengan demikian, pada penelitian ini hasil dari algoritma EHO akan dibandingkan dengan hasil metode simpleks menggunakan *Solver Add-In* pada *Microsoft Excel*.

Mutiple Constraints Knapsack 0-1

Pada permasalahan *Multiple Constraints Knapsack* 0-1, tiap-tiap item pilihan memiliki batasan lebih dari satu dimensi. Batasan (kendala) tersebut contohnya seperti berat, biaya, waktu, pekerja, dan lainnya [4]. Tujuannya yaitu untuk memperoleh solusi optimum dengan memilih kombinasi barang dimana semua batasan tidak melebihi kapasitas yang tersedia. *Multiple Constraints Knapsack 0-1* dapat dirumuskan sebagai berikut [2].

Fungsi tujuan:

$$Z = \sum_{i=1}^{n} p_i x_i \qquad (1)$$

Kendala:

$$\sum_{i=1}^{n} w_i x_i \leq C \quad (2)$$

$$\sum_{i=1}^{n} b_i x_i \leq M \quad (3)$$

$$\sum_{i=1}^{n} v_i x_i \leq V \quad (4)$$

Algoritma *Elephant Herding Optimization* (EHO)

Metode EHO terinspirasi oleh perilaku menggiring kelompok gajah atau perpindahan kelompok gajah dari tempat satu ke tempat yang lain. Di alam, gajah-gajah dengan kelompok yang berbeda hidup bersama di bawah kepemimpinan *matriarch*, dan gajah jantan akan meninggalkan kelompok keluarga ketika tumbuh dewasa. Kedua perilaku ini dapat dimodelkan menjadi dua operator yaitu *clan updating operator* dan *separating operator* [6].

Tahap-tahap dalam EHO meliputi *clan updating operator* dan *separating operator*:

a. *Clan Updating Operator*

Semua gajah hidup bersama di bawah kepemimpinan seorang matriarch di setiap kelompok. Oleh karena itu, untuk setiap gajah dalam kelompok, posisi berikutnya dipengaruhi oleh matriarch ci. Untuk gajah anggota di kelompok ci diperbarui berdasarkan Persamaan berikut.

Pemimpin kelompok (*Matriarch*) memperbarui posisinya dengan Persamaan berikut

$$x_{ci,j}(t+1) = \beta\, x_{center,ci} + x_{ci,j}(t) \quad (6)$$

b. *Separating Operator*

Gajah jantan akan meninggalkan kelompok keluarga dan hidup sendiri ketika dewasa. Proses pemisahan ini dapat dimodelkan ke dalam separating operator seperti persamaan berikut.

$$x_{worst,ci} = x_{\min} + (x_{\max} - x_{\min} + 1)\, r \quad (7)$$

Berdasarkan deskripsi *clan updating operator dan separating operator*, metode EHO dikembangkan dan diringkas seperti yang ditunjukkan pada *pseudocode* berikut [*6*].

Langkah 1 : Initialization. Atur penghitung generasi $t = 1$; menginisialisasi populasi; *MaxGen* generasi maksimum.

Langkah 2 : While $t <$ *MaxGen* **do**

Urutkan semua gajah sesuai dengan kemampuan mereka.

Menerapkan *clan updating operator* menggunakan Persamaan 5 dan 6.

Menerapkan *separating operator* menggunakan Persamaan 7.
Evaluasi populasi dengan posisi yang baru diperbarui.
$t = t + 1$.

2. Metodologi

Sebagai bahan simulasi, terdapat dua data yang digunakan pada penelitian ini. Data pertama adalah data primer dari Toko Citra Tani terletak di Jl. PB Sudirman No.74, Kecamatan Panti, Kabupaten Jember. Data kedua adalah data sekunder dari [5]. Data yang digunakan berisi nama barang, jumlah ketersediaan barang, satuan barang, berat barang, volume barang, harga jual barang dan harga beli barang. Adapun media penyimpanan (*knapsack*) yang digunakan untuk data 1 memiliki kapasitas berat 5.000 kg, volume maksimal *knapsack* 9.000.000 dan modal Rp. 20.000.000,-. Sedangkan media penyimpanan (*knapsack*) yang digunakan untuk data 2 memiliki kapasitas berat *4.000* kg, volume maksimal *knapsack* 8.540.000 dan modal Rp. 10.000.000,-.

Langkah-langkah Penerapan Algoritma EHO pada permasalahan *Multiple Constraints Knapsack 0-1* yang akan dilakukan pada penelitian ini adalah sebagai berikut.

1. Inisialisasi jenis barang (Brg), jumlah barang (N), jumlah kelompok ($nClan$), jumlah individu (nCi), *Max Gen* (iterasi), parameter (α,β), volume maksimal (V), berat maksimal (C), modal (M), harga beli (b), harga jual (j).

2. Bangkitkan bilangan acak. Posisi partikel ($\boldsymbol{X}$) dibangkitkan secara acak pada selang nilai [0,1]. Posisi partikel ini merepresentasikan persentase jumlah barang yang akan dimasukkan ke dalam *knapsack*.

$$X_i = random[0,1], i = 1,...., n$$
$$Y_i = round[X_i], i = 1,...., n$$

3. Pengecekan Solusi. Setiap solusi individu akan dicek apakah memenuhi kendala *multiple constraints knapsack* atau tidak. Apabila terdapat barang yang jumlahnya melebihi ketersediaanya, maka akan dilakukan penalti. Penalti dilakukan dengan cara memilih secara acak barang yang akan ditinggalkan.

$$\sum_{i=1}^{n} w_i Y_i \le C \tag{8}$$

$$\sum_{i=1}^{n} b_i Y_i \le M \tag{9}$$

$$\sum_{i=1}^{n} v_i Y_i \le V \tag{10}$$

4. Menghitung *Fitness*. *Fitness* diperoleh dari total keuntungan barang yang dipilih.

$$Z = \sum_{i=1}^{n} (j - b) Y_i$$
$$Z = \sum_{i=1}^{n} p_i Y_i \qquad (11)$$

5. Simpan *Matriarch.* Simpan posisi terbaik setiap kelompok (*Clan*) sebagai *Matriarch* (memiliki *fitness* terbaik).

6. Terapkan *Clan Updating Operator*.
   a) Perbarui posisi. Jika posisinya tidak sama dengan dengan *Matriarch* maka gunakan Persamaan (5). Jika posisinya sama dengan *Matriarch* maka gunakan Persamaan (6).
   b) Cek solusi baru. Cek solusi menggunakan Persamaan (8), (9) dan (10).
   c) Menghitung *fitness* menggunakan persamaan (11).

7. Terapkan *Separating Operator*. Nilai *fitness* terkecil setiap *Clan* diperbarui posisinya dengan Persamaan (7). Jika posisi gajah tidak berada pada ruang pencarian, artinya nilai nilai min(Xi) < 0 atau max(Xi) > 1 maka perlu dilakukan transformasi nilai menggunakan Persamaan berikut.

$$X_i = \begin{cases} \dfrac{X_i - \min(X_i)}{\max(X_i) - \min(X_i)}, & \text{jika } \min(X_i) < 0 \\ \dfrac{X_i}{\max(X_i)}, & \text{jika } \max(X_i) > 1 \end{cases}$$

8. Update *Matriarch*. Jika posisi gajah terbaik (memiliki *fitness* terbesar) lebih baik dari posisi *Matriarch* maka posisi gajah terbaik tersebut menggantikan *Matriarch*. Namun jika tidak lebik baik maka M*atriarch* dipertahankan.

9. Periksa kriteria pemberhentian. Jika iterasi telah mencapai iterasi maksimal, maka proses pencarian dihentikan. Jika belum, kembali ke langkah 5.

3. Hasil dan Pembahasan

Pada penelitian ini, terlebih dahulu dilakukan pengujian parameter guna mengetahui pengaruhnya terhadap hasil. Program yang telah dibuat dijalankan pada Laptop dengan CPU AMD E1-1500 APU with Radeon(tm) HD Graphics 1.48 GHz 2GB RAM 64bit OS. Hasil dari pengujian parameter adalah sebagai berikut.

a. Uji parameter *pop*
Pengujian parameter yang digunakan bernilai 5, 10, 15, 20 dan 25. Parameter lain yang digunakan yaitu *nCi* = 5; α= 0,5; β= 0,5; dan *MaxGen* = 1000. Setiap nilai parameter dilakukan *running* sebanyak 10 kali.

Tabel 1. Hasil uji parameter *nClan* data 1

nClan	Profit Rata-rata	Profit Terbaik	Profit Terburuk	Rata-rata Iterasi Konvergen	Rata-rata Waktu Komputasi (s)
10	2714000	2714000	2714000	136,4	104,70009
15	2714000	2714000	2714000	77,5	111,37789
20	2714000	2714000	2714000	62,1	116,09804
25	2714000	2714000	2714000	57,3	121,83335

Tabel 2. Hasil uji parameter *nClan* data 2

nClan	Profit Rata-rata	Profit Terbaik	Profit Terburuk	Rata-rata Iterasi Konvergen	Rata-rata Waktu Komputasi (s)
5	14231000	14261000	14153500	615,6	96,3749
10	14242100	14261000	14201000	629,9	104,57839
15	14245850	14278500	14211000	664,5	110,40114
20	14250300	14291000	14226000	686,3	116,77666
25	14269000	14291000	14248500	735,9	123,00336

b. Uji parameter *nCi*
Pengujian parameter yang digunakan bernilai 5, 10, 15, 20 dan 25. Parameter lain yang digunakan yaitu *nClan* = 25; α= 0,5; β= 0.5; dan *MaxGen* = 1000. Setiap nilai parameter dilakukan *running* sebanyak 10 kali.

Tabel 3. Hasil uji parameter *nCi* data 1

nCi	Profit Rata-rata	Profit Terbaik	Profit Terburuk	Rata-rata Iterasi Konvergen	Rata-rata Waktu Komputasi (s)
5	2714000	2714000	2714000	57,3	121,83335
10	2714000	2714000	2714000	44,6	128,18869
15	2714000	2714000	2714000	31,6	148,22361
20	2714000	2714000	2714000	24,6	154,66417
25	2714000	2714000	2714000	17,1	169,6979

Tabel 4. Hasil uji parameter *nCi* data 2

nCi	Profit Rata-rata	Profit Terbaik	Profit Terburuk	Rata-rata Iterasi Konvergen	Rata-rata Waktu Komputasi (s)
5	14269000	14291000	14248500	735,9	123,35474
10	14271250	14291000	14248500	639,4	141,45492
15	14274000	14291000	14261000	550,4	158,51989
20	14279750	14291000	14271000	553,6	173,90146
25	14284000	14291000	14278500	586,6	188,89821

c. Uji parameter

Pengujian parameter yang digunakan bernilai 0,1; 0,3; 0,5; 0,7; dan 0,9. Parameter lain yang digunakan yaitu *nClan* = 25; *nCi* = 25; dan *MaxGen* = 1000. Setiap nilai parameter dilakukan *running* sebanyak 10 kali.

Tabel 5. Hasil uji parameter *nCi* data 1

α β	Profit Rata-rata	Profit Terbaik	Profit Terburuk	Rata-rata Iterasi Konvergen	Rata-rata Waktu Komputasi (s)
0,10,1	2714000	2714000	2714000	44,7	162,86598
0,30,3	2714000	2714000	2714000	23,9	167,9206
0,50,5	2714000	2714000	2714000	17,1	169,6979
0,70,7	2714000	2714000	2714000	20,2	170,10366
0,90,9	2714000	2714000	2714000	31,3	172,00592

Tabel 6. Hasil uji parameter *nCi* data 2

α β	Profit Rata-rata	Profit Terbaik	Profit Terburuk	Rata-rata Iterasi Konvergen	Rata-rata Waktu Komputasi (s)
0,10,1	14260050	14291000	14216500	457,2	172,59506
0,30,3	14267250	14291000	14241000	532,2	180,45311
0,50,5	14284000	14291000	14261000	547,4	189,63165
0,70,7	14284750	14291000	14268500	547,1	190,57031
0,90,9	14289000	14291000	14278500	683	192,28992

Setelah dilakukan pengujian parameter, selanjutnya dilakukan simulasi akhir terhadap kedua data. Nilai parameter yang digunakan untuk data pertama adalah *nClan* = 25; *nCi* = 25; α= 0,9; β= 0,9; dan *MaxGen* = 100. Sedangkan nilai parameter yang digunakan untuk data kedua adalah *nClan* = 50; *nCi* = 50; α= 0,9; β= 0,9; dan *MaxGen* = 1000. Hasil dari algoritma EHO dibandingkan dengan hasil *Simplex* untuk dihitung persentase deviasinya berdasarkan rumus berikut.

$$\%Dev = \frac{Simplex - Z_i}{Simplex} \times 100\%$$

Simulasi akhir dilakukan sebanyak 10 kali *running* program. Keseluruhan hasil dari simulasi dapat dilihat pada Tabel 7 dan Tabel 8 berikut.

Tabel 7. Hasil simulasi akhir data 1

Percobaan	Profit	Iterasi Konvergen	Waktu Komputasi (s)	%Dev
1	2714000	32	20,4023	0
2	2714000	16	17,6311	0
3	2714000	9	17,0735	0
4	2714000	64	17,3085	0
5	2714000	8	17,4049	0
6	2714000	29	18,1312	0
7	2714000	43	17,3947	0
8	2714000	6	17,6921	0
9	2714000	17	17,3115	0
10	2714000	30	17,5142	0
Rata-rata	2714000	25,4	17,7864	0

Tabel 8. Hasil simulasi akhir data 2

Percobaaan	Profit	Iterasi Konvergen	Waktu Komputasi (s)	%Dev
1	14271000	849	478,0462	0,139948
2	14291000	944	451,0454	0
3	14288500	411	496,1320	0,017494
4	14278500	475	441,2392	0,087468
5	14291000	732	432,9764	0
6	14291000	128	434,4352	0
7	14291000	636	465,0857	0
8	14291000	333	664,4250	0
9	14291000	742	445,5383	0
10	14271000	113	490,6595	0,139948
Rata-rata	14285500	536,3	479,9903	0,038486

Algoritma EHO memiliki lima parameter yaitu *nClan*, *nCi*,α, βdan *MaxGen*. Berdasarkan hasil uji parameter diketahui bahwa data 1 menunjukkan hasil profit yang sama secara keseluruhan. Oleh karena itu, dibutuhkan simulasi data lain untuk

mengetahui pengaruh parameter terhadap hasil yang didapatkan. Simulasi data yang digunakan yaitu berasal dari penelitian Putri, R. J. M. (2017), berdasarkan hasil uji parameter diketahui bahwa nilai parameter mempengaruhi hasil. Semakin besar nilai parameter maka semakin optimal hasil yang didapatkan.

Berdasarkan hasil simulasi akhir data 1, ditemukan solusi terbaik untuk algoritma EHO dengan metode Simpleks. Profit yang didapatkan dari metode Simpleks pada data 1 yaitu sebesar Rp. 2.714.000,-. Pada Tabel 7 dapat dilihat hasil yang didapatkan dari algoritma EHO yaitu sama dengan hasil dari metode Simpleks. Rata-rata iterasi non-improve pada 25,4 dan nilai persentase deviasi yang didapatkan sebesar 0%.

Berdasarkan hasil simulasi akhir data 2, ditemukan solusi terbaik untuk algoritma EHO dengan metode Simpleks. Profit yang didapatkan dari metode Simpleks pada data 2 yaitu sebesar Rp. 14.291.000,-. Pada Tabel 8 dapat dilihat hasil yang didapatkan dari algoritma EHO mendekati hasil metode Simpleks. Rata-rata iterasi non-improve pada *536,3* dan persentase deviasi bernilai 0,038486%. Profit terkecil yang didapatkan dari hasil simulasi yaitu sebesar Rp. 14.271.000,-.

Berdasarkan uraian di atas, dapat disimpulkan bahwa algoritma EHO efektif untuk menyelesaikan permasalahan *multiple constraints knapsack* 0-1. Algoritma ini tepat untuk diterapkan dalam mendapatkan keuntungan yang optimal. Solusi optimal yang didapatkan dari data 1 memiliki keuntungan Rp. 2.714.000,- dengan total berat 3375 kg, volume 8.024.076 , dan harga beli Rp. 19.675.500,-. Solusi optimal yang didapatkan dari data 2 memiliki keuntungan Rp. 14.291.000,- dengan total berat 405 kg, volume 6.229.012 , dan harga beli Rp. 9.999.000,-.

4. Kesimpulan

Berdasarkan hasil dan pembahasan, diambil kesimpulan sebagai berikut.

a. Permasalahan *multiple constraints knapsack* 0-1 menggunakan algoritma *Elephant Herding Optimization* (EHO) pada kedua data menghasilkan profit terbaik sebesar Rp. 2.714.000,- (data 1) dengan nilai *nClan* = 25; *nCi* = 25, α= 0,9; β= 0,9 dan *MaxGen* = 100, serta profit sebesar Rp. 14.291.000,- (data 2) dengan nilai *nClan* = 50; *nCi* = 50, α= 0,9; β= 0,9 dan *MaxGen* = 1000. Rata-rata iterasi konvergen yang didapatkan sebesar 25,4 (data 1) dan 536,3 (data 2). Pengaruh parameter pada algoritma *Elephant Herding Optimization* (EHO) dalam menyelesaikan permasalahan *multiple constraints knapsack* 0-1 yaitu jika nilai parameternya semakin besar maka profit yang didapatkan akan semakin optimal.
b. Hasil algoritma EHO dibandingkan dengan metode Simpleks pada *Microsoft Excel* memiliki hasil rata-rata presentase deviasi yang optimal yaitu sebesar 0% (data 1) dan 0,038486% (data 2), hal ini menunjukkan bahwa algoritma EHO efektif dalam menyelesaikan permasalahan *multiple constraints knapsack 0-1*.

Daftar Pustaka

[1] Hilviah, F. (2015). Penerapan Algoritma *Dynamic Programming* dan Algoritma *Backtracking* pada Permasalahan *Multiple Constraints Knapsack* 0-1. *Skripsi*. Jember: Jurusan Matematika FMIPA Universitas Jember.

[2] Kellerer, H., Pferschy, U. dan Pisinger, D. (2004). *Knapsack Problem*. Verlag Berlin Heidelberg: Springer.

[3] Pisinger, D. (1995). *A minimal Algorithm for the Multiple-Choise Knapsack Problem*. *Europan Journal of Operational Research*, 83(2): 349-410.

[4] Puchinger, D., Raidl, G. R. dan Pferschy, U. (2007). The Multidimensional Knapsack Problem: Structur And Algorithms. *INFORMS Jurnal on Computing*. 22: 250-265.

[5] Putri, R. J. M. (2017). Penerapan Algoritma *Modified Variable Neighborhood Search* (MVNS) pada Permasalahan *Multiple Constraints Knapsack* 0-1. *Skripsi*. Jember: Jurusan Matematika FMIPA Universitas Jember.

[6] Wang, G., Deb, G. S. dan Coelho, L. D. S. (2015). Elephant Herding Optimization. *Computational and Business Intelligence (ISCBI), 2015 3rd International Symposium on*. IEEE. 1-5.