Artificial Bee Colony Algorithm untuk Menyelesaikan *Travelling Salesman Problem*

[1]Faisal Amri, [1]Erna Budhiarti Nababan, [1]Mohammad Fadly Syahputra

[1]Program Studi S1 Teknologi Informasi
Fakultas Ilmu Komputer dan Teknologi Informasi
Universitas Sumatera Utara
E-mail: faisal16.amri@gmail.com, ernabrn@usu.ac.id, nca.fadly@usu.ac.id

***Abstrak*— Permasalahan *Traveling Salesman Problem* (TSP) dikenal dengan sebagai permasalahan yang bersifat *Nondeterministic Polynomial-Hard* (*NP-Hard*). Penyelesaian eksak untuk masalah TSP ini mengharuskan perhitungan terhadap semua kemungkinan rute yang dapat diperoleh, kemudian memilih salah satu rute yang terpendek. Untuk itu jika terdapat n kota yang harus di kunjungin, maka diperlukanan proses pencarian sebanyak (*n*-1)!/2*n* rute. Pada penelitian ini, penulis menggunakan *Artificial Bee Colony* (ABC) *Algorithm*. ABC *algorithm* dengan *neighborhood operator* bertujuan untuk mendapatkan penyelesaian terbaik jalur terpendek dari TSP. Hasil komputasi menunjukkan ABC dengan *neighborhood operator* sudah mendapatkan penyelesaian terbaik untuk beberapa kasus.**

***Kata Kunci*—*Artificial Bee Colony, neighborhood operator, swap* dan *insert operator, Travelling Salesman Problem*.**

I. PENDAHULUAN

Traveling Salesman Problem (TSP) dikenal sebagai salah satu masalah optimasi yang banyak menarik perhatian para ahli matematika dan khususnya ilmuwan komputer karena TSP mudah didefinisikan dan begitu sulit untuk diselesaikan. Masalah TSP dinyatakan dapat dinyatakan dimana seseorang ingin mengunjungi ke sejumlah kota, dimana rangkaian kota-kota yang dikunjungi harus membentuk suatu jalur sedemikian rupa sehingga kota-kota tersebut hanya boleh dilewati tepat satu kali dan kemudian kembali lagi ke kota awal. Tujuan dari masalah TSP ini adalah untuk mencari rute atau jarak terpendek.

Penyelesaian eksak untuk masalah TSP ini mengharuskan perhitungan terhadap semua kemungkinan rute yang dapat diperoleh, kemudian memilih salah satu rute yang terpendek. Untuk itu jika terdapat n kota yang harus di kunjungin, maka diperlukanan proses pencarian sebanyak (*n*-1)!/2*n* rute [1]. Dengan cara ini komputasi yang harus dilakukan akan meningkat seiring bertambahnya jumlah kota yang harus dilalui. Permasalahan ini dikenal dengan sebagai permasalahan yang bersifat *Nondeterministic Polynomial-Hard* (*NP-Hard*). Hal inilah yang menyebabkan penyelesaian secara eksak sulit dilakukan [2]. Berdasarkan hal tersebut, banyak peneliti lebih memusatkan kepada pengembangan metode-metode pendekatan (*heuristic*) seperti *Ant Colony system* (ACS) [3], *Genetic Algorithm* (GA) [1], *Tabu Search* (TS) [4], *Artificial Bee Colony Algorithm* (ABC *Algorithm*) [5], *Bee Colony Optimization* (BCO) [6], dan lain sebagainya.

Dalam tulisan ini, penulis memilih salah satu algoritma di atas untuk menyelesaikan masalah TSP yaitu dengan menggunakan *Artificial Bee Colony Algorithm* (*ABC Algorithm*), *swarm intelligence* baru berbasis optimasi, untuk mencari jalur terpendek dan waktu tercepat. Penulis memilih menggunakan algoritma ABC karena sederhana, fleksibel [7], dan memiliki kemampuan untuk keluar dari *local minimum* dan dapat secara efisiensi digunakan untuk multimodal dan multivariable optimasi fungsi [8].

ABC *Algorithm* ini telah banyak digunakan beberapa diantaranya pada *Fuzzy Clustering* [9], *Flexible Job Shop Scheduling Problem* [10], *Numerical Function* [8], *Job Shop Scheduling* [11].

II. IDENTIFIKASI MASALAH

TSP dikenal sebagai permasalahan yang bersifat *Nondeterministic Polynomial-Hard* (*NP-Hard*). Hal inilah yang menyebabkan penyelesaian secara eksak sulit dilakukan. Permasalahan dari penulisan ini adalah bagaimana mendapatkan penyelesaian terbaik dalam hal ini jalur terpendek dari TSP [2].

Adapun *objective function* untuk jarak terpendek pada penelitian ini yaitu:

$$T_c = \sum_{k=1}^{d} C_{ij} \quad (1)$$

Dimana *Tc* adalah *objective function*, C_{ij} adalah jarak dari kota-i ke kota-j. *i* adalah kota awal, dan *j* adalah kota tujuan.

Dengan *Hard Constraint* [12] sebagai berikut:

- Kota-kota yang ada hanya dikunjungi sekali.

$$X_{ij} + X_{ji} \leq 1 \qquad for\ all\ i,j \quad (2)$$

- Jarak dari kota i ke kota j = jarak dari kota j ke kota i.

$$d_{ij} = d_{ji} \quad (3)$$

- *Non negative constraint.*

$$X_{ij} \geq 0 \quad (4)$$

- Semua kota harus dikunjung.

$$\sum_i^n \sum_j^n X_{ij} = 1 \quad for\ i\ and\ j \tag{5}$$

Adapun *Soft Constrain* pada penelitian ini adalah: jarak terpendek yang didapatkan

1. Artificial Bee Colony Algorithm

Bonabeau et al. [13] mendefinisikan *swarm intelligence* sebagai suatu percobaan untuk mendesain algoritma atau mendistibusikan alat pemecahan masalah yang terinsirasi oleh tingkah laku kolektif dari koloni sosial serangga dan perkumpulan hewan lainnya. Istilah *swarm* secara *general* merujuk pada suatu koleksi pengendalian dari agen yang saling berinteraksi maupun individual. Salah satu contoh Algoritma *swarm intelligence* adalah *Artificial Bee Colony* (ABC) *Algorithm*.

Artificial Bee Colony (ABC) *Algorithm* adalah pendekatan *population-based metaheuristic* yang diusulkan oleh Karaboga dan Basturk [8]. Pendekatan ini terinsiprasi oleh perilaku cerdas kawanan lebah madu mencari makanan. Pada model ABC *Algorithm* memiliki 3 kelompok lebah, yaitu: *employed bees*, *onlookers*, dan *scouts* di dalam *artificial bee colony*.

Employed bee yang berhubungan dengan sumber makanan tertentu, *onlooker bee* menyaksikan tarian lebah yang digunakan dalam sarang untuk memilih sumber makanan, dan *scout bee* mencari sumber makanan secara acak. *Onlooker* dan *scout bee* merupakan *unemployed bee*. Awalnya *scout bee* menemukan posisi semua sumber makanan, setelah itu tugas dari *employed bee* dimulai. Sebuah *employed bee* buatan secara probabilitas memperoleh beberapa modifikasi pada posisi dalam memori untuk menargetkan sumber makanan baru dan menemukan jumlah nektar atau nilai *fitness* dari sumber baru. Kemudian, *scout bee* mengevaluasi informasi yang diambil dari semua *employed bee* buatan dan memilih sumber makanan akhir dengan nilai probabilitas tertinggi terkait dengan jumlah nektar tersebut. Jika nilai *fitness* yang baru lebih tinggi dari yang sebelumnya, lebah itu akan melupakan yang lama dan menghafal posisi baru. Hal ini disebut sebagai *greedy selection*. Kemudian *employed bee* yang sumber makanan telah habis menjadi *scout bee* untuk mencari sumber makanan lebih lanjut sekali lagi.

Menurut Bhagade and Puranik [14] dalam ABC, solusi yang merupakan sumber makanan dan kuantitas madu dari sumber makanan sesuai dengan kebugaran dari solusi terkait. Jumlah *employed* dan *scout bee* adalah sama, dan jumlah ini sama dengan jumlah sumber makanan. *Employed bee* yang solusinya tidak dapat ditingkatkan melalui sejumlah percobaan, yang ditentukan oleh pemakai dari ABC disebut *limit*, kemudian *employed bee* tersebut menjadi *scout bee* dan solusi yang dihasilkan diabaikan/ditinggalkan.

Bhagade dan Puranik [14] menjelaskan dalam algoritma ini, *employed bee* menghasilkan modifikasi dalam posisi (yaitu solusi) dalam memori dan memeriksa jumlah nektar (*fitness*) dari sumber (solusi). *Employed bee* kemudian mengevaluasi informasi nektar ini dan kemudian memilih sumber makanan dengan probabilitas yang berkaitan dengan nilai *fitness*-nya.

Ada 3 tahapan utama pada *basic* ABC. Tahapan pertama, menghasilkan inisial solusi dari sumber makanan secara acak. Untuk memperbarui solusi yang mungkin, setiap *employed bee* memilih calon posisi sumber makanan baru, yang mana posisi tersebut berbeda dengan sebelumnya. Posisi baru sumber makanan dihitung dengan persamaan berikut ini:

$$x_{ij} = \theta_{ij} + \emptyset\left(\theta_{ij} - \theta_{kj}\right) \tag{6}$$

Dimana:
x_i : calon solusi dari θ_i.
θ_i : posisi *employed bee* ke-*i*
θ_k : Tetangga (*Neighbor*) e*mployed bee* dari θ_i.
$\emptyset$: Bilangan acak antara [-1,1]
$i \in \{1, 2, 3, \dots, n\}$
$k \in \{1, 2, 3, \dots, n\}$
Dimana nilai dari $i \neq k$
n : jumlah *employed bee*
$j \in \{1, 2, 3, \dots, D\}$
D : Dimensi penyelesaian.

Pada tahapan kedua, setiap *onlooker bee* memilih salah satu sumber makanan yang diperoleh dari *employed bee*. Probabilitas sumber makanan akan dipilih dapat diperoleh dari persamaan di bawah ini:

$$P_{ij} = \frac{F(\theta_i)}{\sum_{k=1}^{S} F(\theta_k)} \tag{7}$$

Dimana :
Pij : kemungkinan memilih *employed bee* ke-*i*
$F(\theta)_I$: *Fitness Value* dari *employed bee* ke-*i*
S : jumlah *employed bee*
θ_i : posisi dari *employed bee*

Setelah memilih sumber makanan, *onlooker bee* pergi ke sumber makanan yang dipilih dan memilih sumber calon makanan baru. Selanjutnya pada tahapan terakhir, *limit* adalah batasan yang telah ditetapkan dalam siklus ABC *Algorithm* dan mengendalikan banyaknya solusi tertentu yang tidak diperbarui. Setiap sumber makanan yang tidak meningkat melewati *limit* akan ditinggalkan dan diganti dengan posisi baru dan *employed bee* menjadi *scout bee*. Posisi acak yang baru dipilih oleh *scout bee* akan dihitung melalui persamaan di bawah ini:

$$\theta_{ij} = \theta_{j\,min} + rand.\left(\theta_{j\,max} - \theta_{j\,min}\right) \tag{8}$$

Dimana :
rand: bilangan acak antara [0,1]
$\theta_{j\,max}$: batas atas dari sumber posisi didalam dimensi *j*
$\theta_{i\,min}$: batas bawah dari sumber posisi didalam dimensi *j*

2. Algoritma Euclidean

Algoritma Euclidean merupakan salah satu metode pengukur jarak dalam arti yang sebenarnya [15][16]. kebanyakan kasus yang berhubungan dengan penghitungan jarak maka sering merujuk pada algoritma euclidean. Jarak euclidean dihitung berdasarkan akar kuadrat dari sepasang benda, rumusnya adalah sebagai berikut :

$$d_{ij} = \sqrt{\sum_{k-1}^{n}\left(x_{ik} - x_{jk}\right)^2} \tag{9}$$

Dimana :
dij = jarak euclidean antara *i* dan *j*
n = banyaknya jarak ke-*n*
x_{ik} = jarak x dari *i* ke *k*
x_{jk} = jarak x dari *j* ke *k*

III. PENELITIAN TERDAHULU

Permasalahan TSP sudah banyak diteliti oleh para ilmuwan. Puspitorini [2] telah meneliti masalah TSP dengan menggunakan teknik jaringan syaraf kohonen dengan *objective function* adalah jarak terpendek. Hasilnya, untuk mendapatkan rute perjalanan terpendek sangat dipengaruhi oleh parameter pelatihan. Sehingga jika proses pelatihan dilakukan beberapa kali dengan data koordinat kota yang sama akan memungkinkan mendapatkan jalur dan panjang jalur perjalanan yang berbeda. Dan untuk jumlah kota yang besar dibutuhkan memori yang cukup besar.

Wong et al [6] telah meneliti masalah TSP dengan menggunakan Algoritma *Bee Colony Optimization* (BCO), dengan *objective function* adalah jarak terpendek dan waktu komputasi tercepat. Untuk *initial solution*, mereka menggunakan teknik *transition heuristic dan nearest neighbourhood heuristic*. Dan untuk *improvement solution*, teknik *2-opt, Frequency-based Prunning Strategy* (FBPS) dan *Fixed-Radius Near Neigbourhood* (FRNN). Wong et al melakukan perbandingan antara BCO + 2-*opt*, BCO + FRNN 2-*opt*, dan BCO + FRNN 2-*opt* + FBPS. Hasilnya untuk *objective function* adalah jarak, ketiga metode berhasil mencapai jarak terpendek 19 dari 20 data yang diberikan, sedangkan untuk waktu komputasi, FRNN 2-*opt* + FPBS mendapatkan waktu terbaik dari 3 teknik yang digunakan.

Marikanis dan Pardolos [17] telah meneliti masalah TSP dengan menggunakan teknik *Expanding Neighborhood Greedy Randomized Adaptive Search Procedure* (GRASP) dengan *objective function* adalah jarak terpendek. Untuk *initial solution*, mereka menggunakan *a randomized greedy function* dan untuk *improvement solution* menggunakan *local search (kruskal, 2-opt,* dan *3-opt).* Hasilnya 45% dari data yang digunakan berhasil mendapatkan jarak terpendek.

Li et al [5] telah meneliti masalah TSP dengan menggunakan teknik *Discrete Artificial Bee Colony* (DABC) dengan *objective function* adalah jarak terpendek. Untuk *initial solution* yang digunakan, mereka menggunakan teknik *random* dan *improvement solution* menggunakan teknik ABC, *swap operator dan swap sequence*. Untuk pengujian, DABC akan dibandingkan dengan teknik *Particle Swarm Intelligence* (PSO). Hasilnya, dengan parameter yang sama, DABC mendapatkan hasil yang lebih baik dari PSO. Hanya saja hasil yang didapatkan tidak mencapai jarak terpendek yang telah diketahui.

Pathak dan Tiwari [18] telah meneliti masalah TSP dengan menggunakan teknik *Artificial Bee Colony* (ABC), dengan o*bjective function* adalah jarak terpendek. Untuk *initial solution* yang digunakan, mereka memakai teknik *random* dan *improvement solution* menggunakan teknik ABC dan *Shortest Path Value* (SPV). Untuk pengujian, ABC akan dibandingkan dengan *Genetic Algorithm* (GA) dengan data uji 30 dan 60 kota. Hasilnya ABC dengan SPV mendapatkan hasil yang lebih baik dibandingkan dengan GA.

Berikut ini merupakan ringkasan penelitian terdahulu yang dirangkum pada tabel 1.

Tabel 1. Penelitian Sebelumnya

Nama/Tahun	Metode yang digunakan	Teknik yang digunakan	
		Initial Solution	*Improvement* Solution
Wong, L.P., Low, M.Y.H., Chong, C.S. /2009	*Bee Colony Optimisation*	*transition heuristic, Nearest Neighbourhood heuristic*	*BCO, 2-opt, Frequency-based Prunning Strategy* (FBPS) *and Fixed-Radius Near Neigbourhood* (FRNN)
Puspitorini, S. /2008	Jaringan Saraf Kohonen	-	-
Marikanis, Y. dan Pardalos, P.M. /2005	*Expanding Neighborhood Greedy Randomized Adaptive Search Procedure*	*a randomized greedy function*	*Local search*
Li, L., Cheng, Y., Tan, L., and Niu, B./ 2012	*Discrete Artificial Bee Colony*	*Random Method*	ABC, *Swap Operator, Swap Sequence (Random Method)*
Pathak, N., Tiwari, S.P. /2012	*Artificial Bee Colony*	*Random Method*	ABC, *Shortest Path Value*

Berdasarkan hasil penelitian terdahulu, penulis mengusulkan penelitian menggunakan:

- Algoritma: Modifikasi dari *Artificial Bee Colony Algorithm* (*Discrete Artificial Bee Colony*)
- *Initial Solution*: *Random Method*
- *Improvement Solution*: ABC *Algorithm* dan *neighbor operator* (*swap operator, swap sequence, insert operator*, dan *insert sequence*).

Perbedaan dengan penelitian terdahulu, disini penulis tidak hanya menggunakan *swap operator* dan *swap sequene* pada improvement solution, tetapi juga menambahkan *insert operator* dan *insert sequence*, karena kedua operator ini merupakan *neighborhood* operator jenis *point-to-point*.

IV. METODE PENELITIAN

Permasalahan *Travelling Salesman Problem* (TSP) akan diselesaikan dengan menggunakan *Artificial Bee Colony* (ABC) *Algorithm*. Penyelesaian dengan menggunakan algoritma tersebut bertujuan untuk mendapatkan jarak tempuh yang minimal dari sebuah *trail* tertutup.

Data file .tsp yang digunakan sebagai data uji adalah data TSP Simetri yang hanya mendukung tipe EDGE_WEIGHT_TYPE: EUC_2D, yaitu kordinat posisi dengan format Euclidian 2 dimensi. Adapun data tersebut adalah sebagai berikut:

a. Berlin52.
b. Eil51.
c. Eil76.

d. Eil101.
e. KroA100.
f. Pr76.
g. St70.

Pseudo-code merupakan deskripsi tingkat tinggi informal dan ringkasan algoritma yang menggunakan konvensi struktural suatu bahasa pemrograman yang ditujukan untuk dibaca oleh manusia bukan oleh mesin dengan tujuan untuk mempermudah pemahamanan terhadap algoritma. *Pseudo-code* dari ABC *Algorithm* yang diusulkan untuk menyelesaikan masalah TSP dapat dilihat pada gambar 1.

Step 1 - Inisialisasi semua parameter yang diperlukan, yaitu *colony size*, maksiterasi, *limit*.
Step 2 - Fase *initial*
- Untuk tiap *population size*, *initial solution* dihasilkan secara *random*.
- Hitung *fitness* menggunakan (9).

Step 3 - Iterasi = 1
Step 4 - Fase *Employed bee*
- Untuk tiap *employed bee*, *update employed bee* dengan menggunakan *neighborhood operator, swap operator* dan *swap sequence*.
- Hitung *fitness* setiap *employed bee* menggunakan (9).
- Jika solusi yang baru hasilnya lebih baik dari sebelumnya gantikan solusi lama dengan solusi baru, jika tidak tambahkan *limit* dengan 1.

Step 5 - Hitung probabilitas tiap *employed bee*.
Step 6 - Fase *onlooker bee*
- Untuk tiap *onlooker bee*, pilih solusi dari *employed bee* dengan menggunakan (7) dan gunakan teknik *roulette wheel selection*.
- Tentukan solusi yang baru *employed bee* terpilih dengan menggunakan *neighborhood operator, insert operator dan insert sequence*.
- Hitung fitness setiap *onlooker bee* menggunakan (9).
- Jika solusi yang baru hasilnya lebih baik dari sebelumnya gantikan solusi lama dengan solusi baru, jika tidak tambahkan limit dengan 1.

Step 7 - Fase *scout bee*
- Hitung jumlah *trail* dan simpan jumlah makasimal, M, untuk tiap *bee* yang tidak mengalami peningkatan solusi.
- Jika M > *limit*, tinggalkan solusi yang tidak mengalami peningkatan, jika M < *limit, scout bee* memilih solusi sebelumnya

Step 8 - Hapalkan solusi terbaik dicapai saat ini
Step 9 - Iterasi = Iterasi + 1.
Step 10 - Sampai Iterasi = Maksiterasi.

Gambar 1. *Pseudocode* ABC-TSP

1. Input Parameter

Tahapan yang pertama dilakukan, yaitu menginputkan parameter-parameter yang dibutuhkan dalam penyelesaian permasalahan TSP dan menginputkan nama *file* dataset .tsp yang akan digunakan. Parameter-parameter yang dimaksud meliputi :

a. *File* TSP, yang memuat dataset TSP untuk selanjutnya diolah di dalam sistem. *File* tsp ini memiliki 3 informasi yang digunakan dalam pengolahan sistem, yaitu banyaknya *size problem* (kota), titik kordinat tiap kota. Untuk tiap titik kordinat dapat dihasilkan jarak tiap titik untuk mempermudahkan pengolahan data.
b. *Colony size*, merupakan jumlah dari *Employed bee* ditambah *Onlooker bee* yang akan digunakan dalam system. Dimana jumlah *Employed bee = Onlooker bee* dalam hal ini disebut *population size*.
c. *Limit*, merupakan batasan dari *population size* yang tidak mengalami peningkatan kualitas untuk sejumlah iterasi.
d. Maksiterasi, merupakan banyaknya iterasi yang akan dilakukan. Dalam hal ini disebut juga kriteria berhenti.

2. Fase *Initial*

Fase *initial* merupakan proses untuk mendapatkan *initial solution* untuk tiap *employed bee*. Dimana tiap bee merepresentasikan kandidat perjalanan (*tour*), dan dimana elemen pertama dan terakhir merepresentasikan kota asal.

Untuk menghasilkan *initial solution* dilakukan secara *random* [1]. Sebagai contoh untuk menghasilkan *bee* 1 (2, 4, 6, 3, 5, 8, 7, 9, 1, 10, 2), pembangkit bilangan acak digunakan untuk menghasilkan bilangan antara 1-10, dan sekali bilangan dihasilkan tidak diperbolehkan muncul lagi. Bilangan pertama yang dihasilkan akan dimunculkan pada akhir *tour* untuk memberikan *tour* tertutup. Kemudian pembangkit bilangan acak menghasilkan bilangan lain sampai semua bilangan dihasilkan dan membentuk sebauh *tour* penuh dan tertutup.

Setelah setiap *employed bee* menghasilkan *initial solution*, proses berikutnya adalah menghitung *fitness* dari tiap *employed bee* dengan menggunakan (9). Kemudian dideklarasikan juga *limit* dan status awal dengan nilai 0.

Initial solution yang terbentuk pada fase *initial* selanjutnya akan diperbaiki pada *improvement solution*. Pada *improvement solution* terdiri dari fase *employed bee* dan fase *onlooker bee*.

3. *Improvement Solution*

Improvement solution berada pada fase *employed bee* dan *onlooker bee*. *Improvement solution* digunakan untuk meng-*update* solusi dengan metode *neighborhood operator*. *Neighborhood operator* yang akan digunakan yaitu *Swap Operator* dan *Swap Sequence* pada fase *employed bee* dan *Insertion Operator* dan *Insertion Sequence* pada fase *onlooker bee*.

Fase *Employed bee*

Sebelumnya telah dijelaskan pada fase ini akan digunakan *neighborhood operator*, *Swap Operator* dan *Swap Sequence*. Berikut penjelasan mengenai *Swap operator* dan *Swap Sequence*.

- *Swap Operator* (SO)

Swap Operator digunakan untuk menentukan 2 kota dari solusi yang akan di tukar, dimana kota1 dan kota2 yang ditukar menggunakan teknik *random* dan bilangan yang dihasilkan untuk kota1 ≠ kota2. Sebagai contoh bilangan *random* yang dihasilkan adalah SO(9, 4). Maka solusi baru yang dihasilkan dari *initial solution* dengan SO adalah sebagai berikut:

$$Bee_i = Bee_i + SO \quad (10)$$

Dimana: Bee_i = individu *bee* ke-*i*
SO = *Swap operator* (kota1, kota2)
Kota 1 ≠ Kota 2

Sebagai contoh untuk bee-1, secara *random* nilai dari *swap operator* adalah SO(4, 9), maka proses *swap* yang terjadi dapat dilihat pada penjelasan berikut ini.

.

Bee-1 = (7, 0, 10, 8, 9, 1, 13, 11, 5, 6, 12, 2, 3, 4) + SO(4, 9)

7	0	10	8	9	1	13	11	5	6	12	2	3	4
7	0	10	6	9	1	13	11	5	8	12	2	3	4

- *Swap Sequence*(SS)

Swap sequence merupakan kumpulan dari *swap operator*, solusi baru untuk tiap *employed bee* dihasilkan berdasarkan *tour* sebelumnya dan *swap sequence*.

$$\begin{aligned} Bee_i &= Bee_i + SS \\ &= Bee_i + (SO_1, SO_2, SO_3, \ldots, SO_n) \\ &= (Bee_i + SO_1) + SO_2) + SO_3 +) \ldots +) SO_n) \end{aligned} \quad (11)$$

Dimana: *SS* = *Swap sequence*
SO_n = *Swap operator* ke-*n*
n = banyaknya *swap sequence*

Setelah kriteria berhenti dari proses *employed bee* telah terpenuhi, maka dilakukan perhitungan probabilitas dengan menggunakan (7). untuk tiap *employed bee*. Perhitungan ini akan digunakan pada fase berikutnya.

Fase *Onlooker Bee*

Sebelumnya telah dijelaskan pada fase ini akan digunakan *neighborhood operator*, *Insert operator* dan *Insert Sequence*. Berikut penjelasan mengenai *Insert operator* dan *Insert Sequence*.

- *Insert Operator* (IO)

Insert Operator digunakan untuk menentukan sebuah kota dari solusi yang akan di tukar posisinya, dimana kota awal dan posisi tujuan yang ditukar menggunakan teknik *random* dimana bilangan *random* yang dihasilkan untuk kota awal ≠ tujuan, dan kemudian sisa urutan solusi akan digeser. Sebagai contoh bilangan *random* yang dihasilkan adalah IO(9, 4). Maka solusi baru yang dihasilkan dari *initial tour* dengan IO adalah sebagai berikut:

$$Bee_i = Bee_i + IO \quad (12)$$

Dimana: Bee_i = individu *bee* ke-*i*
IO = *Insert Operator* (kota awal, tujuan)
Kota awal ≠ tujuan

Sebagai contoh untuk *bee*-1, secara *random* nilai dari *insert operator* adalah IO(4, 9), maka proses *insert* yang terjadi dapat dilihat pada penjelasan berikut ini.

.

Bee-1 = (7, 0, 10, 8, 9, 1, 13, 11, 5, 6, 12, 2, 3, 4) + *IO*(4, 9)

7	0	10	8	9	1	13	11	5	6	12	2	3	4
7	0	10	6	8	9	1	13	11	5	12	2	3	4

- *Insert Sequence*(IS)

Insert sequence merupakan kumpulan dari *insert operator*, solusi baru untuk tiap *employed bee* dihasilkan berdasarkan *tour* sebelumnya dan *insert sequence*.

$$\begin{aligned} Bee_i &= Bee_i + IS \\ &= Bee_i + (IO_1, IO_2, IO_3, \ldots, IO_n) \\ &= (Bee_i + IO_1) + IO_2) + IO_3 +) \ldots +) IO_n) \end{aligned} \quad (13)$$

Dimana: *IS* = *Insert Sequence*
IO_n = *Insert Operator* ke-*n*
n = banyaknya *insert sequence*

Setelah kriteria berhenti dari proses *onlooker bee* telah terpenuhi, selanjutnya memasuki fase *scout bee*.

Fase *Scout bee*

Setelah melalui dua fase *improvement solution*, fase *employed bee* dan fase *onlooker bee*, maka akan dilakukan perhitungan kualitas dari masing-masing *employed bee*. Jumlah *scout bee* disini bersifat dinamis, tergantung pada jumlah *employed bee* yang telah melebihi *limit*. Apabila *limit* dari *bee* yang melakukan *improvement solution* melebihi *maximum limit* yang ditetapkan, maka solusi dari *bee* tersebut akan dihilangkan dan diganti dengan solusi baru dengan menggunakan teknik *random*, meng-*update* jarak yang dihasilkan, dan me-reset *limit* kembali menjadi 0.

4. Kriteria Berhenti

Kriteria berhenti dari sistem ini akan dilakukan sebanyak maksiterasi yang didefinisikan. Iterasi akan dilakukan sampai kriteria berhenti terpenuhi, dan selama belum terpenuhi, maka akan mengulang langkah 3.

V. HASIL DAN DISKUSI

Pada penelitian ini, data yang diuji merupakan data benchmark yang bersumber dari TSPLIB [19].

Pengujian tersebut menggunakan data parameter-parameter tetap yakni, *Colony Size* (*CS*) = 100; *Limit* = CS * *Size Problem* * 100; *Number of sequence* (*Nse*) = 2**Size Problem* [5]*;* maksiterasi = 100.000. Masing-masing data akan diuji sebanyak 30 kali.

Tabel 2. Perbandingan Hasil ABC dengan *benchmark*

Kasus	*Best* ABC	RE(%)	*Benchmark*
Berlin52	7544,366	0,031	7542
Eil51	429,983	0,935	426
Eil76	545,387	1,373	538
Eil101	675,721	7,429	629
KroA100	23722,961	11,47	21282
Pr76	112477,091	3,992	108159
St70	699,93	3.693	675

Berdasarkan hasil pada tabel 2, terlihat semakin besar *size problem* yang diuji, tingkat kesalahan hasil yang diperoleh juga semakin besar. Dataset dengan ukuran terbesar, 100 *size problem*, menghasilkan tingkat kesalahan terbesar yaitu 11,47%. Dan semakain kecil *size problem* yang di uji, tingkat kesalahan yang diperoleh juga semakain kecil. Dataset yang kecil, 52 *size problem*, dengan tingkat kesalahan 0,031%. Tingkat kesalahan pada data berlin52, eil51, dan eil76 merupakan kesalahan karena adanya perbedaan perhitungan hasil yang didapatkan dengan menggunakan ABC *algorithm* dengan hasil *benchmark*. Pada hasil benchmark merupakan jarak sebuah *walk graph*, dimana kota awal ≠ kota tujuan, sedangkan hasil ABC merupakan jarak sebuah *cycle graph*, dimana kota awal = kota tujuan.

VI. KESIMPULAN

ABC dengan metode *improvement solution* dan *neighborhood operator*, *swap operator (swap sequence)* dan *insert operator (insert sequenec)* dapat digunakan dalam menyelesaikan permasalahan TSP. Dari pengujian yang telah dilakukan dapat disimpulkan beberapa hal yakni:

1. 3 dari 7 data *benchmark* mencapai nilai optimal dalam hal ini jarak terpendek.
2. Secara garis besar, semakin besar *size problem*, dalam hal ini jumlah kota yang diproses, tingkat kesalahan juga semakin meningkat.

DAFTAR PUSTAKA

[1] Otri, S. 2011. Improving The Bees Algorithm For Complex Optimisation Problems, Manufacturing Engineering Centre, Cardiff University, United Kingdom.

[2] Puspitorini, S. 2008. Penyelesaian Masalah Traveling Salesman Problem Dengan Jaringan Saraf Self Organizing. Media Informatika. Vol. 6(1): 39-55.

[3] Stutzle, T. 1997. MAX-MIN Ant System for the quadratic assignment problem. Technical Report, AIDA-97-4, FB Informatik, TU Darmstadt, Germany.

[4] Freisleben, B. and Merz, P. 1996. A genetic local search algorithm for solving symmetric and asymmetric travelling salesman problems. Proceedings of International Conference on Evolutionary Computation. Indianapolis, USA. hal.616-621.

[5] Li, L., Cheng, Y., Tan, L., and Niu, B. 2012. *Discrete Artificial Bee Colony Optimization for TSP Problem*. Lecture Notes in Computer Science. Vol: 6840. Hal: 566-573.

[6] Wong, L.P., Low, M.Y.H., and Chong, C.S. 2009. An efficient Bee Colony Optimization algorithm for Traveling Salesman Problem sing frequency-based pruning. IEEE International Conference on. Hal: 775 – 782.

[7] Karaboga, D. 2005. *"An idea based on Honey Bee Swarm for Numerical Optimization"*. Technical Report TR-06. Kayseri/Türkiye: Erciyes University, Engineering Faculty Computer Engineering Department.

[8] Karaboga, D. Basturk, B. 2007. A Powerful And Efficient Algorithm For Numerical Function Optimization Artificial Bee Colony (Abc) Algorithm. *J Glob Optim* Vol 39: hal. 459–471.

[9] Karaboga, D. Ozturk, C. 2010. Fuzzy Clustering with Artificial Bee Colony Algorithm. Scientific Research and Essays Vol. 5(14): hal. 1899-1902.

[10] Li, J.Q., Xie, S.X., Pan, Q.K., Wang, S. 2011. A Hybrid Artificial Bee Colony Algorithm for Flexible Job Shop Scheduling Problem. Int. J. of Computers, Communications & Control, Vol. 6(2), hal. 286-296.

[11] Zhang, R. dan Wu, C. 2011. An Artificial Bee Colony Algorithm for the Job Shop Scheduling Problem with *Random* Processing Times. Entropy. Vol: 13, hal: 1708-1729.

[12] Fereidouni, S. 2011. Solving Traveling Salesman Problem By Using A Fuzzy Multi-Objective Linear Programming. African Journal of Mathematics and Computer Science Research. Vol: 4(11), hal: 339-349.

[13] Bonabeau E, Dorigo M, Theraulaz G (1999) Swarm intelligence: from natural to artificial systems. Oxford University Press, Inc, New York.

[14] Bhagade, A.S., and Puranik, P.V. 2012. Artificial Bee Colony (ABC) Algorithm for Vehicle Routing Optimization Problem. International Journal of Soft Computing and Engineering (IJSCE). Vol: 2 (2). Hal: 2231-2307.

[15] Folio, E. 2008, Distance Transform, Technical Report no0806, revision 1748, Laboratoire de Recherche et Dveloppement de l'Epita. Le Kremlin-Bictre cedex – France.

[16] Landgrebe, D.A. 2003, Signal Theory Methods in Multispectral Remote Sensing, John Wiley & Sons, Inc. Hoboken, New Jersey.

[17] Marikanis, Y. and Pardalos, P.M. 2005. Expanding Neighborhood GRASP for the Traveling Salesman Problem. Computational Optimization and Applications. 32: 231–257.

[18] Pathak, N. dan Tiwari, S.P. 2012. Travelling Salesman Problem Using Bee Colony With SPV. International Journal of Soft Computing and Engineering (IJSCE). Vol: 2(3): 2231-2307.

[19] TSPLIB http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/STSP.html diakses 23-Juni-2012, http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/ diakses 23-Juni-2012