PENYELESAIAN PERMASALAHAN PENJEMPUTAN DAN PENGANTARAN *TRAVELING SALESMAN* SESUAI ATURAN FIFO DENGAN ALGORITMA *ITERATED LOCAL SEARCH*

Ajeng Dwi Andina[1)] dan Sri Mardiyati[2)]

[1).2)]Departemen Matematika, FMIPA Universitas Indonesia Depok, Jawa Barat
E-mail :[1)]*ajengandina09@gmail.com*, [2)]*srimardiyati25@gmail.com*

ABSTRACT

The pickup and delivery traveling salesman problem with first-in-first-out (TSPPDF) is a routing problem to service n customers in the pickup and delivery which is the pickup and delivery operations must be executed in a first-in-first-out (FIFO). Starting from an origin vertex (depot), visiting all the pick-up and delivery, then returned to an origin vertex with minimum total cost or distance. In this undergraduate thesis, the FIFO Nearest Neighbor algorithm (FNN) will be used to solve TSPPDF. Then the results of TSPPDF which uses an FNN algorithm will be compared with TSP solver. After that, the results of the FNN algorithm will be optimized manually using the Iterated Local Search (ILS) algorithm.

Keywords *:*

Traveling Salesman Problem with Pickup and Delivery (TSPPD); *First-in-first-out*; FIFO *Nearest Neighbor*; *Iterated Local Search.*

1. Pendahuluan

Traveling Salesman Problem(TSP) merupakan suatu permasalahan dimana seseorang akan melakukan perjalanan ke sejumlah kota dalam satu perjalanan, dimulai dari kota asal, orang tersebut harus mengunjungi semua kota tepat satu kali dan kembali ke kota asal dengan total biaya/jarak perjalanannya minimal (Hillier et all, 2001). Penerapan *Traveling Salesman Problem* (TSP) dalam kehidupan sehari-hari antara lain, pengambilan dan pengantaran suatu objek dimana objek tersebut bisa berupa orang atau barang. *Traveling Salesman Problem*(TSP) merupakan suatu permasalahan dimana seseorang akan melakukan perjalanan ke sejumlah kota dalam satu perjalanan, dimulai dari kota asal, orang tersebut harus mengunjungi semua kota tepat satu kali dan kembali ke kota asal dengan total biaya/jarak perjalanannya minimal (Hillier et all, 2001). Penerapan *Traveling Salesman Problem* (TSP) dalam kehidupan sehari-hari antara lain, pengambilan dan pengantaran suatu objek dimana objek tersebut bisa berupa orang atau barang.

TSP memiliki beberapa perluasan, salah satunya, TSP yang diberikan kendala pengambilan dan sekaligus pengantaran suatu objek, yang dinamakan dengan *Traveling Salesman Problem with Pickup and Delivery* (TSPPD) (Mosheiov,1994). TSPPDbertujuan menemukan rute optimal yang mana digunakan untuk melayani sejumlah berhingga pelanggan. Pelayanan tersebut berupa jasa penjemputan dan sekaligus pengantaran. Akan tetapi, di dalam kehidupan nyata, pelanggan yang menggunakan jasa penjemputan dan sekaligus pengantaran sering mengeluhkan ketidakadilan dalam mendapatkan pelayanan, misalnya pelanggan mendapat pelayanan jemput pertama kali akan tetapi pelanggan tersebut baru mendapat pelayanan antar setelah pelanggan lain dilayani. Untuk mengatasi permasalahan ini, maka dilakukan penambahan kendala pada *pickup and delivery* yang harus mengikuti aturan *first-in-first-out* (FIFO), yang dikenal dengan *TSPPD with FIFO loading* (TSPPDF) (Cordeau et all, 2009). Dimana memiliki tujuan yang sama dengan TSPPD, hanya saja dalam melakukan penjemputan dan pengantaran harus diperhatikan aturan FIFO.

Sama dengan TSP, untuk menyelesaikan TSPPD dapat menggunakan metode eksak dan metode heuristik. Metode eksak tersebut diantaranya *Branch and Bound* (Kalantari et al, 1985) dan *Branch and Cut* (Hernandez-Perez et al, 2004). Sedangkan metode heuristik yang digunakan diantaranya *Tabu Search* (Gendreau et al, 1999) dan *Iterated Local Search* (Battarra M., 2013). Dalam permasalahan TSPPDF kali ini, akan diselesaikan dengan metode FIFO *Nearest Neighbor* (FNN). FNN merupakan metode yang sederhana dan mudah diterapkan.

Ide dasar FNN sama dengan metode *nearest neighbor* yaitu, memilih simpul terdekat dari simpul yang terakhir dikunjungi. Kemudian akan dibandingkan hasil penyelesaian TSPPDF yang menggunakan algoritma FNN

dengan TSP *solver* dimana metode yang digunakan dalam TSP *solver* adalah metode *branch-and-bound*. Kemudian dari solusi yang didapatkan dengan metode FNN, akan digunakan metode *Iterated Local Search* (ILS) untuk meningkatkan solusi tersebut membuat bantuan empat operator *local search*, yaitu *the couple-exchange operator*, *the component-exchange operator*, *the relocate-couple operator*, dan *the relocate-component operator* secara manual.

2. Sistem Persamaan Linier

2.1 Pemodelan TSPPDF

Pada subbab sebelumnya, telah dijelaskan mengenai TSP dengan penjemputan dan sekaligus pengantaran (TSPPD) beserta contoh TSSPD dalam kehidupan sehari-hari yang diantaranya penjemputan dan pengantaran orang dan pengambilan dan sekaligus pengiriman barang. Dalam melakukan penjemputan dan sekaligus pengantaran tersebut, seringkali pelanggan mengeluhkan ketidakadilan. Ketidakadilan tersebut berupa pelanggan yang pertama kali dijemput baru diantar setelah pelanggan lain diantar. Untuk mengatasi masalah ini, TSPPD ditambahkan suatu aturan, yaitu aturan FIFO. Dengan kata lain, jika pelanggan i mendapat penjemputan sebelum pelanggan j, maka pelanggan i harus mendapat pengantaran sebelum pelanggan j.

Penjemputan dan pengantaran *traveling salesman problem* dengan aturan FIFO (TSPPDF) merupakan permasalahan menentukan rute satu perjalanan dalam melayani n pelanggan dimana pada penjemputan dan pengantaran harus mengikuti aturan FIFO (Cordeau et al, 2009). Sama dengan TSPPD, TSPPDF juga melakukan rute perjalanan yang dimulai dari depot, lalu mengunjungi setiap tempat penjemputan dan tempat pengantaran tepat satu kali, kemudian kembali lagi ke depot.

Dalam membentuk formula TSPPDF digunakan notasi-notasi berikut ini:

- $H = \{h_1, h_2, \dots, h_n\}$merupakan himpunan pelanggan yang ingin dilayani, dimana setiap pelanggan $h_k \in H$ memiliki dua simpul, yaitu simpul u_k yang menyatakan tempat objek pelanggan h_k dijemput dan simpul u_{k+n} yang menyatakan tempat objek pelanggan h_k diantar. Dalam hal ini, objek dapat berupa orang atau barang.
- Depot dinotasikan dengan u_0.

TSPPD dapat direpresentasikan oleh graf lengkap berbobot dan tidak berarah, sehingga TSPPDF juga dapat direpresentasikan oleh graf lengkap berbobot dan tidak berarah. Notasi untuk TSPPDF yang berhubungan dengan graf $G = (V, A)$ adalah sebagai berikut:

- $V = \{u_0, u_1, \dots, u_n, u_{n+1}, \dots, u_{2n}\}$merupakan himpunan simpul dimana $\{u_1, \dots, u_n\}$ merupakan himpunan simpul penjemputan dan $\{u_{n+1}, \dots, u_{2n}\}$ merupakan himpunan simpul pengantaran.
- $A = \{(u_i, u_j) : u_i, u_j \in V, i \neq j\}$merupakan himpunan busur.
- $c_{u_i u_j}$ menyatakan bobot antara simpul $u_i \in V$ dan $u_j \in V$
- $V' = V \backslash \{u_0\}$ yaitu himpunan V yang tidak mengandung depot
- A' merupakan subset dari A dengan kedua simpul akhir berada di V'.

Variabel-variabel keputusan untuk masalah TSPPDF adalah:

- Variabel biner $x_{u_i u_j}, (u_i, u_j) \in A$

$$x_{u_i u_j} = \begin{cases} 1, & \text{Jika dilakukan perjalanan dari simpul } u_i \text{ ke simpul } u_j \\ 0, & \text{lainnya} \end{cases}$$

- Variabel biner $y^1_{u_i u_j h_k}$, menyatakan apakah terdapat busur $(u_i, u_j) \in A$ yang dilewati dalam lintasan dari depot ke simpul penjemputan pelanggan $h_k \in H$.

$$y^1_{u_i u_j h_k} = \begin{cases} 1, & \text{Jika terdapat busur } (u_i, u_j) \text{ dalam lintasan dari simpul } u_0 \text{ ke simpul } u_k \\ 0, & \text{lainnya} \end{cases}$$

- Variabel biner $y^2_{u_i u_j h_k}$, menyatakan apakah terdapat busur $(u_i, u_j) \in A$ yang dilewati dalam lintasan dari simpul penjemputan pelanggan $h_k \in H$ ke simpul pengantaran pelanggan $h_k \in H$.

$$y^2_{u_i u_j h_k} = \begin{cases} 1, & \text{Jika terdapat busur } (u_i, u_j) \text{ dalam lintasan dari simpul } u_k \text{ ke simpul } u_{k+n} \\ 0, & \text{lainnya} \end{cases}$$

- Variabel biner $y^3_{u_i u_j h_k}$, menyatakan apakah terdapat busur $(u_i, u_j) \in A$ yang dilewati dalam lintasan dari simpul pengantaran pelanggan $h_k \in H$ ke depot

$$y^3_{u_i u_j h_k} = \begin{cases} 1, & \text{Jika terdapat busur } (u_i, u_j) \text{ dalam lintasan dari simpul } u_{k+n} \text{ ke } u_0 \\ 0, & \text{lainnya} \end{cases}$$

Dan dari notasi-notasi di atas, formulasi lengkap TSPPDF adalah sebagai berikut:

$$min \sum_{(u_i,u_j)\in A} c_{u_i u_j} x_{u_i u_j} \quad (3.1)$$

Dengan kendala

$$\sum_{u_j \in V} x_{u_i u_j} = 1 \qquad u_i \in V \quad (3.2)$$

$$\sum_{u_i \in V} x_{u_i u_j} = 1 \qquad u_j \in V \quad (3.3)$$

$$\sum_{j:(u_i,u_j)\in A} y^1_{u_i u_j h_k} - \sum_{j:(u_j,u_i)\in A} y^1_{u_j u_i h_k} = \begin{cases} 1, & \text{jika } u_i = u_0 \\ -1, & \text{jika } u_i = u_k \\ 0, & \text{lainnya} \end{cases} \quad (u_i \in V, h_k \in H) \quad (3.4)$$

$$\sum_{j:(u_i,u_j)\in A} y^2_{u_i u_j h_k} - \sum_{j:(u_j,u_i)\in A} y^2_{u_j u_i h_k} = \begin{cases} 1, & \text{jika } u_i = u_k \\ -1, & \text{jika } u_i = u_{k+n} \\ 0, & \text{lainnya} \end{cases} \quad (u_i \in V, h_k \in H) \quad (3.5)$$

$$\sum_{j:(u_i,u_j)\in A} y^3_{u_i u_j h_k} - \sum_{j:(u_j,u_i)\in A} y^3_{u_j u_i h_k} = \begin{cases} 1, & \text{jika } u_i = u_{k+n} \\ -1, & \text{jika } u_i = u_0 \\ 0, & \text{lainnya} \end{cases} \quad (u_i \in V, h_k \in H) \quad (3.6)$$

$$\sum_{i:(u_i,u_l)\in A'} y^2_{u_i u_l h_k} + \sum_{i:(u_i,u_{l+n})\in A'} y^2_{u_i u_{l+n} h_k} \le 1 \quad (h_k, h_l \in H: h_k \ne h_l) \quad (3.7)$$

$$y^1_{u_i u_j h_k} + y^3_{u_i u_j h_k} = x_{u_i u_j} \quad ((u_i, u_j) \in A \backslash A', h_k \in H) \quad (3.8)$$

$$y^1_{u_i u_j h_k} + y^2_{u_i u_j h_k} + y^3_{u_i u_j h_k} = x_{u_i u_j} \quad ((u_i, u_j) \in A', h_k \in H) \quad (3.9)$$

$$x_{u_i u_j} = 0 \text{ atau } 1 \quad ((u_i, u_j) \in A) \quad (3.10)$$

$$y^1_{u_i u_j h_k} = 0 \text{ atau } 1 \quad ((u_i, u_j) \in A, h_k \in H) \quad (3.11)$$

$$y^2_{u_i u_j h_k} = 0 \text{ atau } 1 \quad ((u_i, u_j) \in A', h_k \in H) \quad (3.12)$$

$$y^3_{u_i u_j h_k} = 0 \text{ atau } 1 \quad ((u_i, u_j) \in A, h_k \in H) \quad (3.13)$$

(Cordeau et al, 2009).

Tujuan dari model TSPPDF ini adalah untuk meminimalkan total biaya yang ditempuh dalam satu perjalanan untuk penjemputan dan sekaligus pengantaran dengan cara meminimalkan total jarak perjalanan dan diformulasikan pada (3.1). Dimana fungsi kendala (3.2) dan (3.3) menyatakan semua simpul harus di kunjungi dan di tinggalkan tepat satu kali. Kemudian fungsi kendala (3.4), (3.5), dan (3.6) secara terurut menyatakan solusi harus memuat sebuah lintasan yang berasal dari depot ke simpul penjemputan pelanggan $h_k \in H$, solusi harus memuat sebuah lintasan yang berasal dari simpul penjemputan pelanggan $h_k \in H$ ke simpul pengantaran pelanggan $h_k \in H$ tanpa melewati depot, dan solusi harus memuat sebuah lintasan yang berasal dari simpul pengantaran pelanggan $h_k \in H$ ke depot. Fungsi kendala (3.7) menyatakan solusi harus mengikuti aturan FIFO. Sedangkan fungsi kendala (3.8) dan (3.9) menyatakan busur-busur yang dilewati dalam solusi.

Penyelesaian TSPPDF di atas akan diselesaikan menggunakan metode *heuristic*FIFO *nearest neighbor*.

2.2 Penyelesaian TSPPDF dengan FIFO *nearest neighbor*

FIFO *Nearest Neighbor* memiliki ide yang sama dengan metode NN yaitu memilih simpul terdekat dari simpul yang terakhir dikunjungi. Akan tetapi, langkah-langkah metode FIFO *nearest neighbor* berbeda dari metode NN. Berikut adalah langkah-langkah metode FNN.

1. Cari jarak antar simpul dan inisialisasi u_0 sebagai depot, Buat daftar penjemputan dengan simpul-simpul penjemputan dari setiap pelanggan.
2. Bandingkan jarak dari depot ke setiap simpul yang berada di daftar penjemputan. Pilih simpul yang memiliki jarak terdekat dari depot. Kemudian hapus simpul tersebut dalam daftar penjemputan dan masukkan simpul tersebut ke dalam rute. Lalu buat daftar antrian pengantaran dan masukkan simpul pengantaran yang bersesuaian dengan simpul penjemputan di atas sebagai antrian terakhir.
3. Cari simpul yang memiliki jarak terdekat dari simpul terakhir yang dikunjungi. Simpul yang akan dikunjungi tersebut harus simpul yang berada di daftar penjemputan atau berada di antrian pertama dalam daftar antrian pengantaran.
4. Setelah mendapatkan simpul yang terdekat misalkan simpul u_i, perhatikan kembali simpul u_i berasal dari daftar penjemputan atau daftar pengantaran.
   - Jika simpul u_i berasal dari daftar penjemputan, maka hapus simpul tersebut dalam daftar penjemputan, masukkan ke dalam rute, kemudian tambahkan simpul pengantaran yang bersesuaian dengan simpul u_i yaitu simpul u_{i+n} ke antrian terakhir dalam daftar antrian pengantaran.
   - Jika simpul u_i berasal dari antrian pertama dalam daftar pengantaran, maka hapus simpul dari daftar, masukkan ke dalam rute, kemudian cari kembali simpul yang dekat dengan simpul u_{i+n}.
5. Ulangi langkah 3 dan 4 hingga tidak ada lagi simpul yang berada di daftar penjemputan maupun daftar pengantaran, kemudian dari simpul terakhir yang dikunjungi kembali ke depot. Lalu proses berhenti, tetapkan SS akhir, dan hitung jarak.

2.3 Peningkatan Solusi TSPPDF dengan Algoritma *Iterated Local Search*

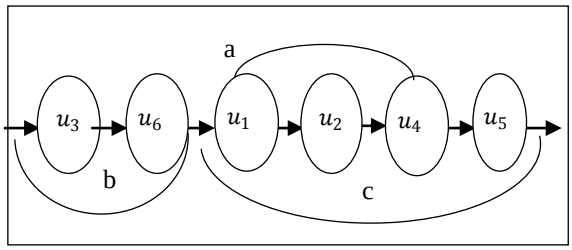
Dalam algoritma ILS menggunakan 3 parameter yang dinotasikan dengan:

- α : jumlah iterasi sejak update terakhir dari solusi yang lama.
- μ : jumlah iterasi maksimum tanpa mengupdate solusi yang lama.

- θ : jumlah*perturbation* yang dilakukan.

Sebelum membahas lebih lanjut mengenai algoritma *Iterated Local Search* (ILS), terlebih dahulu akan dijelaskan mengenai beberapa definisi-definisi yang nantinya akan digunakan pada tahapan selanjutnya. Definisi-definisi tersebut adalah sebagai berikut:

- *Couple* didefinisikan sebagai pasangan (u_k, u_{k+n}) simpul penjemputan dan simpul pengantaran dari pelanggan $h_k \in H$. Contoh *couple* digambarkan pada Gambar 2.1(a)
- *Component* didefinisikan sebagai lintasan dari simpul penjemputan ke simpul pengantaran, pada awal dan akhir tidak ada pelanggan yang berada di kendaraan. Contoh *component* digambarkan pada Gambar 2.1(b) dan 2.1 (c)



Gambar 2.1 (a) *couple*, (b) *component* berukuran 2, dan (c) *compoent* berukuran 4

- Posisi didefinisikan sebagai urutan dimana simpul $u_i \in V$ dikunjungi. Dinotasikan dengan $\pi(u_i)$dimana $\pi(u_0) = 0$. Dari Gambar 3.1, jika $\pi(u_3) = p$, maka $\pi(u_6) = p + 1, \pi(u_1) = p + 2, \pi(u_2) = p + 3, \pi(u_4) = p + 4$ dan $\pi(u_5) = p + 5$.
- *Service sequence* (SS)didefinisikan sebagai urutan penjemputan pelanggan. Dinotasikan dengan $\sigma(q)$ yang berarti urutan ke-q dalam SS. Dari Gambar 3.1, jika $\sigma(s) = h_3$, maka $\sigma(s + 1) = h_1$ dan $\sigma(s + 2) = h_2$.

Sebelumnya telah dijelaskan bahwa di dalam algoritma ILS digunakan *local search* untuk mengoptimalkan solusi. *Local search* tersebut berupa operator sebanyak 4 operator. Operator-operator tersebut adalah:

- *The couple-exchange operator*

  Operator ini memilih dua *couple* yaitu(u_k, u_{k+n}) dan (u_l, u_{l+n}) yang kemudian posisi u_k ditukar dengan posisi u_l dan posisi u_{k+n} ditukar dengan posisi u_{l+n}. Lakukan operasi ini hingga SS terbaik saat ini sama dengan SS baru.
- *The component-exchange operator*

  Cara kerja operator ini sama dengan operator *couple-exchange* hanya saja yang ditukar berupa dua *component*. Lakukan juga operasi ini untuk setiap *component* yang ada di solusi awal. Contoh untuk operator *component-exchange* dapat dilihat dalam gambar 3.4 dimana *component* yang ingin ditukar adalah $(-u_3 - u_2 - u_6 - u_5-)$dengan $(-u_1 - u_4-)$.

Misal rute awal
$u_0 - u_1 - u_4 - u_3 - u_2 - u_6 - u_5 - u_0$
Tukar
Tukar
Rute setelah penggunaan operator *component-exchange*
$u_0 - u_3 - u_2 - u_6 - u_5 - u_1 - u_4 - u_0$

Gambar 2.2 Operator *component-exchange*

- *The relocate-component operator*

  Operator ini digunakan untuk menempatkan *component* dalam posisi yang berbeda. Lakukan operasi ini pada setiap *component* yang ada

Misal rute awal
$u_0 - u_1 - u_4 - u_3 - u_6 - u_2 - u_5 - u_0$
Rute setelah penggunaan operator *component-exchange*
$u_0 - u_2 - u_5 - u_1 - u_4 - u_3 - u_6 - u_0$
Dan
$u_0 - u_1 - u_4 - u_2 - u_5 - u_3 - u_6 - u_0$

Gambar 2.3 Operator *relocate-component*

- *The relocate-couple operator*

  Operator ini digunakan untuk menempatkan *couple* dalam posisi yang berbeda dan dilakukan hingga SS terbaik saat ini sama dengan SS baru. Dalam penempatan simpul pengantaran suatu *couple*, harus memenuhi kendala FIFO.

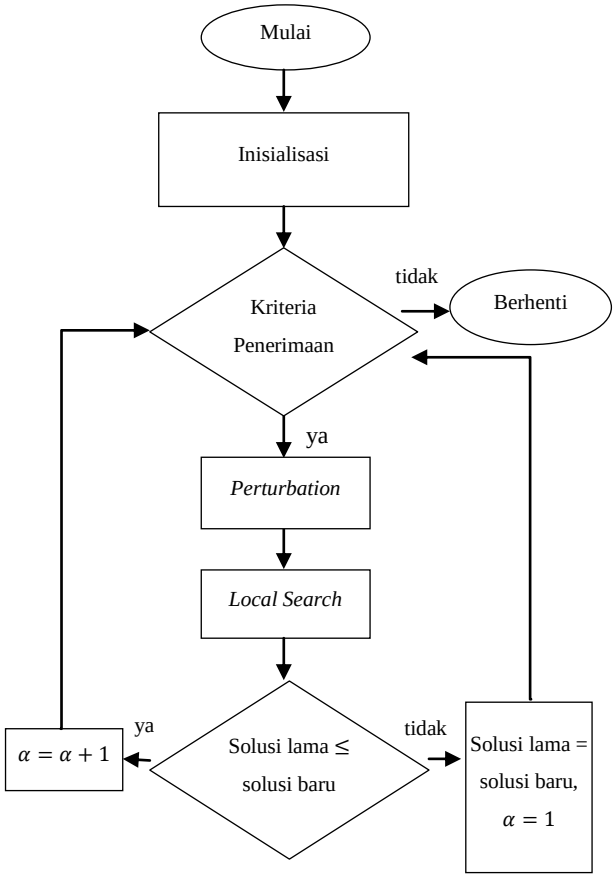
Misal rute awal
$u_0 - u_1 - u_4 - u_3 - u_6 - u_2 - u_5 - u_0$
Rute setelah penggunaan operator *couple-exchange*
$u_0 - u_2 - u_5 - u_1 - u_4 - u_3 - u_6 - u_0$
Dan
$u_0 - u_2 - u_1 - u_5 - u_4 - u_3 - u_6 - u_0$

Gambar 2.4 Operator *relocate-couple*

Gunakan ke empat operator *local search* di atas untuk mengubah solusi sebelumnya menjadi kemungkinan-kemungkinan solusi baru hingga semua operator telah dijalankan dimana perubahan solusi tersebut didasari dengan SS yang didapat dari *perturbation*. Kemudian dari kemungkinan-kemungkinan tersebut pilih satu solusi baru yang memiliki jarak terpendek. Setelah itu solusi baru tersebut akan dibandingkan dengan solusi terbaik saat ini.

Berikut adalah gambaran langkah-langkah algoritma ILS bersama dengan flowchartnya.

1. *(Inisialisasi)* Tetapkan solusi yang dihasilkan algoritma FNN sebagai solusi terbaik, tetapkan SS, dan tetapkan $\alpha = 1$.
2. *(Kriteria Penerimaan)*Jika $\alpha < \mu$, lanjutkan ke langkah selanjutnya. Jika $\alpha \geq \mu$, proses berhenti.
3. *(Perturbation)* Inisialisasi SS yang diganggu dengan menyalin SS dari solusi terbaik saat ini. Sebanyak θ kali, pilih secara acak dua pelanggan yang berada dalam SS yang diganggu, kemudian tukar keduanya. Tetapkan SS yang diganggu sebagai SS baru yang digunakan untuk membuat solusi.
4. *(Local Search)* Gunakan operator *couple-exchange, component-exchange, relocate-component,* dan *relocate-couple* yang didasari dengan SS baru untuk membentuk kemungkinan-kemungkinan solusi baru, hingga semua operator telah dijalankan. Kemudian pilih rute yang memiliki jarak terdekat dari kemungkinan-kemungkinan tersebut. Setelah itu bandingkan dengan solusi terbaik saat ini.
   - Jika solusi baru memiliki total jarak yang lebih kecil, maka update solusi terbaik dan buat $\alpha = 1$.
   - Jika solusi baru memiliki total jarak yang lebih besar, maka $\alpha = \alpha + 1$.



Gambar 2.5 Flowchart Algoritma ILS

3. Hasil Penelitian

Dalam penelitian ini, diberikan masalah pada tabel 3.1 dan tabel 3.2 dalam melayani 3 pelanggan serta masalah yang dibuat sendiri untuk melayani 5 dan 10 pelanggan yang terdapat dalam tabel 3.3. Berikut adalah data untuk 3 pelanggan.

Tabel 3.1 Simpul penjemputan dan pengantaran pelanggan

Pelanggan	Tempat Penjemputan	Tempat Pengantaran
h_1	u_1	u_4
h_2	u_2	u_5
h_3	u_3	u_6

Tabel 3.2 Koordinat simpul

Simpul	Koordinat Simpul
u_0	(1,7)
u_1	(2,4)
u_2	(1,5)
u_3	(2,6)
u_4	(3,5)
u_5	(3,4)
u_6	(2,7)

Kemudian dengan menggunakan formula jarak Euclid didapat jarak antar simpul yang direpresentasikan dalam matriks yang disebut dengan matriks *adjacent* atau matriks kedekatan, seperti berikut

$$c_{u_i u_j} = \begin{bmatrix} 0 & 3.6 & 2 & 1.4 & 2.8 & 3.6 & 1 \\ 3.6 & 0 & 1.4 & 2 & 1.4 & 1 & 3 \\ 2 & 1.4 & 0 & 1.4 & 2 & 2.2 & 2.2 \\ 1.4 & 2 & 1.4 & 0 & 1.4 & 2.2 & 1 \\ 2.8 & 1.4 & 2 & 1.4 & 0 & 1 & 2.2 \\ 3.6 & 1 & 2.2 & 2.2 & 1 & 0 & 3.2 \\ 1 & 3 & 2.2 & 1 & 2.2 & 3.2 & 0 \end{bmatrix}$$

Dan berikut ini adalah jarak antar simpul-simpul yang dibentuk dalam suatu matriks dimana melayani 5 dan 10 pelanggan. Dimana matriks dibawah ini berupa daftar dalam daftar. Baris pertama matriks adalah daftar pertama dan baris kedua matriks adalah daftar kedua, begitu seterusnya. Sedangkan kolom pertama matriks adalah kumpulan entri pertama dari setiap daftar dan kolom kedua matriks adalah kumpulan entri kedua dari setiap daftar begitu seterusnya.

Tabel 3.3 Matriks jarak

Jumlah Simpul	Jarak
11	[[0,12.4,8,15,9.2,11,13,9.2,14,16,12.4],[12.4,0,11,16,10.1,14,10.1,13,9.2,7,15],[8,11,0,13,18,15,17,9.2,8,16,13],[15,16,13,0,9.2,10.1,12.4,10.1,16,17,9.2],[9.2,10.1,18,9.2,0,12.4,11,18,9.2,12.4,13],[11,14,15,10.1,12.4,0,10.1,11,14,16,15],[13,10.1,17,12.4,11,10.1,0,8,11,9.2,15],[9.2,13,9.2,10.1,18,11,8,0,12.4,17,16],[14,9.2,8,16,9.2,14,11,12.4,0,13,10.1],[16,7,16,17,12.4,16,9.2,17,13,0,11],[12.4,15,13,9.2,13,15,15,16,10.2,11,0]]
21	[[0,13.3,7,10,19,17,25,14,11,6,9.4,15,9.4,14,21.1,19,25,16.2,13.3,12,11],[13.3,0,11,18,21.1,8.3,12,6,20,13.3,18,14,8.3,17,21.1,9.4,10,13.3,16.2,20,5],[7,11,0,15,5.3,22,24,16.2,11,7,23,8.3,21.1,18,10,20,21.1,12,13.3,16.2,19],[10,18,15,0,10,15,21.1,8.3,12,19,20,12,18,21.1,14,15,11,16.2,9.4,25,11],[19,21.1,5.3,10,0,6,18,15,13.3,11,14,23,11,9.4,17,14,20,23,16.2,13.3,25],[17,8.3,22,15,6,0,9.4,14,19,21.1,7,18,21.1,14,11,7,21.1,10,6,15,12],[25,12,24,21.1,18,9.4,0,12,7,18,10,19,9.4,25,13.3,15,17,11,21.1,13.3,8.3],[14,6,16,8.3,15,14,12,0,5.3,23,16.2,11,15,19,23,10,16.2,22,12,8.3,17],[11,20,11,12,13.3,19,7,5.3,0,18,8.3,18,21.1,14,20,6,10,20,13.3,19,12],[6,13.3,7,19,11,21.1,18,23,18,0,22,17,19,12,11,8.3,20,19,5.3,15,22],[9.4,18,23,20,14,7,10,16.2,8.3,22,0,12,16.2,5.3,22,13.3,21.1,15,11,21.1,14],[15,14,8.3,12,23,18,19,11,18,17,12,0,13.3,21.1,19,6,14,8.3,22,15,20],[9.4,8.3,21.1,18,11,21.1,9.4,15,21.1,19,16.2,13.3,0,17,5.3,22,24,18,13.3,7,23],[14,17,18,21.1,9.4,14,25,19,14,12,5.3,21.1,17,0,12,17,21.1,10,14,21.1,14],[21.1,21.1,10,14,17,11,13.3,23,20,11,22,19,5.3,12,0,6,20,17,10,13.3,16],[19,9.4,20,15,14,7,15,10,6,8.3,13.3,6,22,17,6,0,11,16.2,21.1,21.1,9.4],[25,10,21.1,11,20,21.1,17,16.2,10,20,21.1,14,24,21.1,20,11,0,14,9.4,21.1,12],[16.2,13.3,12,16.2,23,10,11,22,20,19,15,8.3,18,10,17,16.2,14,0,5.3,23,18],[13.3,16.2,13.3,9.4,16.2,6,21.1,12,13.3,5.3,11,22,13.3,14,10,21.1,9.4,5.3,0,20,15],[12,20,16.2,25,13.3,15,13.3,8.3,19,15,21.1,15,7,21.1,13.3,21.1,20,23,20,0,22],[11,5.3,19,11,25,12,8.3,17,12,22,14,20,23,14,16.2,9.4,12,18,15,22,0]]

Selanjutnya dari tiga masalah di atas akan dilakukan komparasi penyelesaiannya mengunakan algoritma FNN dengan TSP *Solver* yang mana menggunakan metode *Branch and Bound* dimana akan dilihat metode mana yang lebih optimal. Dan Tabel 3.4 berikut adalah tabel perbandingan hasil dari algoritma FNN dan TSP *solver*.

Tabel 3.4 Perbandingan Algoritma FNN dengan TSP *Solver*

Jumlah Simpul	Metode	Rute	Jarak
7 Simpul	FNN	$u_0 - u_3 - u_6 - u_1 - u_2 - u_4 - u_5 - u_0$	10.8
	TSP *Solver*	$u_0 - u_3 - u_1 - u_2 - u_6 - u_4 - u_5 - u_0$	11.2
11 Simpul	FNN	$u_0 - u_2 - u_7 - u_3 - u_4 - u_8 - u_1 - u_9 - u_6 - u_5 - u_{10} - u_0$	108.6
	TSP *Solver*	$u_0 - u_2 - u_1 - u_4 - u_3 - u_5 - u_7 - u_6 - u_9 - u_8 - u_{10} - u_0$	110.1
21 Simpul	FNN	$u_0 - u_9 - u_2 - u_4 - u_5 - u_{10} - u_8 - u_7 - u_1 - u_6 - u_{19} - u_{12} - u_{14} - u_{15} - u_{20} - u_3 - u_{18} - u_{17} - u_{11} - u_{16} - u_{13} - u_0$	187.0
	TSP *Solver*	$u_0 - u_9 - u_2 - u_4 - u_5 - u_1 - u_7 - u_8 - u_6 - u_{10} - u_3 - u_{19} - u_{12} - u_{14} - u_{15} - u_{11} - u_{17} - u_{18} - u_{16} - u_{20} - u_{13} - u_0$	193.2

Dari perbandingan di atas, terlihat bahwa solusi yang dihasilkan dengan menggunakan algoritma FNN memiliki jarak yang lebih pendek daripada TSP *solver*, yang artinya algoritma FNN memberikan solusi yang lebih optimal dibandingkan dengan TSP *solver*.

Dan berikut merupakan Tabel solusi TSPPDF dalam melayani pelanggan yang lebih banyak dengan menggunakan algoritma FNN data jarak diambil dari TSPLIB (Reinelt G., 1991).

Tabel 3.5 Hasil FNN

Jumlah Simpul	Rute	Jarak
25 Simpul	$u_0 - u_5 - u_{11} - u_7 - u_4 - u_3 - u_8 - u_1 - u_2 - u_9 - u_6 - u_{12} - u_{17} - u_{10} - u_{23} - u_{19} - u_{16} - u_{15} - u_{20} - u_{13} - u_{14} - u_{21} - u_{18} - u_{24} - u_{22} - u_0$	12817.0
51 Simpul	$u_0 - u_5 - u_{11} - u_{14} - u_{21} - u_{23} - u_7 - u_{18} - u_{13} - u_4 - u_3 - u_8 - u_1 - u_2 - u_{24} - u_9 - u_{15} - u_{25} - u_6 - u_{12} - u_{30} - u_{17} - u_{10} - u_{20} - u_{19} - u_{16} - u_{22} - u_{36} - u_{39} - u_{46} - u_{48} - u_{32} - u_{43} - u_{38} - u_{29} - u_{28} - u_{33} - u_{26} - u_{27} - u_{49} - u_{34} - u_{40} - u_{50} - u_{31} - u_{37} - u_{42} - u_{35} - u_{45} - u_{44} - u_{41} - u_{47} - u_0$	18975.6
75 Simpul	$u_0 - u_5 - u_{11} - u_{14} - u_{21} - u_{31} - u_{23} - u_7 - u_{27} - u_{18} - u_{13} - u_4 - u_3 - u_8 - u_{34} - u_{26} - u_{33} - u_{24} - u_2 - u_1 - u_9 - u_{29} - u_{15} - u_{25} - u_6 - u_{12} - u_{35} - u_{32} - u_{30} - u_{42} - u_{37} - u_{17} - u_{10} - u_{20} - u_{19} - u_{16} - u_{22} - u_{36} - u_{48} - u_{28} - u_{51} - u_{58} - u_{68} - u_{60} - u_{44} - u_{64} - u_{55} - u_{50} - u_{41} - u_{40} - u_{45} - u_{71} - u_{63} - u_{70} - u_{61} - u_{39} - u_{38} - u_{46} - u_{66} - u_{52} - u_{62} - u_{43} - u_{49} - u_{72} - u_{69} - u_{67} - u_{74} - u_{54} - u_{47} - u_{57} - u_{56} - u_{53} - u_{59} - u_{73} - u_{65} - u_0$	28223.5

Kemudian dari contoh masalah dalam menyelesaikan kasus mencari jarak terpendek untuk melayani 3 pelanggan di pembahasan sebelumnya dilakukan peningkatan solusi dengan menggunakan algoritma ILS secara manual. Dengan menggunakan FNN di dapat total jarak adalah 10.8. Kemudian dengan melakukan langkah-langkah algoritma ILS di subab sebelumnya, di dapatkan total jarak sebesar 8.8. Hal ini menunjukkan dalam kasus melayani 3 pelanggan ini, algoritma ILS dapat meningkatkan solusi yang dihasilkan dari FNN.

4. Kesimpulan

Kesimpulan dari penelitian ini adalah sebagai berikut:

- Berdasarkan tabel 3.4 dan tabel 3.5 pada pembahasan di atas, algoritma FIFO *Nearest Neighbor* dapat menyelesaikan suatu masalah TSPPDF baik dalam melayani pelanggan yang sedikit ataupun pelanggan yang lebih banyak.
- Berdasarkan tabel 3.5 pada pembahasan di atas, hasil penyelesaian TSPPDF dalam melayani 3, 5, dan 10 pelanggan dengan menggunakan algoritma FNN tidak berbeda jauh dengan menggunakan TSP *solver*. Akan tetapi, kepuasan pelanggan terhadap pelayanan penjemputan dan pengantaran lebih puas dengan hasil menggunakan algoritma FNN dibandingkan dengan TSP *solver*.
- Berdasarkan percobaan secara manual untuk menyelesaikan kasus mencari jarak terpendek untuk melayani 3 pelanggan pada pembahasan di atas, algoritma ILS dapat meningkatkan solusi yang dihasilkan oleh algoritma FNN.

REFERENSI

[1] Davendra, D. (2010). *Traveling Salesman Problem, Theory and Applications.* India: InTech.

[2] Dumitrescu, I. (2008). Polyhedral Results for The Pickup and Delivery Travelling Salesman Problem. *Technical report CIRRELT-2008-07*, 1-53.

[3] Erdogan, G., Cordeau, J.-F., & Laporte, G. (2009). The Pickup and Delivery Traveling Salesman Problem with FIFO loading. *Computer & Operations Research*, 1800-1808.

[4] Gutin, G., & Abraham, P. P. (2005). *The Traveling Salesman Problem and Its Variations.* USA.

[5] Hillier, F. S., & Lieberman, G. J. (2001). *Introduction to Operations Research.* USA: Mc Graw Hill Education.

[6] Jacob, B. (1990). *Linear Algebra*. United States of America: W.H. Freeman and Company.

[7] Karkory, F. A., & Abudalmoli, A. A. (2013). Implementation of Heuristics for Solving Travelling Salesman Problem Using Nearest Neighbor and Minimum Spanning Tree Algorithm. *International Journal of Mathematical, Computational, Natural and Physical Engineering Vol:7, No:10*, 987-997.

[8] Lourenco, H. R., Martin, O. C., & Stutzle, T. (2003). *Iterated Local Search*, 1-42.

[9] Munir, R. (2010). *Matematika Diskrit.* Bandung: Informatika Bandung.

[10] Reinelt, G. (1991). TSPLIB-- A Traveling Salesman Problem Library. *ORSA Journal on Computing*, 3:3:76-84.

[11] Taha, H. A. (2007). *Operations Research: An Introduction.* United States of America: Pearson Education.