TENSOR Pure and Applied Mathematics Journal
September 2022 | Vol. 3 | No. 2
p-ISSN: 2723-0325
Halaman 73–84
e-ISSN: 2723-0333

Grasshopper Optimizaton Algorithm (GOA) untuk Menyelesaikan Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD)

Anatasia Naomi[1], Asri Bekti Pratiwi[2*], Herry Suprajitno[3]
[1,2,3] Department of Mathematics, Faculty of Science and Technology, Universitas Airlangga
*Email: asri.bekti@fst.unair.ac.id

Manuscript submitted : September 2022
Accepted for publication : Oktober 2022
doi : https://doi.org/10.30598/tensorvol3iss2pp73-84

Abstract: *The purpose of writing this article is to solve the Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD) using the Grasshopper Optimization Algorithm (GOA). Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD) is a problem of forming routes that serve each customer, by delivering and retrieving simultaneously. The purpose of VRPSPD is to minimize the total mileage to serve all customers with the limit that each customer is served exactly once, and the vehicle load does not exceed its maximum capacity. Grasshopper Optimization Algorithm (GOA) is an algorithm inspired by nature by mimicking the living behavior of grasshopper swarms in search of food sources. GOA has several main stages, namely initialization of parameters, determination of target grasshoppers, calculating the coefficient of decline, calculating the distance between grasshoppers, and calculating the new position of the grasshoppers. Implementation of the GOA program to complete VRPSPD using the C++ programming language using 3 types of data, data with 13 customers, 22 customers, and 100 customers. Based on the results of the running program, it can be concluded that the more iterations and the number of populations, the solution obtained tends to be better.*

Keywords: Grasshopper Optimization Algorithm, Swarm-based Algorithm, Vehicle Routing Problem, Vehicle Routing Problem with Simultaneous Pickup and Delivery.

1. Introduction

Perekonomian global tengah menghadapi Revolusi Industri 4.0 di mana para kompetitor saling berlomba untuk meningkatkan kualitas dan kuantitas produksi manufaktur guna memuaskan keinginan pelanggan. Revolusi Industri 4.0 adalah integrasi dari *Cyber Physical System* (CPS) *dan Internet of Things and Services* (IoT dan IoS) ke dalam proses industri meliputi manufaktur dan logistik serta proses lainnya [1]. Revolusi Industri 4.0 menjadi tren utama yang menggabungkan teknologi otomatisasi dengan teknologi *cyber*. Pada Revolusi Industri 4.0, waktu merupakan sebuah hal yang krusial. Perusahaan yang bergerak di bidang industri dituntut agar mampu menyediakan hasil produksi serta mengantarkannya ke tangan pelanggan dalam periode waktu yang singkat [2]. Salah satu kegiatan yang tidak bisa lepas dari perusahaan industri adalah distribusi atau pengiriman barang.

Distribusi merupakan kegiatan pemasaran yang berusaha memperlancar dan mempermudah penyampaian barang dan jasa dari produsen ke pelanggan, sehingga penggunaannya sesuai dengan yang diperlukan jenis, jumlah, harga, tempat, dan saat dibutuhkan. Namun terkadang biaya distribusi mampu menghabiskan sebagian dari harga jual barang yang dihasilkan, sehingga diperlukan suatu metode untuk mengurangi biaya distribusi [3].

Vehicle Routing Problem (VRP) adalah masalah penentuan rute yang optimal dari satu depot menuju sejumlah pelanggan yang tersebar secara geografis dengan memperhatikan sejumlah batasan [4]. Tujuan VRP adalah untuk membuat suatu rute dengan jarak terpendek, dengan sekelompok kendaraan yang sudah diketahui kapasitasnya, agar dapat memenuhi permintaan pelanggan dengan lokasi dan jumlah permintaan yang telah diketahui [5].

Salah satu perluasan dari VRP adalah Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD). VRPSPD adalah suatu permasalahan pada pembentukan rute kendaraan yang melayani setiap pelanggan, baik pengiriman maupun pengambilan barang secara bersamaan. Beberapa contoh kasus VRPSPD adalah industri air minum kemasan, dan distribusi Liquefied Petroleum Gas (LPG) **[**6].

Grasshopper Optimization Algorithm (GOA) adalah algoritma yang terinspirasi dari perilaku belalang untuk menyelesaikan permasalahan optimasi. Pada langkah awal inisialisasi dengan GOA, belalang bergerak secara luas sehingga membantu mereka untuk mencari makanan secara global (umum), dan bergerak secara lokal pada tahap terakhir optimasi sehingga memungkinkan untuk mengeksploitasi ruang pencarian [7]. Algortima ini mampu menyelesaikan permasalahan optimasi lebih baik dibandingkan beberapa algoritma lain seperti *Bat Algorithm*, *Algoritma Genetika*, *Flower Polination Algorithm*, *Firefly Algorithm*, dan *Particle Swarm Optimization Algorithm*. Hasil yang diperoleh menggunakan GOA memiliki solusi lebih baik dengan tingkat akurasi tinggi, serta mudah untuk diimplementasikan [8].

Berdasarkan latar belakang tersebut, maka akan dikaji penyelesaian *Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD)* dengan menggunakan *Grasshopper Optimization Algorithm* (GOA) untuk menentukan rute optimal untuk sejumlah kendaraan ke sejumlah pelanggan dengan memperhitungkan kapasitas muatan yang diberikan dan yang diambil.

2. Model *Vehicle Routing Problem with Simultaneous Pickup and Delivery*

Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD) adalah pengembangan dari VRP. Dalam masalah ini yang harus diperhatikan adalah adanya aktivitas pengiriman sekaligus pengambilan barang pada pelanggan melalui sebuah kendaraan dengan kapasitas yang sama. Dengan demikian, muatan pada kendaraan harus diperhatikan pada setiap pelanggan untuk memastikan kendaraan tidak melebihi muatan (*overload*). Tujuan dari permasalahan ini adalah meminimalkan total biaya berdasarkan total jarak tempuh untuk melayani seluruh pelanggan. dengan batasan setiap pelanggan mendapatkan pelayanan tepat satu kali oleh satu kendaraan dan muatan kendaraan tidak melebihi kapasitas maksimalnya.

Model matematika dari VRPSPD adalah sebagai berikut **[9]** :

Fungsi tujuan dari VRPSPD adalah

$$Min\ Z = \sum_{i \in J_0} \sum_{j \in J_0} \sum_{k \in K} C_{ij} x_{ijk} \quad (1)$$

dengan variabel keputusan:

$$x_{ijk} = \begin{cases} 1, & kendaraan\ k\ melayani\ j\ setelah\ melayani\ i \\ 0, & untuk\ yang\ lainnya \end{cases}$$

Fungsi kendala yang ada:

1. Setiap pelanggan dikunjungi tepat satu kali oleh satu kendaraan.

$$\sum_{i \in J_0} \sum_{k \in K} x_{ijk} = 1 \ , j \in J \quad (2)$$

2. Setiap kendaraan harus meninggalkan pelanggan yang telah dikunjungi.

$$\sum_{i \in J_0} x_{irk} = \sum_{j \in J_0} x_{rjk} \ , r \in J, k \in K \quad (3)$$

3. Total jumlah permintaan yang diambil dari semua lokasi yang dikunjungi kendaraan k tidak boleh melebihi beban awal kendaraan saat meninggalkan depot.

$$l \geq \sum_{i \in J_0} \sum_{j \in J} D_j x_{ijk} \ , k \in K \quad (4)$$

4. Jumlah beban permintaan pengiriman dan pengambilan pada pelanggan j oleh kendaraan k tidak melebihi sisa kapasitas kendaraan.

$$l_j \geq l - D_j + P_j - M(1 - x_{0jk}), \ j \in J, k \in K \quad (5)$$

5. Jumlah beban permintaan pengiriman dan pengambilan pada pelanggan j oleh kendaraan k tidak melebihi sisa kapasitas kendaraan k setelah melayani pelanggan i.

$$l_j \geq l_i - D_j + P_j - M\left(1 - \sum_{k \in K} x_{ijk}\right), \ i \in J, j \in J, i \neq j \quad (6)$$

6. Jumlah beban dari kendaraan k tidak melebihi kapasitas kendaraan itu sendiri.

$$l \leq C \quad (7)$$

$$l_j \leq C \ , \ j \in J \quad (8)$$

7. Setiap kendaraan hanya mempunyai satu rute kendaraan.

$$\pi_j \geq \pi + i + 1 - n\left(1 - \sum_{k \in K} x_{ijk}\right), \ i \in J, j \in J, i \neq j \quad (9)$$

8. Urutan pelayanan pelanggan j tidak boleh bernilai 75egative

$$\pi_j \geq 0 \ , \ j \in J \quad (10)$$

C_{ij} = Jarak antara pelanggan i dan j

J = Himpunan dari pelanggan yang harus dilayani

J_0 = Himpunan dari semua pelanggan termasuk depot

D_j = Jumlah permintaan yang harus dikirim ke pelanggan j

P_j = Jumlah pengambilan yang harus diambil dari pelanggan j

C = Kapasitas dari kendaraan

l_j = Jumlah beban suatu kendaraan setelah melayani pelanggan j

l = Beban suatu kendaraan saat meninggalkan depot

n = Banyaknya pelanggan

i = Indeks pelanggan awal

j = Indeks pelanggan tujuan

k = Indeks kendaraan

r = Indeks pelanggan

K = Himpunan dari kendaraan yang tersedia

π_j = Posisi dari pelanggan j

M = Angka yang sangat besar (M dapat dihitung sebagai nilai maksimum dari total permintaan pengiriman dan pengambilan atau total jarak dari setiap pelanggan)

x_{ijk} = Variabel biner yang menandakan perjalanan kendaraan k dari pelanggan i ke pelanggan j

3. *Grasshopper Optimization Algorithm* (GOA)

Grasshopper Optimization Algorithm (GOA) merupakan algoritma yang terinspirasi dari alam dengan meniru perilaku hidup kawanan belalang dalam mencari sumber makanan. Belalang merupakan salah satu serangga dengan kumpulan terbanyak di antara makhluk lainnya. Siklus hidup dari belalang, terdiri dari dua fase, fase nimfe dan fase dewasa. Pada fase nimfa, belalang cenderung bergerak secara perlahan dengan cara melompat pada jarak yang pendek, sedangkan pada fase dewasa, belalang sudah memiliki sayap sehingga dapat bergerak dengan cara terbang sehingga membuat pergerakannya secara tiba-tiba dengan jarak yang jauh.

Proses pencarian makanan belalang terbagi menjadi dua tendensi, yaitu eksplorasi dan eksploitasi. Pada eksplorasi, belalang didorong untuk bergerak secara mendadak, sedangkan ketika eksploitasi, belalang akan bergerak secara lokal untuk mencari sumber makanan yang lebih baik. Meskipun belalang sering terlihat secara individual, kenyataannya belalang termasuk serangga dengan kelompok terbesar dibandingkan makhluk hidup lainnya. Kebiasaan berkelompok pada belalang dapat ditemukan pada fase nimfa maupun fase dewasa. Selain itu, factor gravitasi dan factor angin juga mempengaruhi arah pergerakan belalang dalam mencari sumber makanan. Berdasarkan perilaku kumpulan belalang tersebut, maka dapat disusun model matematika [7]:

$$X_i = S_i + G_i + A_i \tag{11}$$

Keterangan:

X_i : posisi belalang ke-*i*.
S_i : interaksi social belalang ke-*i.*
G_i : gaya gravitasi yang dialami belalang ke-*i*
A_i : pengaruh angin yang dialami belalang ke-*i*.

Namun, persamaan tersebut tidak dapat digunakan secara langsung untuk menyelesaikan permasalahan optimasi, terutama karena belalang cepat mencapai zona nyaman, dan gerombolan belalang tersebut tidak dapat berkumpul pada suatu titik tertentu menyebabkan belalang tidak lagi mengeksplorasi dan mengeksploitasi ruang pencarian sekitarnya untuk mendapatkan sumber makanan yang lebih baik. Sehingga untuk menyelesaikan masalah pengoptimalan, model matematika dimodifikasi menjadi sebagai berikut

$$X_i^d = c\left(\sum_{\substack{j=1 \\ j\neq i}}^{N} c\,\frac{ub_d - lb_d}{2}\, s\left(\left|x_j^d - x_i^d\right|\right)\frac{x_j - x_i}{d_{ij}}\right) + \widehat{T_d} \tag{12}$$

Keterangan:

ub_d : batas atas dimensi *d*
lb_d : batas bawah dimensi *d*
$\widehat{T_d}$: solusi terbaik sementara,
c : koefisien penurunan untuk mengecilkan *attraction*, *repulsion*, dan *comfort zone* pada belalang.

Adapun nilai dari parameter c dapat dihitung menggunakan rumus

$$c = cmax - l\frac{cmax - cmin}{L} \tag{13}$$

Keterangan:

$cmax$: Nilai maksimum dari koefisien penurunan (c)
$cmin$: Nilai minimum dari koefisien penurunan (c)
l : Iterasi saat ini
L : Jumlah iterasi maksimum.

Dengan, $s(r)$ adalah gaya sosial yang dapat dihitung dengan rumus:

$$s(r) = fe^{-\frac{r}{l}} - e^{-r} \tag{14}$$

f = intensitas ketertarikan dengan belalang lain,
l = skala daya tarik suatu belalang
d_{ij} $= |x_j - x_i|$ adalah jarak antara belalang *i* dengan belalang *j*
$\widehat{d_{ij}}$ $= \frac{x_j - x_i}{d_{ij}}$ adalah vector satuan dari jarak belalang *i* ke belalang *j*.

Jenis interaksi sosial yang dilakukan oleh belalang bergantung pada besaran nilai dari gaya sosial ($s(r)$). Belalang dikatakan akan melakukan ketertarikan (*attraction*) apabila nilai fungsi $s > 0$, sedangkan akan melakukan penolakan (*repulsion*) apabila nilai fungsi $s < 0$. Ketika $s = 0$, maka tidak akan terjadi *attraction* maupun *repulsion*, hal ini disebut dengan zona nyaman (*comfort zone*).

4. Langkah Menyelesaikan VRPSPD dengan Menggunakan GOA

Langkah-langkah *Grasshopper Optimization Algorithm* (GOA) dalam menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* diberikan sebagai berikut, sedangkan flowchart disajikan pada **Gambar 1**.

a. Input Data
Data yang digunakan pada penelitian ini berupa data jarak antar pelanggan dan depot, dan data jumlah permintaan serta pengambilan barang tiap pelanggan. Terdapat 3 jenis data sekunder yang digunakan yaitu, data dengan 13 pelanggan [10], data dengan 22 pelanggan [11], data dengan 100 pelanggan [12].

b. Inisialisasi Parameter
Parameter yang digunakan untuk menyelesaikan VRPSPD dengan menggunakan GOA adalah banyak belalang, maksimal iterasi, cmax, cmin, batas atas (ub_d) dan batas bawah dimensi (lb_d). Sedangkan parameter yang digunakan dalam VRPSPD adalah kapasitas maksimal kendaraan, dan kapasitas angkut kendaraan pada awal berangkat dari depot.

c. Membangkitkan Posisi Awal Belalang
Pembangkitan posisi awal belalang dilakukan dengan membangkitkan bilangan *real* secara acak pada interval (0,1) sejumlah pelanggan. Pembangkitan ini dilakukan secara berulang sebanyak jumlah belalang.

d. Evaluasi Nilai Fungsi Tujuan
Evaluasi fungsi tujuan dilakukan dengan menjumlahkan jarak tempuh kendaraan pada masing-masing rute yang selalu berawal dan berakhir pada depot tanpa melanggar 77 angkah kendala kapasitas maksimal kendaraan dan kapasitas muatan kendaraan pada awal meninggalkan depot. Langkah-langkah dalam evaluasi fungsi tujuan adalah sebagai berikut:

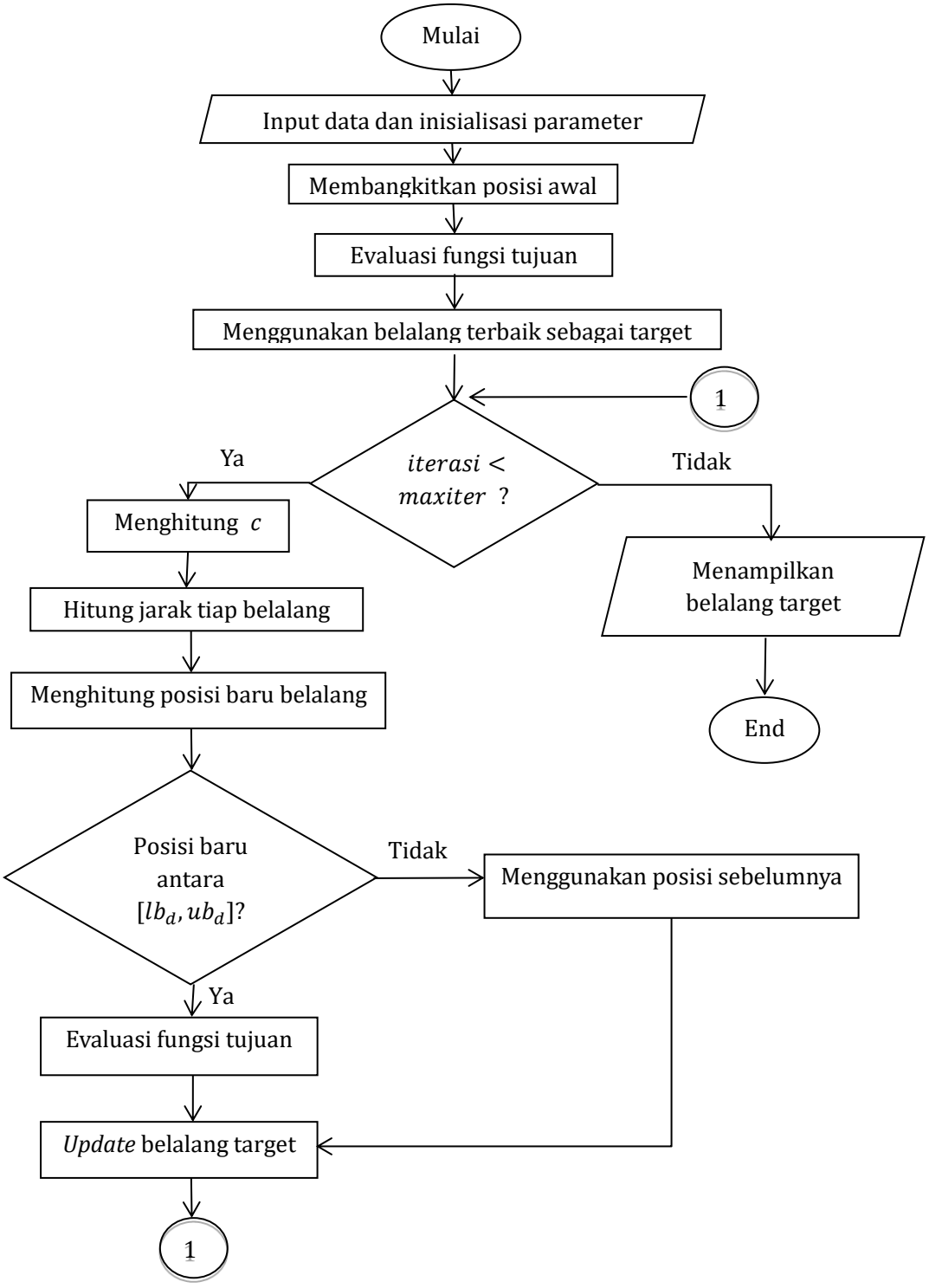
1) Setelah memperoleh posisi awal tiap belalang pada tahap sebelumnya, posisi awal belalang kemudian dikonversi dari bilangan real ke dalam bentuk kode permutasi. Setiap elemen dari posisi belalang ditransformasikan menjadi urutan pelanggan dengan cara mengurutkan bilangan real dari terkecil hingga terbesar sehingga diperoleh urutan pelanggan yang akan menjadi calon solusi.
2) Dalam representasi kode permutasi pada proses sebelumnya diperoleh urutan pelanggan yang disebut calon rute. Calon rute inilah yang akan dibentuk menjadi rute solusi tanpa melanggar kapasitas maksimum kendaraan sehingga dapat meminimumkan total jarak tempuh Ketika melayani seluruh permintaan dan pengambilan barang dari pelanggan. Kendaraan akan berangkat dari depot dan mengunjungi pelanggan satu per satu hingga 77angkah kendala kapasitas maksimal

kendaraan atau 78angkah kendala kapasitas bawaan kendaraan terpenuhi. Jika 78angkah kendala tersebut telah terpenuhi, maka kendaraan akan 78angkah ke depot dan kendaraan lain akan pergi untuk melayani pelanggan selanjutnya yang belum dilayani.

3) Proses (2) ini diulangi hingga seluruh pelanggan telah selesai dilayani. Nilai fungsi tujuan diperoleh dari menjumlahkan jarak yang ditempuh tiap kendaraan. Proses untuk menghitung nilai fungsi tujuan diulangi sejumlah populasi belalang.

e. Menentukan Belalang Target

Pada 78angkah ini, target awal diperoleh dari proses pengurutan nilai fungsi tujuan dari terkecil hingga terbesar. Belalang dengan nilai fungsi tujuan terkecil akan dijadikan sebagai belalang target.



Gambar 1. Flowchart Penyelesaian VRPSPD Menggunakan GOA

f. Update Koefisien Penurunan
Langkah selanjutnya adalah mengupdate koefisien penurunan c. koefisien penurunan c bertujuan untuk menyeimbangkan antara proses ekplorasi dengan eksploitasi pada belalang sekaligus mengecilkan *comfort zone* belalang agar sebanding dengan iterasi yang dilakukan. Perhitungan koefisien penurunan *c* menggunakan persamaan (13).

g. Menghitung jarak antar belalang
Langkah selanjutnya dari *Grasshopper Optimization Algorithm* (GOA) adalah menghitung jarak antar belalang. Jarak antar belalang dihitung dengan cara sebagai berikut:

$$d_{ij} = \sqrt{\sum_{k=1}^{n}\left(x_{i,k} - x_{j,k}\right)^2} \tag{17}$$

Keterangan:
d_{ij} : Jarak belalang ke-i dengan belalang ke-j
x_i : Posisi dari belalang ke-i
x_j : Posisi dari belalang ke-j
n : Banyak dimensi belalang

h. Update Posisi Belalang
Langkah selanjutnya adalah melakukan update posisi setiap belalang. Adapun Langkah-langkah untuk menghitung posisi baru belalang sebagai berikut:
1) Menghitung Gaya Sosial antar Belalang. Perhitungan gaya social untuk tiap belalang menggunakan persamaan (14). Parameter f dan l pada persamaan (14) dapat mengubah daerah penolakan (*repulsion*), penarikan (*attraction*), dan *comfort zone* secara signifikan, dan pada nilai tertentu, daerah penolakan (*repulsion*) bernilai sangat kecil. Oleh karena itu, dipilih nilai f dan l yang proporsional adalah 0.5 dan 1.5.
2) Menghitung Posisi Baru Belalang. Setelah memperoleh gaya social untuk tiap belalang, maka akan dihitung posisi baru belalang menggunakan persamaan (12). Posisi baru belalang yang telah diperoleh kemudian dicek Kembali apakah berada dalam batasan $[lb_d, ub_d]$, jika posisi baru tersebut ada yang melewati batasan, maka posisi baru belalang tersebut dianggap tidak memenuhi dan akan digunakan posisi lama belalang sebagai posisi barunya.

i. Menentukan Belalang Target Baru
Penentuan belalang target baru ini menggunakan informasi posisi belalang baru sehingga diperoleh nilai fungsi tujuan baru pada setiap belalang. Jika tidak ada nilai fungsi tujuan baru yang lebih baik dari sebelumnya, maka belalang target tidak perlu diupdate.

j. Cek Maksimum Iterasi
Kriteria yang harus dipenuhi agar algoritma berhenti adalah maksimal iterasi. Proses *Grasshopper Optimization Algorithm* akan selesai jika sudah mencapai maksimal iterasi sehingga diperoleh jarak tempuh yang optimal dari rute yang terbentuk.

5. Hasil dan Pembahasan

Program untuk menyelesaikan *Vehicle Routing Problem with Simultaneous Pickup and Delivery* menggunakan *Grasshopper Optimization Algorithm* diimplementasikan pada data dengan berbagai jumlah pelanggan. Bahasa pemrograman yang digunakan adalah C++ dengan menggunakan bantuan *software* Borland C++. Terdapat data dengan 3 jenis pelanggan, data dengan 13 pelanggan [10], 22 pelanggan [11], dan 100 pelanggan [12].

Parameter yang digunakan untuk menyelesaikan VRPSPD menggunakan GOA dengan data 13 pelanggan adalah kapasitas maksimal sebesar 180 barang, kapasitas angkut kendaraan Ketika meninggalkan depot adalah sebesar 70% dari kapasitas maksimalnya, atau sebesar 126 barang, sedangkan parameter pada GOA

akan digunakan nilai $cmin = 0.00001$, batas atas dimensi $(ubd) = 5$, dan batas bawah dimensi $(lbd) = -5$ [10], sedangkan untuk banyak belalang, maxiterasi, serta nilai cmax yang digunakan bervariasi. Hasil yang diperoleh selengkapnya disajikan pada Tabel 1. Berdasarkan Tabel 1 diperoleh solusi terbaik dengan total jarak sebesar 103 satuan jarak dengan jumlah belalang 100, maksimum iterasi 1000, dengan $cmax = 0.7$. Rute yang terbentuk disajikan di Tabel 2.

Tabel 1. Hasil *Running* Program dengan Data 13 Pelanggan

Jumlah Belalang	Max Iterasi	cmax		
		0.3	0.7	1
10	10	126	123	118
	100	118	117	112
	1000	116	111	108
50	10	117	116	115
	100	111	112	114
	1000	108	110	111
100	10	111	113	113
	100	107	109	110
	1000	106	**103**	106

Tabel 2. Hasil terbaik dari Data 13 Pelanggan

Kendaraan	Rute	Jarak	Total
1	0 - 1 - 2 - 11 - 12 - 13 - 0	28	
2	0 - 10 - 5 - 4 - 3 - 0	38	103
3	0 - 8 - 7 - 6 - 9 - 0	37	

Parameter yang digunakan untuk menyelesaikan VRPSPD menggunakan GOA dengan data 22 pelangggan adalah kapasitas maksimal sebesar 10500 barang, kapasitas angkut kendaraan Ketika meninggalkan depot adalah sebesar 70% dari kapasitas maksimalnya, atau sebesar 7350 barang, sedangkan parameter pada GOA akan digunakan nilai $cmin = 0.00001$, batas atas dimensi $(ubd) = 5$, dan batas bawah dimensi $(lbd) = -5$ [11], sedangkan untuk banyak belalang, maxiterasi, serta nilai cmax yang digunakan bervariasi. Hasil yang diperoleh selengkapnya disajikan pada Tabel 3.

Tabel 3. Hasil *Running* Program dengan Data 22 Pelanggan

Jumlah Belalang	Max Iterasi	cmax		
		0.3	0.7	1
10	10	156	165	163
	100	155	162	160
	1000	153	149	157
50	10	161	158	157
	100	146	157	148
	1000	142	137	143
100	10	154	155	160

	100	145	153	144
	1000	137	**122**	142

Berdasarkan Tabel 3 diperoleh solusi terbaik dengan total jarak sebesar 122 satuan jarak dengan jumlah belalang 100, maksimum iterasi 1000, $cmax = 0.7$. Rute yang terbentuk disajikan pada Tabel 4.

Tabel 4. Hasil terbaik dari Data 22 Pelanggan

Kendaraan	Rute	Jarak	Total
1	0 - 4 - 10 - 12 - 16 - 14 - 1 - 7 - 0	33	
2	0 - 9 - 8 - 2 - 5 - 3 - 22 - 21 - 0	55	122
3	0 - 20 - 18 - 6 - 15 - 11 - 19 - 13 - 17 - 0	34	

Parameter yang digunakan untuk menyelesaikan VRPSPD menggunakan GOA dengan data 100 pelangggan adalah kapasitas maksimal sebesar 700 barang, kapasitas angkut kendaraan Ketika meninggalkan depot adalah sebesar 70% dari kapasitas maksimalnya, atau sebesar 490 barang, sedangkan parameter pada GOA akan digunakan nilai $cmin = 0.00001$, batas atas dimensi $(ubd) = 5$, dan batas bawah dimensi $(lbd) = -5$ [12], sedangkan untuk banyak belalang, maxiterasi, serta nilai cmax yang digunakan bervariasi. Hasil yang diperoleh selengkapnya disajikan pada Tabel 5.

Tabel 5. Hasil *Running* Program dengan Data 100 Pelanggan

Jumlah Belalang	Max Iterasi	cmax		
		0.3	0.7	1
10	**10**	3539	3512	3451
	100	3456	3488	3461
	1000	3435	3298	3395
50	**10**	3482	3487	3393
	100	3333	3302	3332
	1000	3302	3299	3295
100	**10**	3392	3297	3376
	100	3213	3248	3340
	1000	**3208**	3227	3276

Berdasarkan Tabel 5 diperoleh solusi terbaik dengan total jarak sebesar 3208 satuan jarak dengan jumlah belalang 100, maksimum iterasi 1000, $cmax = 0.3$. Rute yang terbentuk disajikan pada **Tabel 6**.

Tabel 6. Hasil terbaik dari Data 100 Pelanggan

Tabel 6. Hasil terbaik dari Data 100 Pelanggan

Kendaraan	Rute	Jarak	Total
1	0 - 2 - 11 - 62 - 98 - 9 - 26 - 8 - 14 - 90 - 75 - 85 - 61 - 86 - 27 - 30 - 50 - 72 - 55 - 47 - 33 - 37 - 18 - 22 - 4 - 3 - 19 - 89 - 91 - 99 - 70 - 79 - 20 - 13 - 66 - 96 - 49 - 67 - 83 - 82 - 10 - 88 - 24 - 52 - 34 - 21 - 78 - 81 - 51 - 38 - 16 - 63 - 56 - 25 - 69 - 39 - 6 - 0	1731	3208
2	0 - 41 - 29 - 64 - 76 - 74 - 80 - 59 - 58 - 73 - 60 - 40 - 44 - 46 - 12 - 32 - 84 - 45 - 53 - 68 - 43 - 95 - 97 - 71 - 28 - 15 - 77 - 7 - 92 - 94 - 65 - 54 - 93 - 5 - 17 - 23 - 35 - 31 - 36 - 57 - 48 - 42 - 100 - 87 - 1 - 0	1477	

Berdasarkan hasil implementasi program pada data dengan tiga jenis pelanggan menggunakan jumlah populasi, maksimal iterasi, dan nilai cmax yang bervariasi, terlihat bahwa solusi dari nilai fungsi tujuan bervariasi. Dapat ditarik kesimpulan bahwa semakin banyak jumlah populasi dan maksimal iterasi maka solusi berupa jarak yang ditempuh cenderung lebih kecil.

6. Kesimpulan

Implementasi program untuk menyelesaikan kasus VRPSPD pada data 13 pelanggan diperoleh solusi terbaik dengan total jarak sebesar 103 satuan jarak, pada data 22 pelanggan diperoleh solusi terbaik dengan total jarak sebesar 122 satuan jarak, dan pada data 100 pelanggan diperoleh solusi terbaik dengan total jarak sebesar 3208 satuan jarak. Berdasarkan hasil *running* program pada ketiga jenis data yang digunakan, dapat disimpulkan bahwa semakin banyak jumlah belalang dan jumlah iterasi yang digunakan, maka solusi berupa jarak yang ditempuh cenderung lebih kecil.

Referensi

[1] Kagermann, H., Wahlster, W., & Helbig, J. (2013). *Recommendations for implementing the strategic initiative Industrie 4.0: Final report of the Industrie 4.0 Working Group*. Forschungsunion: Berlin, Germany.

[2] Schwab, K. (2016). *The Fourth Industrial Revolution*. Geneva: World Economic Forum.

[3] Oentoro, D., (2010). *Modern Marketing Management*. Yogyakarta: Laksbang Pressindo.

[4] Min, H. (1989). The Multiple Vehicle Routing Problem with Simultaneous Delivery and Pick-Up Points. *Transportation Research A, 23(5)*, 377-386.

[5] Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European journal of operational research, 59(3)*, 345-358.

[6] Rahmi, Y., & Murti, A., (2013). Application of Saving Matrix Method in Scheduling and Determination of Premium Distribution Routes at Malang City Gas Stations. *Journal of Mechanical Engineering, 04*, 17-26.

[7] Catay, B. (2010). A new saving-based ant algorithm for the vehicle routing problem with simultaneous pickup and delivery. *Expert systems with applications, 37(10*), 6809-6817.

[8] Saremi, S., Mirjalili, S., & Lewis, A. (2017). Grasshopper optimisation algorithm: theory and application. *Advances in engineering software*, *105*, 30-47.

[9] Solomon, M. M., & Desrosiers, J. (1988). Survey paper—time window constrained routing and scheduling problems. *Transportation science*, *22*(1), 1-13.

[10] Meraihi, Y., Gabis, A. B., Mirjalili, S., & Ramdane-Cherif, A. (2021). Grasshopper optimization algorithm: theory, variants, and applications. *IEEE Access, 9*, 50001-50024.

[11] Dethloff, J. (2001). Vehicle routing and reverse logistics: The vehicle routing problem with simultaneous delivery and pick-up: Fahrzeugeinsatzplanung und Redistribution: Tourenplanung mit simultaner Auslieferung und Rückholung. *OR-spektrum*, *23*, 79-96.

[12] Mohandas, K. & Ganesh, K., (2008). Mixed-Integer Linear Programming for Vehicle Routing Problem with Simultaneous Pick-up with Maximum Route-Length. International Journal of Applied Management and Technology, 6(1).

[13] Min, H. (1989). The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transportation Research Part A: General*, *23*(5), 377-386.

[14] Solomon, M. M., & Desrosiers, J. (1988). Survey paper—time window constrained routing and scheduling problems. *Transportation science*, *22*(1), 1-13.