Contemporary Mathematics and Applications
Vol. 2, No. 2, 2020, pp. 97-108.

Hybrid Crow Search Algorithm - Simulated Annealing untuk Menyelesaikan *Vehicle Routing Problem with Time Windows*

Bella Pristianisa Subari[1], Asri Bekti Pratiwi[1,*] & Herry Suprajitno[1]

[1]Departemen Matematika, Fakultas Sains dan Teknologi, Universitas Airlangga

[*]Corresponding author: asri.bekti@fst.unair.ac.id

Abstract. Penulisan artikel ini bertujuan untuk menyelesaikan permasalahan *Vehicle Routing Problem with Time Windows* (VRPTW) dengan menggunakan *Hybrid Crow Search Algorithm* (CSA) dengan *Simulated Annealing* (SA). *Hybrid* CSA dengan SA adalah gabungan dari kedua algoritma dengan cara melakukan proses CSA kemudian hasil terburuknya diperbaiki dengan proses SA untuk sepuluh iterasi pertama. Proses algoritma ini dimulai dengan inisialisasi parameter, membangkitkan posisi dan memori awal, menghitung fungsi tujuan, memperbarui posisi gagak, menghitung fungsi tujuan posisi baru gagak, *update* memori gagak, menentukan solusi terburuk dari posisi gagak kemudian dilakukan modifikasi, hasil modifikasi dengan SA menggantikan solusi terburuk pada posisi gagak, proses berlanjut sampai maksimal iterasi dipenuhi dan menentukan solusi terbaik dari memori gagak. Berdasarkan hasil implementasi pada tiga tipe data dapat disimpulkan bahwa semakin banyak jumlah iterasi, jumlah gagak, dan proses *Simulated Annealing* maka nilai fungsi tujuan yang diperoleh cenderung semakin baik, sedangkan nilai probabilitas kewaspadaan (AP) tidak memberikan pengaruh pada solusi permasalahan.

Keywords: *crow search algorithm, hybrid, simulated annealing, VRPTW.*

1 Pendahuluan

Dunia industri saat ini memiliki perkembangan yang sangat pesat, perkembangan industri tersebut sebanding dengan pertumbuhan perusahaan. Perusahaan tentunya membutuhkan perancangan sistem distribusi yang efektif untuk memperlancar sistem produksinya, salah satunya dalam segi transportasi [1]. Khususnya transportasi barang yang merupakan salah satu hal penting dalam sistem distribusi bagi perusahaan [2], sehingga transportasi merupakan faktor penting dalam pendistribusian barang. Setiap perusahaan akan membuat cara agar mendapat keuntungan yang optimal yaitu dengan meminimalkan biaya distribusi yang dapat diminimalkan dengan meminimalkan rute pendistribusian. Masalah pendistribusian ini termasuk dalam *Vehicle Routing Problem* (VRP). VRP merupakan permasalahan penentuan rute distribusi yang terdiri dari satu depot yang melayani sejumlah pelanggan, rute pengiriman harus dimulai dan berakhir di depot, dan pengiriman dilakukan oleh beberapa kendaraan dengan kapasitas tertentu. Semua permintaan pelanggan harus dipenuhi dan setiap pelanggan dilayani oleh satu kendaraan tepat satu kali [3]. Namun beberapa kasus tertentu, beberapa produsen dihadapkan dengan

batasan waktu dalam penerimaan barang. Kasus ini merupakan perluasan dari VRP yang disebut dengan *Vehcile Routing Problem with Time Windows* (VRPTW) [4].

VRPTW merupakan perancangan serangkaian rute biaya minimum, yang berawal dan berakhir pada depot pusat, untuk armada kendaraan yang melayani pelanggan dengan permintaan yang diketahui. Setiap pelanggan harus dialokasikan tepat satu kendaraan agar kapasitas kendaraan tidak melebihi batas. Pelayanan pada pelanggan dilakukan sesuai dengan *time winodws* yang ditentukan oleh waktu buka dan waktu tutup ketika pelanggan mengizinkan dilakukannya pelayanan [5].

Crow Search Algorithm (CSA) adalah algoritma yang terinspirasi dari kecerdasan perilaku gagak. Gagak berperilaku seperti intelijen memiliki kewaspadaan, mengenali wajah, memperingatkan kawanan yang berpotensi tidak aman, dan mengingat tempat menyembunyikan makanan [6]. CSA memiliki dua parameter yaitu jarak terbang dan probabilitas kewaspadaan gagak. CSA dapat membuat proses pencarian optimasi menjadi area yang menjanjikan dengan waktu yang singkat tetapi terdapat kemungkin untuk terjebak dalam optimal local [7].

Simulated Annealing (SA) adalah algoritma dengan metode yang dianalogikan dengan proses *annealing* atau proses pendinginan material suatu logam padat dan beku menjadi kristal dengan energi minimum. Proses *annealing* memerlukan pengontrolan yang hati-hati terhadap suhu dan laju pendinginan [8].

Berdasarkan penjelasan dua algoritma diatas dapat memperkaya perilaku pencarian dari optimasi dan meningkatkan kamampuan serta efisiensi dalam menyelesaikan permasalahn VRPTW dengan menggunakan *Hybrid* CSA dengan SA karena algoritma CSA termasuk algoritma baru dan belum ada penelitian yang menyelesaikan permasalahan VRPTW dengan algoritma tersebut. Namun tampaknya algoritma ini masih terdapat kemungkinan munculnya optimal local [7]. Sehingga dilakukan *hybrid* dengan SA karena algoritma tersebut dapat melepaskan diri dari optimal lokal dengan harpaan dapat menghasilkan solusi yang lebih baik, untuk meningkatkan eksplorasi terutama pada awal eksekusi untuk menghindari optimal lokal [9].

2 *Vehicle Routing Problem with Time Windows*

Vehicle Routing Problem (VRP) apabila ditambahkan kendala *time window* pada masing-masing pelanggan, maka permasalahan tersebut menjadi *Vehicle Routing Problem Time Windows* (VRPTW), dimana VRPTW sendiri adalah permasalahan bagaimana sebuah depot, pusat distribusi barang, dengan sejumlah kendaraan berkapasitas tertentu melayani sejumlah pelanggan pada lokasi terpisah, dengan permintaan dan batasan *time window* tertentu, dengan tujuan meminimalkan total biaya perjalanan, tanpa mengabaikan batasan kapasitas kendaraan [4]. Desain rute dilakukan sedemikian hingga setiap pelanggan

hanya dikunjungi sekali oleh satu kendaraan, dan setiap kendaran memulai dan mengakhiri rutenya pada depot [2]. Tujuan dari VRPTW adalah menentukan rute yang optimal untuk sejumlah kendaraan agar pelanggan bisa dilayani tepat waktu [10].

Menurut [10] model matematika dari VRPTW disajikan sebagai berikut:

$$Min\ Z = \sum_{k=1}^{K} \sum_{i=0}^{L} \sum_{j=0}^{L} c_{ij} x_{ijk} \quad (1)$$

Dengan kendala-kendala:

1. Total permintaan dari setiap kota dalam rute yang dilalui setiap kendaraan tidak melebihi kapasitas muat kendaraan

$$\sum_{i=1}^{L} g_i y_{ik} \leq q, \forall\ k = 1,2, \dots, K \quad (2)$$

2. Setiap pelanggan dikunjungi tepat satu kali oleh satu kendaraan:

$$\sum_{k=1}^{K} y_{ik} = 1, \forall\ i = 1,2, \dots, L \quad (3)$$

3. Memastikan bahwa setiap rute yang dilalui telah lengkap:

$$\sum_{i=0}^{L} x_{ijk} = y_{jk,}\ \forall\ k, j, \quad k = 1,2, \dots, K, \quad j = 1,2, \dots, L \quad (4)$$

$$\sum_{j=0}^{L} x_{ijk} = y_{ik,}\ \forall\ k, i, \quad k = 1,2, \dots, K, \quad i = 1,2, \dots, L \quad (5)$$

$$\sum_{i \in S, j \in S}^{L} \sum_{i \neq j}^{L} x_{ijk} \leq |S| - 1, S \subset \{1,2, \dots, L\}, \quad \forall\ k = 1,2, \dots, K \quad (6)$$

4. Memastikan bahwa dua pelanggan yang saling berdekatan berada dalam rute yang sama:

$$sl_i + t_{ij} \leq se_j, \forall\ k, i, j \quad k = 1,2 \dots, K, i = 1,2, \dots, L, \quad j = 1,2, \dots, L \quad (7)$$

5. Batasan pelayanan kepada pelanggan (*time window*):

$$e_i \leq se_i \leq l_i\ \forall\ i = 1,2, \dots, L \quad (8)$$

Dengan,

$$y_{ik} = \begin{cases} 1, permintaan\ dari\ pelanggan\ i\ dipenuhi\ kendaraan\ k \\ 0, yang\ lainnya \end{cases}$$

$$x_{ijk} = \begin{cases} 1, kendaraan\ k\ melakukan\ perjalanan\ dari\ pelanggan\ i\ ke\ j \\ 0, yang\ lainnya \end{cases}$$

Keterangan:

K : Total Kendaraan, $K(k = (1,2, \dots, K)$

L : Total Pelanggan, $L(i = 1,2, \dots, L)$

x_{ijk} : Kendaraan k yang melayani pelanggan j setelah mengunjungi pelanggan i

c_{ij} : Biaya pengiriman dari pelanggan i ke pelanggan j (dapat juga berarti jarak)

q : Kapasitas kendaraan

g_i : Permintaan dari pelanggan $i(i = 1,2, \dots, L)$

$[e_i, l_i]$: *Time windows* pada pelanggan i

e_i : Waktu paling awal untuk melakukan pelayanan pada pelanggan i

l_i : Waktu paling akhir untuk melakukan pelayanan pada pelanggan i

se_i : Waktu ketika pelayanan pada node pelanggan i dimulai

se_j : Waktu ketika pelayanan pada node pelanggan j dimulai

sl_i : Waktu ketika pelayanan pada node pelanggan i berakhir

t_{ij} : Waktu perjalanan dari node pelanggan i ke node pelanggan j

S : Himpunan pelanggan

$|S|$ adalah kardinalitas S

3 *Crow Search Algorithm*

Crow Search Algorithm (CSA) adalah algoritma pencarian metaheusristik yang terinspirasi dari perilaku cerdas gagak yang menyimpan kelebihan makanan mereka di tempat persembunyian dan mengambilnya saat makanan dibutuhkan. Burung gagak diketahui mengawasi burung-burung lain, mengamati di mana burung-burung lain menyembunyikan mkanan mereka dan mencurinya begitu pemiliknya pergi. Jika seekor gagak melakukan pencurian, akan diperlukan tindakan pencegahan ekstra seperti memindahkan tempat persembunyian untuk menghindari menjadi korban di masa depan [7].

Berdasarkan pada perilaku gagak yang cerdas, algoritma metaheuristik berbasis populasi CSA akan dikembangkan. Prinsip-prinsip dari CSA sebagai berikut [7]:

1. Gagak hidup dalam bentuk kawanan.
2. Gagak menghafal posisi tempat persembunyian mereka.
3. Gagak mengikuti satu sama lain untuk melakukan pencurian.
4. Gagak melindungi tempat penyimpanan mereka dari kemungkinan pencurian.

a. Pergerakan Kawanan Gagak

Jumlah dari gagak (jumlah kawanan) adalah N dan posisi dari gagak-i pada saat (iterasi) $iter$ pada ruang pencarian ditentukan dari vektor $X^{i,iter}(i = 1,2,\dots,N; iter = 1,2,\dots,iter_{\max})$. Sedangkan $iter_{max}$ adalah nilai maksimum dari iterasi. Setiap gagak memiliki memori dimana posisi dan tempat persembunyiannya telah disimpan. Pada iterasi $iter$, posisi dan tempat sembunyi dari gagak-i ditunjukkan oleh $m^{i,iter}$. Posisi ini adalah yang terbaik bagi gagak-i. Setiap gagak akan mengingat pengalaman posisi terbaiknya. Gagak bergerak dilingkungan dan mencari sumber makanan yang lebih baik (tempat persembunyian) [7].

b. Probabilitas Kewaspadaan (*AP*)

Diasumsikan bahwa pada iterasi $iter$, gagak-j ingin mengunjungi tempat persembunyian, $m^{j,iter}$. Pada iterasi ini, gagak-i memutuskan untuk mengikuti gagak-j guna mendekati tempat persembunyian dari gagak-j. Pada kasus ini ada 2 kejadian yang mungkin terjadi, yaitu:

Keadaan 1: Gagak-j tidak tahu bahwa gagak-i mengikutinya. Sebagai hasilnya, gagak-i akan mendatangi tempat persembunyian dari gagak-j. Pada kasus ini, posisi baru dari gagak-i diperoleh sebagai berikut:

$$x^{i,iter+1} = x^{i,iter} + r_i \times fl \times \left(m^{j,iter} - x^{i,iter}\right) \quad (9)$$

dimana r_i adalah bilangan *real* dengan distribusi seragam antara 0 dan 1 dan fl menunjukkan jarak terbang dari gagak-i.

Keadaan 2: Gagak-j tahu bahwa gagak-i mengikutinya. Sebagai hasilnya, untuk melindungi persembunyiannya dari pencurian, gagak-j akan mengelabui gagak-i dengan terbang ke posisi yang lain dari ruang pencarian.

Secara keseluruhan keadaan 1 dan 2 dapat ditunjukkan seperti berikut:

$$x^{i,iter+1} = \begin{cases} x^{i,iter} + r_i \times fl \times \left(m^{j,iter} - x^{i,iter}\right), r_j \geq AP \\ random, lainnya \end{cases} \quad (10)$$

dimana $0 < r_j < 1$ berdistribusi seragam dan AP menujukkan probabilitas kewaspadaan [7].

4 *Simulated Annealing*

Simulated Annealing adalah algoritma yang meniru proses *Annealing* pada pemrosesan material suatu logam padat dan beku menjadi kristal dengan energi minimum dengan ukuran kristal lebih besar sehingga dapat mengurangi cacat pada struktur logam. Proses *annealing* memerlukan kontrol yang hati-hati terhadap suhu dan laju pendinginan (sering disebut sebagai jadwal *annealing*) agar diperoleh struktur yang keras [8].

Terdapat empat hal utama yang perlu diperhatikan dalam penggunaan algoritma ini untuk memodelkan sebuah masalah yaitu [11]:

1. Representasi yang akurat dari konfigurasi dalam suatu permasalahan.
2. Proses modifikasi, langkah acak atau perubahan apa yang harus dilakukan terhadap elemen-elemen konfigurasi untuk menghasilkan konfigurasi berikutnya.
3. Fungsi evaluasi atau fungsi objektif yang dapat menyatakan baik buruknya suatu solusi terhadap permasalahan.
4. Jadwal penurunan suhu dalam proses *annealing* dan berapa lama proses tersebut dilakukan.

a. Probabilitas Penerimaan

Dalam permaslahan meminimalkan sebarang perubahan yang lebih baik atau menurunkan nilai fungsi tujuan f akan diterima, sedangkan untuk beberapa perubahan yang menaikkan nilai fungsi tujuan f juga diterima dengan menghitung probabilitas P. Probabilitas P disebut probabilitas penerimaan, ditentukan dengan rumus sebagai berikut:

$$P = e^{-\frac{\delta f}{T}} \qquad (11)$$

T merupakan suhu untuk mengontrol proses *annealing* dan δf merupakan perubahan dari fungsi tujuan. Perubahan akan diterima jika $P > r$ dengan r merupakan bilangan real acak pada interval $[0,1]$ [11].

b. Penurunan Suhu

Persoalan penting yang lain adalah bagaimana mengontrol *annealing* atau proses pendinginan sehingga sistem mengalami penurunan suhu secara bertahap dari suhu yang tinggi sampai akhirnya membeku ke kondisi minimum global. Ada du acara untuk mengontrol nilai pendinginan atau penurunan suhu. Biasanya digunakan dua jadwal *annealing* (jadwal pendinginan) yakni pendinginan linier dan geometris. Untuk proses pendinginan linier, sebagai berikut

$$T = T_0 - \beta t \; atau \; T = T_0 - \delta t \qquad (12)$$

dengan T_0 adalah suhu awal dan t adalah waktu semu untuk iterasi. β adalah nilai penurunan, dan dipilih sedemikian sehingga $T \to 0$ ketika $t \to tf$ (jumlah maksimum iterasi), biasanya digunakan

$$\beta = \frac{T_0}{tf} \qquad (13)$$

Pendinginan geometris pada dasarnya menurunkan suhu dengan menggunakan faktor penurunan $0 < \alpha < 1$ sedemikian sehingga T digantikan oleh αT atau

$$T(t) = T_0 \alpha^t, t = 1,2, \dots, tf \qquad (14)$$

Keuntungan dari metode ini adalah bahwa $T \to 0$ ketika $t \to \infty$, dan dengan demikian tidak perlu untuk menentukan jumlah maksimum iterasi tf. Proses pendinginan harus cukup lambat yang memungkinkan sistem mudah stabil [11].

5 *Hybrid* CSA dengan SA untuk Menyelesaikan VRPTW

Pada bab ini akan diberikan prosedur dari *Hybrid* CSA dengan SA untuk menyelesaikan VRPTW. *Pseudo code* dari *Hybrid* CSA dengan SAuntuk menyelesaikan VRPTW disajikan pada Gambar 1.

Langkah 1: Input data dan Inisialisasi Parameter CSA dan SA

Pada langkah ini menginputkan data posisi depot, posisi pelanggan, permintaan masing-masing pelanggan, *time window* tiap pelanggan, banyaknya kendaraan, dan kapasitas maksimum kendaraan. Kemudian inisialisasi parameter yaitu banyak gagak (N), jarak terbang (fl), banyak iterasi ($iter_{max}$), probabilitas kewaspadaan (AP), suhu awal (T_0), suhu akhir (T_{ak}), faktor penurunan suhu (α).

Langkah 2: Inisialisasi Posisi dan Memori Awal Gagak

Inisialisasi posisi gagak dengan membangkitkan bilangan real acak pada interval $(0,1)$ sebanyak jumlah pelanggan yang akan mewakili posisi gagak (X_i) dimana $memori\ awal\ =\ posisi\ awal.$

Langkah 3: Evaluasi Fungsi Tujuan

Mengevaluasi fungsi tujuan dengan menghitung total jarak yang dilalui kendaraan tiap rute.

Langkah 4: Memperbarui Posisi Gagak

Pada langkah ini gagak akan memperbarui posisi gagak dengan memilih secara acak gagak lain untuk diikutinya. Sehingga pada langkah ini akan dibangkitkan bilangan $real\ r_j$ pada interval $(0,1)$ secara acak. Kemudian untuk menentukan posisi baru $x^{i,iter+1}$ akan menggunakan Persamaan (10) dimana r_i merupakan bilangan real yang dibangkitkan secara acak pada interval (0,1). Selanjutnya memastikan bahwa kondisi $i = N$ adalah benar sehingga dapat menuju ke langkah selanjutnya.

Langkah 5: Evaluasi Fungsi Tujuan Posisi Baru

Mengevaluasi fungsi tujuan dengan menghitung total jarak yang dilalui kendaraan tiap rute.

Langkah 6: *Update* Memori Gagak

Pada langkah ini akan dilakukan *update* memori gagak dengan membandingkan nilai fungsi tujuan memori awal gagak dengan nilai fungsi tujuan posisi baru gagak. Syarat yang digunakan untuk memperbarui memori gagak adalah sebagai berikut:

$$m^{i,iter+1} = \begin{cases} x^{i,iter+1}, jika\ f\left(x^{i,iter+1}\right) < f\left(m^{i,iter}\right) \\ m_{(i,iter)}, lainnya \end{cases} \quad (15)$$

dimana f menunjukkan nilai fungsi tujuan. Jika nilai fungsi tujuan posisi baru gagak lebih baik daripada nilai fungsi tujuan memori gagak, gagak akan memperbarui memori dengan posisi baru [7].

Langkah 7: Mengecek Kondisi Iterasi

Pada langkah ini akan dipastikan bahwa kondisi iterasi adalah $iter < 10$, jika kondisi tersebut dipenuhi maka meuju Langkah 8 tetapi jika kondisi tersebut tidak terpenuhi maka menuju Langkah 9.

Langkah 8: *Simulated Annealing*

Pada 10 iterasi pertama akan melalui proses *hybrid* dimana untuk setiap 1 iterasi pada CSA mengalami 100 iterasi pada SA. Solusi awal yang digunakan pada langkah modifikasi SA merupakan posisi baru dari CSA yang memiliki nilai fungsi tujuan terburuk.

Langkah 9: Mengecek Kondisi Iterasi Maksimum

Langkah 4-9 akan melalui perulangan hingga kondisi iterasi yang telah ditentukan tercapai. Ketika kriteria iterasi terpenuhi, posisi terbaik dalam memori akan menjadi solusi terbaik dari permasalahan VRPTW.

Prosedur Hybrid Crow Search Algorithm dengan Simulated Annealing untuk menyelesaikan VRPTW
begin
Input data VRPTW ( );
Inisialisasi parameter CSA dan SA( );
Bangkitkan posisi awal ( ); //Proses *Crow Search Algorithm* (CSA)
$memori\ awal\ =\ posisi\ awal$;
Evaluasi fungsi tujuan posisi awal gagak ();
do
Memperbaharui posisi gagak ();
Evaluasi fungsi tujuan posisi baru gagak dan *update* memori gagak ();
if ($0 \leq iterasi\ \&\ iterasi \leq 10$)
Menentukan gagak terburuk ($f(x)$) ();
do
do
Modifikasi(); //Proses *Simulated Annealing*
Evaluasi fungsi tujuan hasil modifikasi $(f^*(x))$();
$\delta f = f^*(x) - f(x)$;
if ($\delta f > 0$)
$P(probabilitas\ penerimaan) \leftarrow e^{\frac{-\sigma f}{T}}$;
$r \leftarrow$ **random** (0,1);
if ($P > r$)
terima hasil modifikasi;
$sukses \leftarrow sukses + 1$;
else
tolak hasil modifikasi;
$gagal \leftarrow gagal + 1$;
end if
else
terima hasil modifikasi;
$sukses \leftarrow sukses + 1$;
end if
while ($sukses < 3 || gagal < 3$);
$T^* \leftarrow T^t - \alpha T_0$;
while ($T^* \leq T_{ak}$);
Menyimpan solusi terbaik ();
else
Menyimpan solusi terbaik ();
end if
while ($iterasi \leq maxiterasi$)
Menentukan gagak terbaik dari memori();
End

Gambar 1. Prosedur *Hybrid* CSA dengan SA

6 Hasil dan Pembahasan

Program *Hybrid* CSA dengan SA untuk menyelesaikan VRPTW yang dibuat dengan bahasa pemrograman C++ menggunakan Borland C++. Program telah

diimplementasikan dan percobaan dalam beberapa data telah dilakukan. Data yang digunakan diperoleh dari http://neo.lcc.uma.es/vrp/vrp-instances/capacitated-vrp-with-time-windows-instances/. Data yang diambil terdiri dari tiga jenis, yaitu data kecil dengan 25 pelanggan, data sedang dengan 50 pelanggan, dan data besar dengan 100 pelanggan. Berikut ini adalah solusi terbaik dari hasil *running* program yang dibuat dengan nilai parameter $Maxiter$, N, dan AP berbeda-beda serta nilai $fl = 2$ [7], $T_0 = 1000, \alpha = 0.01, T_{ak} = 0$, jumlah kendaraan 25, kapasitas tiap kendaraan 1000. Hasil *running* program menggunakan data kecil disajikan pada Tabel 1.

Tabel 1 Hasil *Running* Program Data Kecil

Maks Iterasi	**Popsize**	***AP***		
		0.2	**0.5**	**0.8**
10	**10**	748	829	844
	50	758	702	797
	100	719	707	715
100	**10**	718	629	663
	50	621	667	607
	100	605	557	612
1000	**10**	567	581	654
	50	613	563	576
	100	577	517	568

Hasil *running* program menggunakan data sedang yang disajikan pada Tabel 2.

Tabel 2 Hasil *Running* Program Data Sedang

Maks Iterasi	Popsize	*AP*		
		0.2	**0.5**	**0.8**
10	**10**	2273	2006	2223
	50	2061	2169	2200
	100	2140	2028	2182
100	**10**	1990	1972	2082
	50	1706	1777	1939
	100	1778	1783	1838
1000	**10**	1780	1710	1572
	50	1701	1616	1526
	100	1642	1675	1666

Hasil *running* program menggunakan data besar yang disajikan pada Tabel 3.

Tabel 3 Hasil *Running* Program Data Besar

Maks Iterasi	Popsize	*AP*		
		0.2	**0.5**	**0.8**
10	**10**	4251	4171	4245
	50	4129	4196	4191
	100	4089	4102	4100
100	**10**	3893	4016	4123
	50	3680	3886	4012
	100	3744	3880	3960
1000	**10**	3717	3620	3718
	50	3614	3540	3519
	100	3389	3413	3604

Berdasarkan hasil *running* program, diketahui bahwa semakin besar maksimum iterasi dan jumlah gagak atau *popsize* (N) maka nilai fungsi tujuan yang diperoleh cenderung semakin baik, sedangkan nilai probabilitas kewaspadaan (AP) tidak memberikan pengaruh terhadap nilai fungsi tujuan.

7 Kesimpulan

Implementasi program untuk contoh kasus dengan menggunakan data kecil dengan jumlah pelanggan 25, data sedang dengan jumlah pelanggan 50, dan data besar dengan jumlah pelanggan 100 yang masing-masing dilakukan dengan beberapa jumlah iterasi, jumlah gagak atau *popsize* (N), dan probabilitas kewaspadaan (AP). Dari hasil implementasi dapat disimpulkan bahwa semakin besar jumlah iterasi dan jumlah gagak atau *popsize* (N) maka nilai fungsi tujuan cenderung semakin baik, sedangkan nilai

probabilitas kewaspadaan (AP) tidak memberikan pengaruh terhadap fungsi tujuan yang diperoleh.

8 Daftar Pustaka

[1] Christofides, N., A. Migozzi dan P. Toth. 1979. "*The Vehicle Routing Problem in Combinatorial Optimization*". N. Christofides, A. Mongozzi, P. Toth and C. Sandi. (eds). J. Wiley. 315-338.

[2] Braysy, O. dan Dullaert, W., 2003, *A Fast Evolutionary Metaheuristic For The Vehicle Routing Problem with Time Windows*, International Journal on Artificial Intelligence Tools, Vol. 12, No. 2, 153-172.

[3] Baker, B.M. dan Ayehew, M.A.,2003,. A Genetic Algorithm For The Vehicle Routing Problem. Elsevier:*Computers & Operations Research*, Vol 30 (5), pp. 787-800.

[4] Kallehauge, B., J. Larsen, dan O.B.G. Marsen. 2001. *Lagrangean Duality Applied on Vehicle Routing Problem with Time Windows*, Technical Report. IMM:Technical University of Denmark.

[5] Nasser A. El-Sherbeny. 2010. *Vehicle Routing With Time Windows: An Overview of Exact, Heuristic and Metaheuristic Methods*. Mathematics Department, Faculty of Science, Al-Azhar University, Nasr City 1184, Cairo, Egypt.

[6] Diaz, P., Cuevas, E., dan Avalos, O., 2018, *An Improved Crow Search Algorithm Applied to Energy Problems*, Guadalajara University, Departement of Electronica, Guadalajara, Mexico.

[7] Askarzadeh, A., 2016, *A Novel Metaheuristic Method for Solving Constrained Engineering Optimization Problems: Crow Search Algorithm*, Graduate University of Advance Technology, Institute of Science and High Technology and Environmental Science, Department of Energy Management and Optimization, Kerman, Iran.

[8] Chibante, R., 2010, *Simulated Annealing Theory and Applications*, Sciyo, Crotia.

[9] Assad, A., dan Deep, K., 2018, *A Hybrid Harmony Search and Simulated Annealing Algorithm for Continuous Optimization*, Indian Institute of Technology Roorkee, Department of Mathematics, Roorkee, India.

[10] Pan, F., Chinming, Y., Wang, K. dan Cao, j.,2013., Research on the Vehicle Routing Problem with Time Windows Using Firefly Algorithm, *Journal of Computers.*,Vol 8., No. 9., pp. 2256-2257.

[11] Kirkpatrick, S., Gelatt, C. D. dan Vecchi, M. P., 1983, *Optimization by Simulated Annealing*, *Science*, 220, pp, 671-680