Perbandingan Efektifitas Penyelesaian Masalah pada Toko Citra Tani Jember Menggunakan Algoritma *Grey Wolf Optimization* (GWO) dan Algoritma *Particle Swarm Optimization* (PSO)

Jihan Fadila[1]

[1]Program Studi Ilmu Komputer Universitas Pertamina

Email: [1]105222022@student.universitaspertamina.ac.id

Abstrak - Penelitian ini bertujuan untuk membandingkan efektivitas algoritma Grey Wolf Optimization (GWO) dan Particle Swarm Optimization (PSO) dalam menyelesaikan permasalahan Multiple Constraints Bounded Knapsack pada kasus optimasi pemasokan barang di Toko Citra Tani Jember merujuk pada penelitian Vidiyanti., et al.[1], . Menggunakan data simulasi dengan batasan berat maksimum 5.000 kg, volume maksimum 9.000.000 cm^3, dan modal maksimum Rp 20.000.000, kedua algoritma diuji dengan berbagai parameter populasi dan iterasi. Hasil menunjukkan bahwa GWO lebih efektif dengan rata-rata standar deviasi 0,167% dibandingkan PSO dengan 4,3695%. GWO mencapai optimasi profit terbaik sebesar Rp 3.923.000 dengan pengeluaran Rp 19.998.000 dan standar deviasi 0,13%. Penelitian ini menyimpulkan bahwa GWO lebih optimal dalam menyelesaikan permasalahan Multiple Constraints Bounded Knapsack, meskipun membutuhkan kompleksitas data dan waktu komputasi yang lebih tinggi dibandingkan PSO.

Kata kunci: Grey Wolf Optimization, Particle Swarm Optimization, Multiple Constraints Bounded Knapsack, Optimasi, Pemasokan Barang

1. LANDASAN TEORI

Optimasi adalah proses menentukan nilai minimum atau maksimum dari suatu fungsi tujuan. Banyak permasalahan dalam kehidupan sehari-hari yang memerlukan optimasi, seperti penyusunan jadwal, penentuan rute kendaraan, dan manajemen inventori. Salah satu permasalahan optimasi yang menarik adalah pemilihan barang untuk dimasukkan ke dalam media penyimpanan terbatas, yang dikenal sebagai masalah *knapsack*.

Masalah *knapsack* dapat dibagi menjadi tiga jenis utama: 0-1 *knapsack*, *bounded knapsack*, dan *unbounded knapsack*. Variasi dari masalah ini termasuk *multi-objective knapsack*, *multidimensional knapsack*, *multi knapsack*, dan *quadratic knapsack*.

Penelitian ini merupakan pengembangan ulang dari studi sebelumnya yang dilakukan oleh Vidiyanti., et al.[1]. Penelitian tersebut bertujuan memaksimalkan profit dengan melakukan penentuan pemasokan barang untuk toko Citra Tani Jember, dengan mempertimbangkan batasan-batasan tertentu. Masalah yang ditangani dalam penelitian ini termasuk dalam kategori *Multiple Constraints Bounded Knapsack*, di mana *Multiple Constraints Bounded Knapsack* (MCBK) adalah salah satu variasi dari masalah *knapsack* yang melibatkan lebih dari satu batasan atau kriteria untuk setiap item yang dimasukkan ke dalam *knapsack*. Dalam konteks ini, *knapsack* diinterpretasikan sebagai kapasitas, seperti maksimal stok yang dibutuhkan, maupun maksimal modal yang dimiliki. Secara matematis, *Multiple Constraints Bounded Knapsack* dapat dirumuskan sebagai berikut:

$$z = \sum_{i=1}^{n} p_i \cdot x_i \quad \ldots(1)$$

- z adalah fungsi tujuan atau nilai yang ingin dimaksimalkan atau diminimalkan dalam permasalahan optimasi.
- ∑ (sigma) menunjukkan operasi penjumlahan dari semua nilai pi.xi
- pi adalah nilai atau profit dari item ke-iii.
- xi adalah variabel keputusan yang menunjukkan apakah item ke-i dipilih (jika xi=1) atau tidak (jika xi=0).

Kapasitas berat total :

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

- W adalah batasan berat total stok yang tersedia.

Kapasitas volume total :

$$\sum_{i=1}^{n} v_i \cdot x_i \leq V \quad \ldots(3)$$

- V adalah batasan volume total stok yang tersedia.

Batasan jumlah barang setiap jenis :

$$x_i \leq b_i, \quad \forall i = 1, 2, \ldots, n \quad \ldots(4)$$

- Menggambarkan bahwa jumlah barang jenis ke-i yang dipilih tidak boleh melebihi batasan bi.

Batasan tambahan lainnya :

$$\sum_{i=1}^{n} C_{ji} \cdot x_i \leq C_j, \quad \forall j = 1, 2, \ldots, m \quad \ldots(5)$$

- Cji adalah parameter yang menunjukkan kontribusi atau pengaruh dari barang jenis ke-iii terhadap batasan tambahan Cj.

Dalam penelitian Vidiyanti., et al.[1], algoritma yang digunakan untuk menyelesaikan masalah knapsack adalah algoritma *Grey Wolf Optimization* (GWO). Penelitian sebelumnya menunjukkan bahwa algoritma GWO efektif dalam menangani permasalahan *Constraints Bounded Knapsack* di toko Citra Tani Jember. Selain GWO, terdapat algoritma lain yang juga efisien dalam menyelesaikan permasalahan *Multiple Constraints Bounded Knapsack*, salah satunya adalah algoritma *Particle Swarm Optimization* (PSO).

Particle Swarm Optimization (PSO) merupakan algoritma optimisasi yang terinspirasi dari perilaku sosial burung atau ikan dalam kelompok. Dalam PSO, setiap solusi potensial yang disebut sebagai partikel, menyesuaikan posisi dan kecepatannya dalam ruang pencarian berdasarkan pengalaman pribadinya (personal best) dan pengalaman tetangganya (global best). Interaksi kooperatif antara partikel-partikel ini membantu menjelajahi ruang pencarian secara efisien dan konvergen ke solusi optimal untuk masalah yang diberikan. Sehingga PSO juga cocok digunakan untuk pencarian solusi optimal dalam permasalahan *Multiple Constraints Bounded Knapsack*.

Algoritma *Grey Wolf Optimization* (GWO)

GWO adalah algoritma optimasi yang terinspirasi oleh perilaku berburu serigala abu-abu. Algoritma ini menggunakan hirarki dominan sosial serigala untuk memperbarui posisi serigala dalam populasi. Serigala alfa, beta, dan delta dianggap sebagai solusi terbaik pertama, kedua, dan ketiga, dan serigala lainnya diperbarui posisinya berdasarkan posisi ketiga serigala terbaik tersebut. GWO dapat dirumuskan sebagai berikut:

$$D = \left| C \cdot X_p(t) - X(t) \right|$$
$$X(t+1) = X_p(t) - A \cdot D \quad \ldots(6)$$
$$A = 2a \cdot r_1 - a$$
$$C = 2a \cdot r_2 \quad \ldots(7)$$

- r adalah bilangan random 0-1
- a mengendalikan rentang faktor skala dalam persamaan pembaruan posisi.

Formula pembaruan posisi dalam GWO adalah sebagai berikut:

$$\vec{D}_\alpha = |\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}|$$
$$\vec{D}_\beta = |\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}|$$
$$\vec{D}_\delta = |\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}|$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot \vec{D}_\alpha$$
$$\vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot \vec{D}_\beta$$
$$\vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot \vec{D}_\delta$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \quad \ldots(8)$$

Menurut Mirjalili., et al. [2] *pseudocode* dari algoritma GWO dapat digambarkan sebagai berikut:

Inisialisasi populasi serigala abu-abu *Xi*(1,2,..., n)
Inisialisasi a, A, dan C
Hitung kesesuaian setiap perantara pencari
    Xα = Perantara Pencari terbaik
    Xβ = Perantara Pencari terbaik kedua
    Xδ = Perantara Pencari terbaik ketiga
while (t < maksimal_iterasi)
for
    setiap perantara pencari Perbarui posisi perantara pencari saat ini dengan persamaan (8)
end for
Perbarui a, A, dan C
Hitung Xα, Xβ, Xδ
end while

Algoritma *Particle Swarm Optimization* (PSO)

PSO adalah algoritma optimasi yang terinspirasi oleh perilaku sosial burung yang mencari makanan. Algoritma ini menginisialisasi populasi partikel yang bergerak dalam ruang pencarian untuk menemukan solusi optimal. Setiap partikel memperbarui posisinya berdasarkan kecepatan dan informasi terbaik yang ditemukan oleh partikel itu sendiri serta oleh seluruh populasi. PSO dapat dirumuskan sebagai berikut :

$$\mathbf{v}_i(0) = \mathbf{v}_{i,\min} + (\mathbf{v}_{i,\max} - \mathbf{v}_{i,\min}) \cdot \mathbf{r}_i \quad \ldots(9)$$

- Setiap partikel iii memiliki posisi xi dan kecepatan vi
- vi_{min} dan vi_{max} adalah batasan kecepatan untuk partikel i
- ri adalah bilangan acak dalam interval [0, 1]

Formula pembaruan posisi dalam GWO adalah sebagai berikut:

$$\mathbf{v}_i(t+1) = \omega \cdot \mathbf{v}_i(t) + c_1 \cdot \mathbf{r}_{1,i} \cdot (\mathbf{p}_{i,\text{best}} - \mathbf{x}_i(t)) + c_2 \cdot \mathbf{r}_{2,i} \cdot (\mathbf{g}_{\text{best}} - \mathbf{x}_i(t)) \quad \ldots(10)$$

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \quad \ldots(11)$$

- ω adalah faktor inersia yang mengontrol pengaruh kecepatan saat ini.
- c1 dan c2 adalah konstanta percepatan.
- r_{1i} dan r_{2i} adalah bilangan acak dalam interval [0,1].
- pi_{best} adalah posisi terbaik partikel i sepanjang waktu.
- g_{best} adalah posisi terbaik global (terbaik di antara semua partikel) sepanjang waktu.

Menurut Kennedy., et al. [3] *pseudocode* dari algoritma PSO dapat digambarkan sebagai berikut:

```
Inisiasi swarm berdimensi-n
While (t < maksimal_iterasi )
   for setiap partikel[i] = 1, ..., ns do
      if f(xi) < f(yi) then
         yi = xi;
       endif
      if f(yi) < f(ŷ) then
         ŷ = yi;
       endif
    end for
   for setiap partikel[i] = 1, ..., ns do
     perbarui kecepatan dengan persamaan (10);
      perbarui posisi dengan persamaan (11);
    end for
end while
```

2. METODOLOGI

Dalam penelitian ini, digunakan data simulasi (dummy data) untuk barang-barang yang dijual di Toko Citra Tani, karena dataset asli dari penelitian Vidiyanti, et al. [1] tidak tersedia. Data ini mencakup nama barang, jumlah barang, satuan, berat satuan, volume barang, harga beli per satuan, dan harga jual per satuan. Batasan untuk Knapsack mengikuti penelitian Vidiyanti, et al. [1], yaitu dengan berat maksimum 5.000 kg, volume maksimum 9.000.000 cm^3, dan modal maksimum sebesar Rp 20.000.000,-. Harga jual dan harga beli digunakan untuk menghitung profit. Perhitungan profit diinisiasikan menggunakan fungsi calculate_profit, dengan rumus :

$$P = \sum_{i=1}^{n} (p_i - c_i) \cdot x_i \quad \ldots(12)$$

- n adalah jumlah jenis barang.
- xi adalah jumlah barang ke-i yang dipilih (selection).
- ci adalah harga beli satuan barang ke-i
- pi adalah harga jual satuan barang ke-i

dengan kendala-kendala berikut :

$$\sum_{i=1}^{n} w_i \cdot x_i \leq W$$

$$\sum_{i=1}^{n} v_i \cdot x_i \leq V$$

$$\sum_{i=1}^{n} c_i \cdot x_i \leq M \quad \ldots(13)$$

- W adalah batasan berat total stok yang tersedia.
- V adalah batasan volume total stok yang tersedia.
- M adalah batasan modal.
- wi adalah berat satuan barang ke-i.
- vi adalah volume satuan barang ke-i.

Penerapan algoritma *Grey Wolf Optimization* (GWO)

Langkah-langkah penerapan algoritma *Grey Wolf Optimization* (GWO) adalah sebagai berikut

1) Inisialisasi Parameter dan Populasi
   a. Inisialisasi parameter GWO:
   - pop_size (jumlah populasi serigala abu-abu).
   - max_iter (jumlah iterasi maksimum)
   - pop_size dan max_iter yang akan diubah-ubah sesuai ketentuan untuk membandingkan efektifitas.

   b. Inisialisasi populasi :
   - Populasi awal (pop) terdiri dari sejumlah pop_size vektor, di mana setiap vektor adalah solusi kandidat yang berisi jumlah item yang akan dipasok untuk setiap jenis barang.

X_i= random (0,1), $i = 1....n$
y_i= round (X_i), $i = 1...n$

- Tiga serigala terbaik alpha, beta, delta (Xα, Xβ, Xδ) diinisialisasi dengan populasi awal dan skornya diinisialisasi dengan nilai -∞.

2) Fungsi Profit

Definisi fungsi profit calculate_profit(selection):

- Menghitung total berat, volume, dan modal berdasarkan jumlah barang yang dipilih (selection).
- Menghitung profit total dari barang yang dipilih.
- Memeriksa apakah constraint (berat maksimum, volume maksimum, modal maksimum) terlampaui. Jika ya, mengembalikan -∞ sebagai tanda solusi tidak valid.
- Mengembalikan nilai profit jika constraint terpenuhi.
- Profit terbaik diinisasikan sebagai calculate_profit(Xα)

3) Iterasi Algoritma GWO

Proses iterasi untuk max_iter kali :

a. Evaluasi fitness :

- Untuk setiap serigala dalam populasi, hitung fitness (profit) menggunakan fungsi calculate_profit.
- Perbarui nilai dan posisi alpha, beta, dan delta (Xα, Xβ, Xδ) berdasarkan nilai fitness.

b. Perbarui parameter a :

- Parameter a berkurang secara linier dari 2 ke 0 selama iterasi.

c. Perbarui posisi serigala :

- Untuk setiap serigala dalam populasi, posisi diperbarui berdasarkan posisi alpha, beta, dan delta menggunakan persamaan GWO.
- Posisi baru setiap serigala dihitung dan di-clamp agar tidak melebihi jumlah maksimum barang yang ada.

4) Hasil Akhir

Setelah kriteria pemberhentian selesai, atau dalam konteks ini iterasi selesai maka hasil akhir Solusi terbaik (best_solution) berupa posisi alpha dan Profit terbaik (best_profit) dihitung berdasarkan best_solution.

Penerapan algoritma *Particle Swarm Optimization* (PSO)

Langkah-langkah penerapan algoritma *Particle Swarm Optimization* (PSO) adalah sebagai berikut

1) Inisialisasi Parameter dan Populasi

a. Inisialisasi parameter PSO:

- pop_size (jumlah partikel dalam populasi).
- max_iter (jumlah iterasi maksimum)
- pop_size dan max_iter yang akan diubah-ubah sesuai ketentuan untuk membandingkan efektifitas.
- c1 = 1.5 (koefisien akselerasi kognitif).
- c2 = 1.5 (koefisien akselerasi sosial).
- w = 0.5 (inertia weight ω, mengendalikan pengaruh kecepatan sebelumnya terhadap kecepatan saat ini).

b. Inisialisasi populasi :

- Populasi awal (pop) terdiri dari sejumlah pop_size vektor, di mana setiap vektor adalah solusi kandidat yang berisi jumlah item yang akan dipasok untuk setiap jenis barang.
- Kecepatan awal vel diinisialisasi dengan nilai acak.

c. Inisiasi best position

- pbest menyimpan posisi terbaik dari setiap partikel (solusi terbaik individu).
- pbest_scores menyimpan nilai profit dari posisi terbaik setiap partikel.
- gbest menyimpan posisi terbaik secara global (solusi terbaik dari seluruh populasi).
- gbest_score menyimpan nilai profit dari posisi terbaik global.

2) Fungsi Profit

Definisi fungsi profit calculate_profit(selection):

- Menghitung total berat, volume, dan modal berdasarkan jumlah barang yang dipilih (selection).
- Menghitung profit total dari barang yang dipilih.
- Memeriksa apakah constraint (berat maksimum, volume maksimum, modal maksimum) terlampaui. Jika ya, mengembalikan -∞ sebagai tanda solusi tidak valid.
- Mengembalikan nilai profit jika constraint terpenuhi.

3) Iterasi Algoritma PSO

Proses iterasi untuk max_iter kali :

a. Perbarui kecepaan dan posisi :

- Untuk setiap partikel, hitung kecepatan baru berdasarkan kecepatan sebelumnya, posisi terbaik individu (pbest), dan posisi terbaik global (gbest).
- Pembaruan kecepatan dihitung dengan persamaan(10).
- Pembaruan posisi dihitung dengan persamaan(11).
- Posisi baru di-clamp agar tidak melebihi jumlah maksimum barang yang ada.

d. Evaluasi fitness :

- Untuk setiap partikel, menghitung fitness menggunakan fungsi calculate_profit.

- Perbarui nilai dan posisi terbaik individu (pbest) jika fitness saat ini lebih baik.
- Perbarui nilai dan posisi terbaik global (gbest) jika fitness saat ini lebih baik daripada nilai terbaik global sebelumnya.

4) Hasil Akhir

Setelah kriteria pemberhentian selesai, atau dalam konteks ini iterasi selesai maka hasil akhir Solusi terbaik (best_solution) berupa posisi gbest dan Profit terbaik (best_profit) dihitung berdasarkan best_solution.

3. HASIL DAN PEMBAHASAN

Penelitian ini akan mengevaluasi hasil implementasi algoritma *Grey Wolf Optimization* (GWO) dan membandingkannya dengan implementasi *Particle Swarm Optimization* (PSO) dalam konteks permasalahan *Multiple Constraints Bounded Knapsack*, merujuk pada penelitian yang dilakukan oleh Vidiyanti, et al. [1]. Program dijalankan di aplikasi VsCode pada laptop dengan prosesor Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz (4 CPUs), ~2.4GHz tanpa *multi threading*. Langkah awal penelitian ini adalah menguji parameter untuk melihat hasilnya. Dua parameter yang akan diuji adalah populasi (pop_size) dan iterasi maksimal (max_iter). Hasil dari pengujian parameter tersebut adalah sebagai berikut:

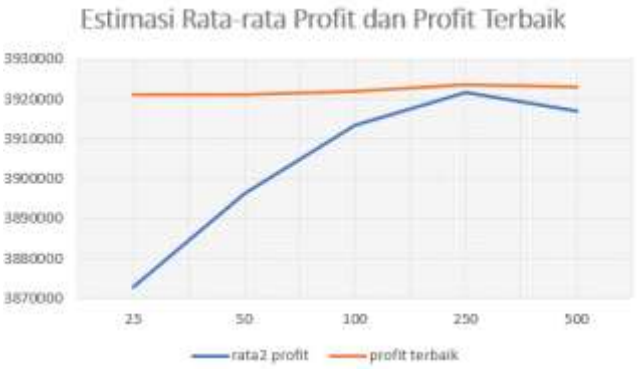
a. Uji Parameter Populasi

Pengujian parameter populasi (pop_size) dilakukan dengan menggunakan nilai parameter yang sama dengan penelitian Vidiyanti, et al. [1], yaitu 25, 75, 150, 250, dan 500 dengan maksimal iterasi sebanyak 1000 iterasi. Populasi merepresentasikan jumlah kandidat solusi (serigala). Setiap kombinasi nilai parameter dijalankan sebanyak 10 kali, kemudian dihitung rata-rata profit, rata-rata iterasi konvergen, dan rata-rata waktu komputasi dari hasil tersebut.

1) Hasil Algoritma *Grey Wolf Optimization* (GWO)

pop	Profit rata2	Profit terbaik	Profit terburuk	Rata2 iterasi	Rata2 waktu Komp
25	3872900	3921000	3736000	709.9	13.1758
50	3896300	3921000	3835000	672.3	42.2757
100	3913500	3922000	3893000	713.7	88.2747
250	3921820	3923500	3918000	660.6	228.8
500	3916980	3923000	3887000	521.2	461.176

3.a.1 Tabel hasil nilai uji parameter populasi algoritma GWO

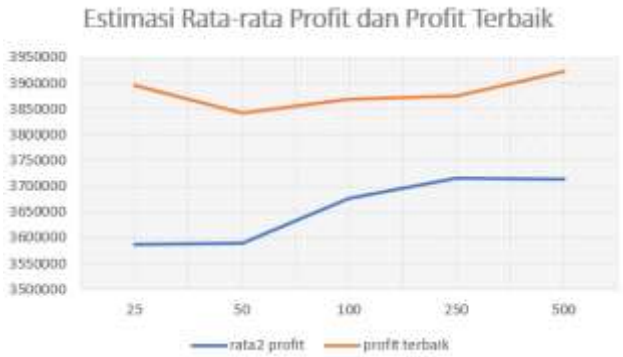


3.a.1.1. Grafik nilai uji parameter populasi algoritma GWO

2) Hasil Algoritma *Particle Swarm Optimization* (PSO)

pop	Profit rata2	Profit terbaik	Profit terburuk	Rata2 iterasi	Rata2 waktu Komp
25	3588200	3896000	3346000	111.1	3.4906
50	3589800	3842000	3335000	74.2	5.0948
100	3676900	3870000	3480000	23.9	12.5226
250	3715000	3875000	3480000	17.3	32.103
500	3713400	3922000	3473000	17.2	58.1703

3.a.2. Tabel hasil nilai uji parameter populasi algoritma PSO



3.a.2.1. Grafik nilai uji parameter populasi algoritma PSO

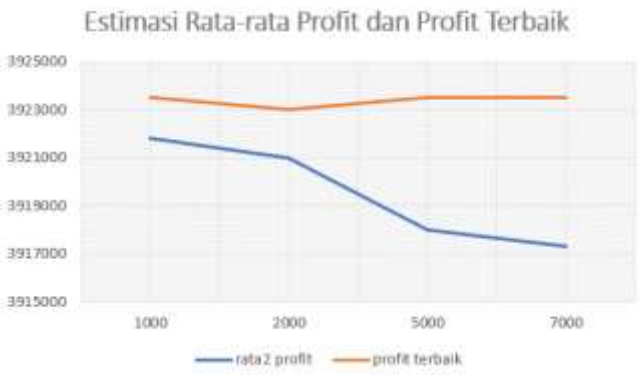
b. Uji Parameter Maksimal Iterasi

Pengujian parameter iterasi maksimal (max iter) dilakukan dengan dengan menggunakan nilai parameter yang sama dengan penelitian Vidiyanti, et al. [1] yaitu 1000, 2000, 5000, dan 7000. Sedangkan parameter populasi sebesar 250, jika meninjau dari kinerja populasi, dalam penelitian ini populasi 250 menghasilkan rata-rata profit terbaik pada kedua algoritma. Kemudian, setiap kombinasi nilai parameter dijalankan sebanyak 10 kali. Selanjutnya, dilakukan perhitungan terhadap rata-rata profit, rata-rata iterasi saat mencapai Z maksimum lokal, dan rata-rata waktu komputasi.

1) Hasil Algoritma *Grey Wolf Optimization* (GWO)

pop	Profit rata2	Profit terbaik	Profit terburuk	Rata2 iterasi	Rata2 waktu Komp
1000	3921820	3923500	3918000	660.6	228.8
2000	3921000	3923000	3917000	540.67	286.65
5000	3918000	3923000	3908000	856.67	646.42
7000	3917333	3923000	3908000	2951.3	886.19

3.b.1. Tabel hasil nilai uji parameter iterasi algoritma GWO

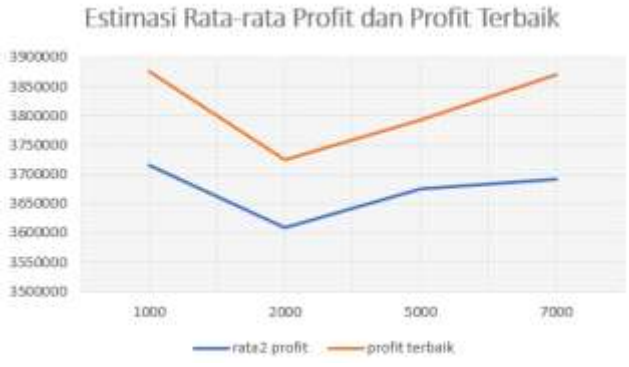


3.b.1.1. Grafik nilai uji parameter iterasi algoritma GWO

2) Hasil Algoritma *Particle Swarm Optimization* (PSO)

pop	Profit rata2	Profit terbaik	Profit terburuk	Rata2 iterasi	Rata2 waktu Komp
1000	3715000	3875000	3480000	17.3	32.103
2000	3609200	3725000	3403000	14	32.91
5000	3676000	3793000	3551000	9	83.69
7000	3691800	3870000	3512000	18	139.46

3.b.2. Tabel hasil nilai uji parameter iterasi algoritma PSO



3.b.2.1. Grafik nilai uji parameter iterasi algoritma PSO

c. Uji efektifitas algoritma

Dari hasil uji parameter populasi dapat dilihat pada tabel diatas pada penerapan algoritma *Grey Wolf Optimization* (GWO) penambahan populasi memberikan hasil profit yang mendekati optimal, dilihat dari nilai profit terburuk meningkat, rata-rata profit juga meningkat dan profit terbaik juga meningkat. Sama halnya pada algoritma *Particle Swarm Optimization* (PSO), penambahan populasi memberikan hasil profit yang cenderung mendekati optimal.

Dari hasil uji parameter maksimal iterasi dapat dilihat pada tabel diatas pada penerapan algoritma *Grey Wolf Optimization* (GWO) penambahan maksimal iterasi tidak memberikan hasil profit yang mendekati optimal, dilihat dari nilai profit rata-rata yang menurun dan nilai profit terburuk yang menurun. Akan tetapi pada algoritma *Particle Swarm Optimization* (PSO), penambahan populasi memberikan hasil profit yang cenderung mendekati optimal, dapat dilihat nilai profit rata-rata cenderung meningkat dan nilai profit terbaik dan terburuk juga cenderung meningkat.

Setelah dilakukan pengujian parameter, langkah selanjutnya yang harus dilakukan yaitu menghitung persentase deviasi dari data yang digunakan. Persentase ini dihitung untuk mengetahui seberapa optimal algoritma yang digunakan dalam penelitian ini yaitu algoritma *Grey Wolf Optimization* (GWO) dan *Particle Swarm Optimization* (PSO) yang diterapkan pada permasalahan *multiple constraints bounded knapsack*. Persentasi deviasi dapat dihitung dengan cara sebagai berikut:

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

Untuk membandingkan efektifitas penggunaan algoritma *Grey Wolf Optimization* (GWO) dan *Particle Swarm Optimization* (PSO), maka dilakukan pengujian ulang dengan nilai parameter yang sama pada kedua algoritma. Dari hasil uji parameter sebelumnya didapatkan hasil pada penerapan algoritma *Grey Wolf Optimization* (GWO), rata-rata profit paling tinggi pada saat populasi 250 dan maksimal iterasi sebanyak 1000 iterasi. Pada penerapan algoritma *Particle Swarm Optimization* (PSO), rata-rata profit paling tinggi pada saat populasi 250 dan maksimal iterasi sebanyak 1000 iterasi.

Untuk melakukan perbandingan terhadap efektifitas kedua algoritma, maka penulis menggunakan nilai parameter berupa populasi sebanyak 250 dan maksimal iterasi sebanyak 1000 iterasi, dimana pada nilai parameter tersebut, kedua algoritma mencapai nilai maksimal. Kedua algoritma akan dijalankan sebanyak 10 kali kemudian menghitung presentase standar deviasi dari profit yang dihasilkan.

1) Hasil Algoritma *Grey Wolf Optimization* (GWO)

no	Harga beli	Profit	Iterasi Konvergen	Rata2 waktu Komp	Dev (%)
1	20000000	3920000	607	151.77	0.16
2	20000000	3923000	323	129.12	0.13
3	19990000	3907000	694	161.71	0.19
4	20000000	3914000	344	144.32	0.16
5	19992000	3922000	704	126.67	0.17
6	20000000	3917000	625	128.02	0.21
7	19990000	3920000	235	126.56	0.13
8	19995000	3886000	499	126.23	0.21
9	19998000	3923000	949	125.01	0.13
10	20000000	3923000	558	125.47	0.18
R	1999650	3915500	553.8	134.49	0.167

3.c.1. Tabel hasil nilai uji efektifitas algoritma GWO

2) Hasil Algoritma *Particle Swarm Optimization* (PSO)

no	Harga beli	Profit	Iterasi Konvergen	Rata2 waktu Komp	Dev (%)
1	19996000	3500000	5	19.592	4.94
2	20000000	3847000	9	23.458	4.28
3	20000000	3536000	18	20.361	3.76
4	19995000	3862000	6	18.219	4.51
5	20000000	3790000	25	17.593	4.31
6	19990000	3425000	70	18.31	3.98
7	19990000	3695000	31	18.186	4.28
8	19990000	3525000	3	17.815	3.83
9	20000000	3908000	11	17.492	4.68
10	19990000	3715000	5	17.722	4.125
R	19995100	3680300	18.318	18.875	4.369

3.c.2. Tabel hasil nilai uji efektifitas algoritma PSO

Dari tabel diatas dapat dilihat bahwa rata-rata standar deviasi pada algoritma *Grey Wolf Optimization* (GWO) adalah sebesar 0,167% menunjukan bahwa data yang dihasilkan dari algoritma tersebut cenderung stabil, dibandingkan dengan rata-rata standar deviasi pada algoritma *Particle Swarm Optimization* (PSO) sebesar 4,3695% dimana presentase ini menunjukan data yang cukup bervariatif atau tidak stabil.

Berdasarkan analisis diatas terlihat bahwa algoritma GWO lebih optimal dibanding PSO dalam menylesaikan permasalahan *multiple constraints bounded knapsack.* Selain itu, algoritma GWO membutuhkan data yang cukup kompleks, serta populasi yang banyak, serta membutuhkan waktu komputasi yang tinggi dan iterasi konvergen yang cukup panjang dibanding algoritma PSO.

Setelah membandingkan kedua algoritma tersebut, dapat dilihat pada tabel diatas bahwa hasil paling optimal didapatkan dari data ke-9 dari hasil algoritma GWO dengan profit yang didapatkan sebesar Rp.3.923.000,00 dan pengeluaran awal atau harga beli sebesar Rp.19.998.000,00 dengan standar deviasi sebesar 0,13% yang menunjukan bahwa hasil tersebut cukup optimal.

Dari data diatas maka didapatkan identifikasi hasil pemasokan barang-barang yang dibeli oleh toko Citra Tani Jember sebagai berikut :

3) Identifikasi hasil pemasokan

No	Nama Barang	Jumlah	Satuan
1	Altex Emulsion Paint 1 kg	50	kaleng
2	Altex Emulton Paint 5 kg	30	kaleng
3	Altex Cat Synthetic - Kaleng Merah	31	kaleng
4	Altex - Meni kayu, Meni Besi	35	kaleng
5	Altex - Plamir Kayu 0,5 kg	60	kaleng
6	Altex - Plamir Kayu 1 kg	45	kaleng
7	Altex - Warna Khusus 1 kg	55	kaleng
8	A ltex - Warna Khusus 5 kg	25	kaleng
9	Lem Rajawali	40	kaleng
10	Paku 3/4"	100	kg
11	Paku 1"	100	kg
12	Bendrat	51	kg
13	Karbit:	50	kg

3.c.3. Tabel hasil identifikasi pemasokan barang dari kedua algoritma

4. KESIMPULAN

Berdasarkan hasil pembahasan diatas, didapatkan kesimpulan sebagai berikut

a. Algoritma *Grey Wolf Optimization* (GWO) lebih optimal atau lebih efektif pada kasus ini dengan parameter tetap populasi sebanyak 250 populasi dan maksimal iterasi sebanyak 1000 iterasi. Penerapan algoritma *Grey Wolf Optimization* (GWO) memperoleh rata-rata standar deviasi sebesar 0,167%, dimana presentase tersebut menunjukan data yang dihasilkan cukup konsisten. Dibandingkan dengan algoritma *Particle Swarm Optimization* (PSO) sebesar 4,3695% dimana presentase ini menunjukan data yang cukup bervariatif atau kurang konsisten.

b. Hasil optimasi terbaik diperoleh dari penerapan algoritma *Grey Wolf Optimization* (GWO) dalam menyelesaikan permasalahan *multiple constraints bounded knapsack* dengan optimasi profit sebesar Rp.3.923.000,00 dan pengeluaran awal atau harga beli sebesar Rp.19.998.000,00 dengan standar deviasi sebesar 0,13%. Solusi terbaik tersebut diperoleh dari profit paling maksimal hasil akhir simulasi data dengan parameter populasi 250 dan maksimal iterasi sebanyak 1000 iterasi dan iterasi konvergen saat Z mencapai 553,8. Besar nilai parameter populasi dan maksimal iterasi mempengaruhi algoritma dalam mencapai hasil optimal.

Dengan hasil analisis diatas, algoritma *Grey Wolf Optimization* (GWO) dinilai mampu memberikan hasil mendekati optimal. Penggunaan *device* berbeda, bahasa pemrograman berbeda, *interpreter* yang berbeda maupun penggunaan *multi Threading* dan paralelisasi mempengaruhi waktu komputasi yang didapatkan.

REFERENSI

[1] Vidiyanti Lestari., Kamsyakawuni, A., & Santoso, K. A. (2019). Implementasi Algoritma Grey Wolf Optimization (GWO) di Toko Citra Tani Jember [Implementation of the Grey Wolf Optimizer (GWO) Algorithm at Citra Tani Jember Store]. Majalah Ilmiah Matematika dan Statistika, 19(2),65-74. https://jurnal.unej.ac.id/index.php/MIMS/index.

[2] Mirjalili, S., Mirjalili, M. S. dan Lewis, A. (2014). Grey Wolf Optimization, Advances in Engineering Software 69, 46-61.

[3] Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. Proceedings of ICNN'95 - International Conference on Neural Networks.

[5] Eberhart, R., & Kennedy, J. (1995). A new Optimization using particle swarm theory. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 39-43.

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

LAMPIRAN

[1] Code dan Dataset : Jihan97ux/105222022-AI: bounded knapsack using GWO and PSO (github.com)