Contemporary Mathematics and Applications
Vol. 1, No. 1, 2019, pp. 34-45.

Penerapan *Cuckoo Search Algorithm* (CSA) untuk Menyelesaikan *Uncapacitated Facility Location Problem* (UFLP)

Asri Bekti Pratiwi[1,*], Nur Faiza[2] & Edi Winarko[3]

[1,2,3]Departemen Matematika Matematika, Fakultas Sains dan Teknologi, Kampus C Universitas Airlangga, Jl. Mulyorejo, Surabaya

*Corresponding Author: asri.bekti@fst.unair.ac.id

Abstract. The aim of this research is to solve Uncapacitated Facility Location Problem (UFLP) using Cuckoo Search Algorithm (CSA). UFLP involves *n* locations and facilities to minimize the sum of the fixed setup costs and serving costs of *m* customers. In this problem, it is assumed that the built facilities have no limitations in serving customers, all request from each customers only require on facility, and one location only provides one facility. The purpose of the UFLP is to minimize the total cost of building facilities and customer service costs. CSA is an algorithm inspired by the parasitic nature of some cuckoo species that lay their eggs in other host birds nests. The Cuckoo Search Algorithm (CSA) application program for resolving Uncapacitated Facility Location Problems (UFLP) was made by using Borland C ++ programming language implemented in two sample cases namely small data and big data. Small data contains 10 locations and 15 customers, while big data consists 50 locations and 50 customers. From the computational results, it was found that higher number of nests and iterations lead to minimum total costs. Smaller value of p_α brought to better solution of UFLP.

Keywords: *Cuckoo Search Algorithm (CSA), Uncapacitated Facility Location Problem (UFLP).*

1 Pendahuluan

Seiring dengan berkembangnya ilmu pengetahuan dan teknologi yang semakin maju, hal ini dapat memicu munculnya industri baru yang mengakibatkan daya saing dalam dunia industri semakin kompetitif. Dalam dunia industri diperlukan strategi untuk setiap permasalahan yang dihadapi agar industri dapat mempertahankan kepercayaan customer sehingga dapat bertahan dan berkembang. Salah satu permasalahan penting dalam dunia industri adalah penempatan fasilitas pada suatu lokasi. Fasilitas dapat didefinisikan sebagai tempat berkumpulnya orang, material, mesin dan sebagainya. Dalam dunia industri diperlukan manajemen fasilitas sehingga dapat mencapai tujuan yakni memproduksi atau menyediakan jasa dengan biaya rendah, kualitas tinggi, dan menggunakan sumber daya yang minimum [1].

Secara umum, permasalahan penempatan fasilitas pada suatu lokasi (*facility location problem*) dapat didefinisikan sebagai penempatan beberapa fasilitas pada beberapa lokasi

yang mungkin sehingga seluruh customer dapat dilayani dengan biaya minimal. Masalah penempatan fasilitas dalam suatu lokasi dapat diklasifikasikan menjadi dua berdasarkan batasan masalah yang digunakan, yaitu pengalokasian jumlah customer yang terbatas (*Capacitated*), dan pengalokasian jumlah customer yang tidak terbatas (*Uncapacitated*). *Capacitated Facility Location Problem* diasumsikan bahwa jumlah fasilitas terbatas sehingga ada customer yang tidak terlayani, sedangkan pada *Uncapacitated Facility Location Problem* (UFLP) diasumsikan bahwa semua customer dapat terlayani oleh fasilitas [2].

Para peneliti sebelumnya telah berhasil mengaplikasikan beberapa metode untuk menyelesaikan *Uncapacitated Facility Location Problem* (UFLP) dengan berbagai algoritma. Algoritma yang pernah digunakan diantaranya adalah *Discrete Particle Swarm Optimization* [2], *Artificial Bee Colony Optimization* [3], *Simplified Binary Artificial Fish Swarm Algorithm* [4], dan *Ant Colony Optimization* [1]. Algoritma Pencarian *Cuckoo* atau *Cuckoo Search Algorithm* (CSA) adalah salah satu algoritma yang diperkenalkan oleh Xin-She Yang dan Suash Deb pada tahun 2009 [5]. CSA merupakan algoritma yang terinspirasi dari sifat parasite spesies *cuckoo* yang meletakkan telurnya di sarang burung inang. Beberapa burung inang dapat terlibat konflik langsung dengan *cuckoo* yang mengganggu. Misalnya jika seekor burung inang menemukan telur bukan milik mereka sendiri, ia akan membuang telur asing ini atau hanya meninggalkan sarangnya dan membangun sarang baru di tempat lain. Beberapa spesies *cuckoo* seperti *the New World brood-parasit Tapera* telah berevolusi sedemikian rupa sehingga warna dan pola telurnya meniru beberapa spesies inang yang terpilih [6].

Menurut Yang dan Deb [5], ada tiga aturan ideal dalam *Cuckoo Search Algorithm* (CSA). Pertama, setiap *cuckoo* meletakkan satu telur pada satu waktu, dan tempat peletakannya dilakukan dalam sarang yang terpilih secara acak. Kedua, sarang terbaik dengan telur yang berkualitas sebagai solusi yang akan digunakan untuk generasi selanjutnya. Ketiga, jumlah sarang burung lain yang tersedia adalah tetap, dan burung pemilik sarang dapat menemukan telur *cuckoo* dengan probabilitas (p_a) anggota bilangan interval terbuka 0 dan 1. Dalam kasus ini, burung pemilik sarang dapat membuang telur *cuckoo* atau meninggalkannya sehingga pemilik sarang dapat membangun sarang yang benar-benar baru dalam lokasi yang baru.

Menurut Yang dan Deb [5], *Cuckoo Search Algorithm* (CSA) ini menggunakan *Lévy flights* yang menyebabkan waktu komputasi yang lebih cepat jika dibandingkan dengan *Genetic Algorithm* (GA) dan *Particle Swarm Optimization* (PSO). Berdasarkan uraian yang telah dijelaskan sebelumnya, maka dibuatlah penerapan *Cuckoo Search Algorithm* (CSA) untuk menyelesaikan *Uncapacitated Facility Location Problem* (UFLP).

2 *Uncapacitated Facility Location Problem* (UFLP)

Dalam bentuk sederhana, masalah penempatan fasilitas pada suatu lokasi adalah suatu permasalahan untuk menentukan n lokasi yang akan dibangun n fasilitas yang akan melayani sejumlah m customer, selain itu terdapat sekumpulan n lokasi dimana akan dibangun n fasilitas dan untuk membangun satu fasilitas pada satu lokasi ke- i dibutuhkan biaya fc_j, customer ke- i dilayani fasilitas ke- j sedemikian hingga terdapat biaya c_{ij}. Variabel keputusan untuk membangun atau tidak membangun fasilitas ke- j dinotasikan y_j. Sedangkan variabel keputusan bagi masing-masing fasilitas ke- j untuk melayani atau tidak melayani customer ke- i dinotasikan x_{ij}. Jika diasumsikan bahwa semua customer dapat terlayani oleh fasilitas, maka masalah ini disebut dengan *Uncapacitated Facility Location Problem* (UFLP). Model matematika dari fungsi tujuan (Z) dirumuskan pada persamaan (1) dengan kendala persamaan (2) dan persamaan (3).

$$Z = \min\left(\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}\, x_{ij} \;+\; \sum_{j=1}^{n} fc_j\, y_j\right) \tag{1}$$

dengan kendala :

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall\, i = 1, \dots, m \tag{2}$$

$$0 \;\le\; x_{ij} \;\le\; y_j \;\;,\;\; i = 1,2, \dots, m;\; j = 1,2, \dots, n \tag{3}$$

$$x_{ij} = \begin{cases} 1, & jika\ customer\ i\ yang\ dilayani\ oleh\ fasilitas\ j \\ 0, & jika\ customer\ i\ yang\ tidak\ dilayani\ oleh\ fasilitas\ j \end{cases}$$

$$y_j = \begin{cases} 1, & jika\ fasilitas\ j\ dibangun \\ 0, & jika\ fasilitas\ j\ tidak\ dibangun \end{cases}$$

Pada model diatas, persamaan (1) menunjukkan bahwa biaya total untuk mendirikan fasilitas dan melayani *customer*. Persamaan (2) menunjukkan bahwa tiap *customer* ke-i hanya dilayani oleh satu fasilitas ke- j. Persamaan (3) menunjukkan bahwa *customer* ke-i hanya bisa dilayani oleh fasilitas ke- j jika fasilitas dibangun pada lokasi ke- j dan variabel keputusannya adalah variabel biner 0 dan 1 [2].

3 *Cuckoo Search Algorithm* (CSA)

Menurut Yang dan Deb [5], *Cuckoo Search Algorithm* (CSA) adalah salah satu dari *Nature-Inspired Algorithm* (Algoritma yang terinspirasi dari alam), yaitu terinspirasi dari sifat parasit beberapa spesies *cuckoo* yang meletakkan telurnya di sarang burung inang lainnya. Ada tiga pendekatan yang digunakan dalam *Cuckoo Search Algorithm* (CSA) dirumuskan sebagai berikut:

a. Setiap c*uckoo* meletakkan satu telur pada satu waktu, dan membuang telur dalam sarang yang dipilih secara acak.
b. Sarang terbaik dengan telur yang berkualitas sebagai solusi yang akan digunakan untuk generasi selanjutnya.

c. Jumlah sarang burung lain yang tersedia adalah tetap, dan burung pemilik sarang dapat menemukan telur *cuckoo* dengan probabilitas $p_a \in (0,1)$. Dalam kasus ini, burung pemilik sarang dapat membuang telur cuckoo atau meninggalkan sarang sehingga pemilik sarang dapat membangun sarang baru di lokasi yang baru.

Secara umum, cara paling efektif untuk seekor burung mencari makanannya adalah dengan menggunakan strategi *Random Walks*, karena langkah selanjutnya ditentukan berdasarkan lokasi saat ini dan peluang peralihan ke tempat selanjutnya. Menurut Lin [7], *Cuckoo Search* menggunakan dua random walks, yaitu *Lévy Flights Random Walks* (LFRW) dan *Biased Random Walks* (BRW).

Lévy Flights Random Walks (LFRW) merupakan random walk dengan stepsize berdasarkan distribusi *Lévy*. Berdasarkan implementasinya [8], *Lévy flights* merupakan pencarian solusi baru yang berada disekitar solusi terbaik sementara dan dilakukan berdasarkan persamaan (4).

$$x_i^{t+1} = x_i^t + \alpha\, S(x_{best}^t - x_i^t)\kappa \tag{4}$$

dengan x_i^t merupakan sarang ke i pada iterasi ke t, $\alpha > 0$ merupakan parameter stepsize, S merupakan *step length* yang mengandung *Lévy flights* dari algoritma Mantegna, κ merupakan bilangan real yang dibangkitan secara acak yang berdistribusi normal dengan rata-rata 0 dan simpangan baku 1, x_{best}^t merupakan sarang terbaik pada iterasi ke t.

Menurut Mantegna [9], *step length* (S) dapat dihitung dengan menggunakan

$$S = \frac{u\,\sigma}{|v|^{\frac{1}{\beta}}} \tag{5}$$

dengan β merupakan parameter, variabel u dan v bilangan real berdistribusi normal yang dibangkitkan secara acak dengan rata-rata 0 dan simpangan baku 1, dengan nilai σ ditentukan berdasarkan:

$$\sigma = \left(\frac{\Gamma(1+\beta)\sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right)\beta 2^{\frac{\beta-1}{2}}} \right)^{\frac{1}{\beta}} \tag{6}$$

dengan $\Gamma(x)$ merupakan fungsi gamma.

Menurut Diethelm [10], fungsi $\Gamma: (0,\infty) \to \mathbb{R}$, merupakan fungsi gamma yang didefinisikan sebagai :

$$\Gamma(z) = \int_0^\infty x^{z-1} e^{-x} dx$$

Pendekatan Lanczos merupakan metode untuk menghitung nilai-nilai fungsi gamma secara numerik yang ditemukan oleh Cornelius Lanczos pada tahun 1964. Misalkan $g \in \mathbb{R}$, dan n merupakan bilangan bulat yang di definisikan sebagai berikut :

$$n = \lfloor g \rfloor + 1$$

dengan $\lfloor g \rfloor$ merupakan fungsi *floor*.

Menurut Vrajitoru dan Knight [11], fungsi *floor* dari bilangan real x yang dinotasikan $\lfloor x \rfloor$ merupakan fungsi yang mengembalikan bilangan bulat ($\mathbb{Z}$) terbesar tidak lebih besar dari x. Fungsi $\lfloor x \rfloor$ dapat dinyatakan :

$$\lfloor x \rfloor = max\{m \in \mathbb{Z}, m \leq x\}$$

Menurut Lanczos [12], bentuk dasar pendekatan Lanczos didefinisikan sebagai :

$$\ln\big(\Gamma(z+1)\big) = \ln(Z\mathcal{P}) + (z + 0.5) - (z + g + 0.5) \quad (7)$$

dengan $\mathcal{Z}$ adalah matriks berukuran $1\ x\ n$, dan $\mathcal{P}$ adalah matriks berukuran $n\ x\ 1$ yang dibangun dari hasil perkalian matriks berukuran $n\ x\ n$ yaitu matriks $\mathcal{D}, \mathcal{B}, \mathcal{C}$ dan matriks $\mathcal{F}$ yang berdimensi $n\ x\ 1$.

Menurut Civicioglu dan Besdok [8]**,** *Biased Random Walks* (BRW) digunakan untuk menemukan solusi baru secara acak yang berada di posisi yang lebih jauh dari solusi terbaik. Cara yang digunakan yaitu membangun solusi baru dengan menggunakan solusi saat ini sebagai dasar dan dengan dua solusi lainnya yang dipilih secara acak, dan solusi baru didapatkan dari penjumlahan solusi baru dengan solusi baru saat ini. BRW dapat diformulasikan sebagai berikut :

$$x_i^{t+1} = \begin{cases} x_i^t + \mathcal{R}\big(x_{permut1}^t - x_{permut2}^t\big),\ jika\ rand_1 > p_a \\ x_i^t,\ jika\ rand_1 \leq p_a \end{cases} \quad (8)$$

dengan indeks acak permutasi j dan k merupakan sarang ke- j dan ke- k dari populasi sarang yang ada, rand_1, $\mathcal{R}$ merupakan bilangan real yang dibangkitkan secara acak pada interval (0,1) dan p_a merupakan probabilitas telur *cuckoo* yang ditemukan oleh burung pemilik sarang.

Langkah-langkah untuk menerapkan *Cuckoo Search Algorithm* (CSA) adalah sebagai berikut:

a. Mendefinisikan fungsi tujuan $f(x), x = (x_1, x_2, \dots, x_n)^T$.
b. Membangkitkan populasi sebanyak n sarang x_j $(j = 1,2, \dots, \text{n})$.
c. Mengevaluasi kualitas/hitung fungsi tujuan F_j.
d. Menentukan *cuckoo* secara acak / membangkitkan sebuah solusi baru dengan *Lévy Flights*, x_i.
e. Mengevaluasi kualitas/hitung fungsi tujuan F_i.
f. Melakukan seleksi dengan memilih diantara sarang secara acak. Jika $F_i < F_j$, maka mengganti sarang ke-j dengan solusi baru.

g. Melakukan pergantian sarang terburuk, jika $rand_1 > p_a$ maka *cuckoo* melakukan pergantian sarang baru dengan *Biased random walk.* Jika $rand_1 \leq p_a$ maka sarang tidak melakukan pergantian.
h. Urutkan sarang berdasarkan nilai fungsi tujuannya dan tentukan sarang terbaiknya.
i. Tentukan dan simpan sarang terbaiknya.

4 Penyelesaian *Uncapacitated Facility Location Problem* (UFLP) dengan Menggunakan *Cuckoo Search Algorithm* (CSA)

Langkah-langkah yang digunakan dalam menyelesaikan permasalahan *Uncapacitated Facility Location Problem* (UFLP) dengan menggunakan *Cuckoo Search Algorithm* (CSA) adalah sebagai berikut :

a Menginputkan data yang berkaitan dengan *Uncapacitated Facility Location Problem* (UFLP) yaitu banyaknya lokasi yang akan dibangun fasilitas (n), banyaknya customer (m), biaya untuk membangun fasilitas pada lokasi ke- j (fc_i), biaya untuk melayani customer ke-i pada fasilitas ke- j (c_{ij}).

b Menentukan parameter *Cuckoo Search Algorithm* yaitu banyak sarang (x), maksimal iterasi, stepsize (α), dan probabilitas pergantian sarang (p_a), nilai β.

c Membangkitkan posisi awal sarang
1. Membangkitkan bilangan real yang dibangkitkan secara acak sebanyak sarang pada interval (0,1) sejumlah n lokasi.
2. Menghitung *open facility* (y_j) untuk menentukan fasilitas yang akan dibangun dengan mempresentasikan bilangan real yang dibangkitkan secara acak ke bentuk biner 0 atau 1 dengan mengalikan 1000 dan modulo(2).

d. Menghitung total biaya membangun fasilitas dan pelayanan *customer* yang akan menjadi fungsi tujuan (F_j).

e. Menentukan x_{best} sementara dengan mengurutkan nilai fungsi tujuan dari terkecil sampai terbesar.

f. Memperbarui posisi sarang dengan Lévy Flights Random Walk (LFRW) dilakukan dengan langkah-langkah berikut:
1. Memilih secara acak sarang ke- i dari sarang (x_{lfrw}).
2. Menentukan nilai u dengan membangkitkan bilangan real secara acak berdistribusi normal $N(0,1)$ sejumlah lokasi.
3. Menentukan nilai v dengan membangkitkan bilangan real secara acak berdistribusi normal $N(0,1)$ sejumlah lokasi.
4. Menghitung nilai σ.
5. Menghitung *step length* (S).
6. Menentukan nilai κ dari membangkitkan bilangan real secara acak berdistribusi normal $N(0,1)$ sejumlah lokasi.
7. Menentukan nilai baru dengan mengalikan *step length* (S).

8. Menggantikan x_{lfrw} dengan x_{lfrw} baru.

g. Menghitung total biaya / fungsi tujuan dari hasil penentuan secara acak dengan *Lévy Flights Random Walk* (F_{lfrw}).

h. Pilih satu sarang j diantara n sarang secara acak, kemudian membandingkan nilai fungsi tujuan sarang j dengan sarang LFRW. Jika nilai total biaya $F_{lfrw} < F_j$, maka mengganti sarang ke- j dengan solusi baru. Jika nilai total biaya $F_{lfrw} \geq F_j$, maka sarang tidak melakukan pergantian.

i. Melakukan pergantian sarang terburuk dengan membangkitkan bilangan real (0,1) secara acak $rand_1$ sejumlah sarang kemudian membandingkan dengan nilai p_a. Jika $rand_1 \leq p_a$, maka sarang tidak melakukan pergantian. Jika nilai $rand_1 > p_a$, maka sarang terburuk terdeteksi dan kemudian melakukan pergantian sarang terburuk dengan *Biased Random Walk* (BRW). Pergantian sarang terburuk dengan *Biased Random Walk* (BRW) dilakukan dengan langkah-langkah sebagai berikut:
   1. Menentukan nilai $\mathcal{R}$ dengan membangkitkan bilangan real secara acak sejumlah lokasi.
   2. Membangkitkan indeks sarang $x_{permut1}$ dan $x_{permut2}$ dengan bilangan bulat secara acak.
   3. Menentukan nilai x_{BRW} dengan perkalian $rand_1$ dengan hasil pengurangan $x_{permut1}$ dan $x_{permut2}$.

j. Membandingkan hasil pergantian sarang terburuk (BRW) dengan sebelumnya. Jika total biaya / fungsi tujuan $F_{BRW} < F_j$ maka pergantian sarang dilakukan.

k. Menentukan x_{best} dengan mengurutkan nilai fungsi tujuan dari terkecil sampai terbesar.

l. Mengulangi proses e) sampai j) hingga iterasi maksimal.

5 Hasil dan Pembahasan

Data berukuran kecil yang digunakan terdiri dari 10 lokasi dan 15 *customer* [13] dengan parameter yang digunakan p_a = 0.25 [5], α = 0.01 [8], β = 1.5 [14]. Sedangkan nilai parameter yang berbeda pada jumlah sarang, *max iterasi,* dan probabilitas pergantian sarang (p_a) dilakukan sebanyak 10 kali *running* program untuk hasil yang terbaik. Berikut ini hasil *running* program dengan data kecil dapat disajikan pada Tabel 1.

Berdasarkan hasil *running* pada Tabel 1 dapat disimpulkan bahwa solusi terbaik pada data kecil adalah dengan total biaya sebesar 149690 dengan banyak jumlah sarang = 100, iterasi maksimum = 1000 dan nilai p_α = 0.25. Hasil tersebut didapatkan dari biaya pembangunan fasilitas pada lokasi 2, 4, 5, 6, 7.

Pola hasil *running* program pada data kecil menunjukkan bahwa semakin banyak jumlah iterasi dan sarang maka nilai fungsi tujuan semakin baik. Sedangkan semakin kecil nilai p_a, maka nilai fungsi tujuan juga semakin baik yaitu total biaya yang minimum. Rincian

biaya pembangunan fasilitas dan biaya pelayanan *customer* dari solusi terbaik data berukuran kecil yang diperoleh selama *running* program dapat dilihat pada Tabel 2.

Tabel 1. Hasil Running Program Data Kecil

Maksimum Iterasi	**Jumlah Sarang**	p_α		
		0.25	**0.50**	**0.75**
10	**10**	156361	158416	161479
	50	155474	155498	155498
	100	151559	156568	156361
100	**10**	153726	156361	156568
	50	155196	155196	155196
	100	150489	153476	153979
1000	**10**	151233	153726	155243
	50	150720	151559	153726
	100	149690	150187	151257

Tabel 2. Solusi Terbaik Penyelesaian Data Berukuran Kecil

Customer	**Fasilitas yang dibangun**	**Biaya**
1	7	4374.53
2	4	1838.96
3	5	9147.6
4	6	20071.7
5	4	1061.75
6	5	7483.61
7	2	28499.2
8	6	8861.74
9	4	1324.95
10	4	869.6
11	4	12638.5
12	5	1231.7
13	6	5185.98
14	5	1953.74
15	5	7310.81
Total Biaya Pelayanan		112190
Total biaya membangun fasilitas		37500
Total Biaya		149690

Berikut hasil penyelesaian data besar dengan banyak lokasi adalah 50 dengan *50 customer* [14], parameter yang digunakan digunakan p_a = 0.25 [5], α = 0.01 [8], β = 1.5 [15]. Hasil *running* program dengan data berukuran besar disajikan disajikan pada Tabel 3.

Tabel 3. Hasil Running Program Data Berukuran Besar

Maksimum Iterasi	**Jumlah Sarang**	p_α		
		0.25	**0.50**	**0.75**
10	**10**	851704	855926	860229
	50	850292	850323	851854
	100	846147	847200	849550
100	**10**	839807	853839	859992
	50	844711	846682	851156
	100	840291	843745	846094
1000	**10**	835907	851076	855760
	50	830280	833259	841292
	100	814321	819310	824492

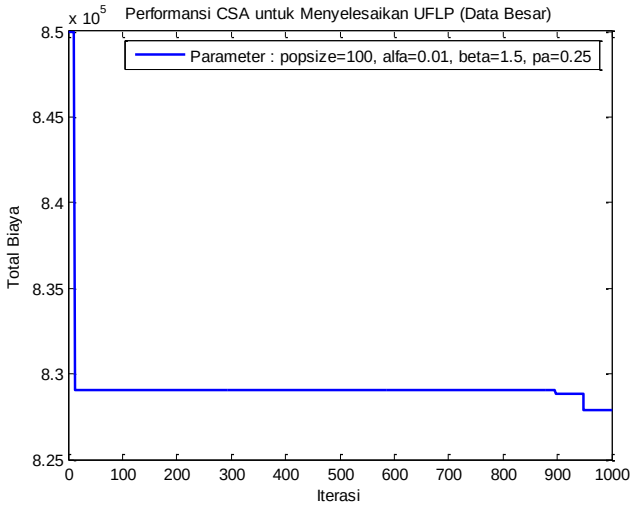
Berdasarkan Tabel 3, diperoleh Solusi terbaik yang didapatkan selama *running* program adalah dengan total biaya sebesar 817731,25 dengan banyak jumlah sarang = 100, iterasi maksimum = 1000 dan nilai p_α = 0.25. Hasil tersebut didapatkan dari biaya pembangunan fasilitas pada lokasi 4, 11, 13, 15, 18, 23, 25, 27, 28, 32, 34, 39, 45, 46.

Tabel 4. Solusi Terbaik Penyelesaian Data Berukuran Besar

Customer	**Fasilitas**	**Biaya**	**Customer**	**Fasilitas**	**Biaya**
1	23	5219.5	26	28	2801.31
2	25	1779.15	27	27	124051
3	26	7341.6	28	28	800,59
4	4	11548.3	29	32	4675.4
5	25	1015.25	30	32	14256
6	32	4947.15	31	32	3580.5
7	34	37594.1	32	32	1199.45
8	13	8861,73	33	32	12141.6
9	25	1275.45	34	34	130895
10	25	768.8	35	25	4891.25
11	11	12638,5	36	25	4154.1
12	39	2463.4	37	4	40426.9

Customer	Fasilitas	Biaya	Customer	Fasilitas	Biaya
13	13	5185.97	38	13	15076.1
14	28	2275.49	39	39	21387.9
15	28	1409,19	40	4	4104.1
16	25	12619.5	41	23	10086
17	11	2542,5	42	11	13627.4
18	18	11423,1	43	25	10071.9
19	11	3542	44	15	9331.25
20	45	4719	45	45	68042,4
21	11	693.5	46	46	21302.8
22	15	14919.4	47	46	8172.38
23	23	0	48	28	1568
24	32	5973.6	49	4	14036,1
25	25	5270.65	50	25	2614.05
			Total Biaya Pelayanan		709321
			Total biaya membangun fasilitas		105000
			Total Biaya		814321

Pola hasil *running* program pada data kecil menunjukkan bahwa semakin banyak jumlah iterasi dan sarang maka nilai fungsi tujuan semakin baik. Sedangkan semakin kecil nilai p_a, maka nilai fungsi tujuan juga semakin baik yaitu total biaya yang minimum. Rincian biaya pembangunan failitas dan biaya pelayanan *customer* dari solusi terbaik data berukuran besar yang diperoleh selama *running* program dapat disajikan pada Tabel 4.



Gambar 1. Performansi CSA

Gambar 1 menyajikan hasil *running* program pada data besar dengan parameter jumlah sarang = 100, α = 0.01, β = 1.5, dan p_a = 0.25. *Running* program dilakukan dengan jumlah maksimum iterasi =1000. Berdasarkan Gambar 1, pada jumlah iterasi kurang dari 50, total biaya turun tajam mendekati optimal namun menjadi konstan hingga pada iterasi mendekati 900. Selanjutnya terjadi penurunan total biaya pada iterasi ke 900. Nilai p_a yang kecil telah berhasil membuat proses pencarian solusi keluar dari jebakan optimum lokal sehingga pada iterasi lebih dari 900 total biaya kembali turun mendekati optimum.

6 Kesimpulan

Penerapan *Cuckoo Search Algorithm* (CSA) untuk menyelesaikan *Uncapacitated Facility Location Problem* (UFLP) menggunakan data berukuran kecil diperoleh solusi terbaik dengan total biaya 149690, sedangkan menggunakan data berukuran besar diperoleh solusi terbaik dengan total biaya yaitu 814321. Semakin kecil nilai nilai p_α, semakin besar jumlah sarang dan maksimum iterasi maka solusi yang diperoleh semakin baik dengan total biaya pembangunan fasilitas dan pelayanan customer lebih kecil.

7 Saran

Untuk penelitian selanjutnya, *Cuckoo Search Algorithm* (CSA) diharapkan mampu diterapkan ke dalam berbagai permasalahan optimasi lainnya, dilakukan juga penerapan dengan menggunakan pengembangan *Cuckoo Search Algorithm* (CSA), seperti *Discrete Cuckoo Search Algorithm* dan *Improve Cuckoo Search Algorithm*, serta dilakukan *hybrid* dengan algoritma *metaheuristic* lainnya yang memungkinkan untuk memperoleh hasil yang lebih baik untuk menyelesaikan *Uncapacitated Facility Location Problem*

8 Daftar Pustaka

[1] Kole, A., Chakrabarti, P., dan Bhattacharyya, S., 2014, *An Ant Colony Optimization Algorithm for Uncapacitated Facility Location Problem*. *Artificial Intellegence and Applications*, Vol 1, no.1, pp 51-61.

[2] Guner, A.R., dan Sevkli, M., 2008, *A Discrete Particle Swarm Optimization Algorithm for Uncapacitated Facility Location Problem*, Journal of Artificial Evolution and Applications, Vol. I, no. 1, pp 1-9.

[3] Tuncbilek, N., Tasgetiren. F., dan Esnaf, S., 2012, *Artificial Bee Colony Optimization Algorithm for Uncapacitated Facility Location Problems*, Journal of Economic and Social Research, Vol 14(1), 1-24.

[4] Azad, M.A.K., Rocha, A.M.A. C., dan Fernandes, E.M.G P., 2013, *A Simplified Binary Artificial Fish Swarm Algorithm for Uncapacitated Facility Location Problems*, World Congress on Engineering, Vol I, no. 1, 31-36.

[5] Yang, X.-S., dan Deb, S. 2009. *Cuckoo Search via Lévy Flights*. In Nature and Biologically Inspired Computing. *World Congress on NaBIC* (pp. 201-214). IEEE.

[6] Payne, R. B., Sorensen, M. D., dan Klitz,K., 2005, *The Cuckoos*, Oxford UniversityPress.
[7] Lin, Y., Zhang, C., dan Liang, Z., 2016, *Cuckoo Search Algorithm with Hybrid Factor Using Dimensional Distance*, Journal of Mathematical Problems in Engineering, Vol. 2016.
[8] Civicioglu, P., Besdok, E., 2013., Comparative Analysis of the Cuckoo Search Algorithm, In: Yang XS.(eds) Cuckoo Search and Firefly Algorithm. Studies in Computational Intelligence,Vol 516, 85-113.
[9] Mantegna, R.N, 1994, *Fast, Accurate Algorithm for Numerical Simulation of Levy Stable Stochastic Processes Phys. Rev E 49*, 4677-4683.
[10] Diethelm, K., 2004, *The Analysis of Fractional Differential Equation*, Springer, New York.
[11] Vrajitoru, D., dan Knight, W., 2014, *Practical Analysis of Algorithms*, Springer, New York.
[12] Lanczos, C., 1964, A Precision Approximation of The Gamma Function, *SIAM J Numer Anal*, 1: 86-89.
[13] Beasly,J.E.,2005a.OR-Library: http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files.cap71.txt, (17 Januari 2019)
[14] Beasly,J.E.,2005a.OR-Library: http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files.cap131.txt, (17 Januari 2019)
[15] Kaveh, A., dan Bakhspoori, T., 2013, *Optimum Design of Steel Frames Using Cuckoo Search Algorithm with Levy Flights*, Struct Des Tall Spec Build, 22: 2013-1036.