EFEKTIVITAS ALGORITMA *SIMULATED ANNEALING* DAN *LARGE NEIGHBORHOOD SEARCH* DALAM PENYELESAIAN *PICKUP AND DELIVERY VEHICLE ROUTING PROBLEM WITH TIME WINDOWS*

SKRIPSI

Diajukan kepada Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Negeri Yogyakarta
untuk Memenuhi Sebagian Persyaratan guna Memperoleh Gelar Sarjana Sains

Oleh:
ANIE VIKTARIA
NIM 10305141039

PROGRAM STUDI MATEMATIKA
JURUSAN PENDIDIKAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS NEGERI YOGYAKARTA
2015

PERSETUJUAN

Skripsi yang berjudul:

EFEKTIVITAS ALGORITMA *SIMULATED ANNEALING* DAN *LARGE NEIGHBORHOOD SEARCH* DALAM PENYELESAIAN *PICKUP AND DELIVERY VEHICLE ROUTING PROBLEM WITH TIME WINDOWS*

Oleh:

Anie Viktaria

NIM 10305141039

Telah disetujui pada tanggal 8 Januari 2015

untuk diujikan di hadapan Dewan Penguji Skripsi Program Studi Matematika

Jurusan Pendidikan Matematika

Fakultas Matematika dan Ilmu Pengetahuan Alam

Universitas Negeri Yogyakarta

Yogyakarta, 8 Januari 2015

Dosen Pembimbing

Nur Insani, M. Sc

NIP. 19810406 200501 2 005

PENGESAHAN

Skripsi dengan judul:

" EFEKTIVITAS ALGORITMA *SIMULATED ANNEALING* DAN *LARGE NEIGHBORHOOD SEARCH* DALAM PENYELESAIAN *PICKUP AND DELIVERY VEHICLE ROUTING PROBLEM WITH TIME WINDOWS* "

Yang disusun oleh:

Nama : Anie Viktaria

NIM : 10305141039

Prodi : Matematika

Skripsi ini telah diujikan di depan Dewan Penguji Skripsi Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Yogyakarta pada tanggal 16 Januari 2015 dan dinyatakan **LULUS**.

Dewan Penguji

Nama	Jabatan	Tanda Tangan	Tanggal
Nur Insani, M.Sc NIP.198104062005012005	Ketua Penguji	.....	26-1-2015
Eminugroho R. S., M. Sc NIP. 198504142009122003	Sekretaris Penguji	.....	23-1-2015
Sahid, M.Sc NIP. 196509051991011001	Penguji I (Utama)	.....	22-1-2015
Emut, M. Si NIP.196212151988121001	Penguji II (Pendamping)	.....	23-1-2015

Yogyakarta, 27 Januari 2015
Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Negeri Yogyakarta
Dekan,

Dr. Hartono
NIP.196203291987021002

PERNYATAAN

Yang bertanda tangan di bawah ini, saya :

Nama	: Anie Viktaria
NIM	: 10305141039
Program Studi	: Matematika
Fakultas	: Matematika dan Ilmu Pengetahuan Alam
Judul Skripsi	: **EFEKTIVITAS ALGORITMA *SIMULATED ANNEALING* DAN *LARGE NEIGHBORHOOD SEARCH* DALAM PENYELESAIAN *PICKUP AND DELIVERY VEHICLE ROUTING PROBLEM WITH TIME WINDOWS***

Menyatakan bahwa skripsi ini benar-benar karya saya sendiri dan sepanjang pengetahuan saya, tidak terdapat karya atau pendapat yang ditulis atau diterbitkan orang lain, kecuali pada bagian-bagian tertentu yang diambil sebagai acuan atau kutipan dengan mengikuti tata penulisan karya ilmiah yang telah lazim.

Apabila ternyata terbukti pernyataan saya ini tidak benar, maka sepenuhnya menjadi tanggungjawab saya, dan saya bersedia menerima sanksi sesuai ketentuan yang berlaku.

Yogyakarta, 8 Januari 2015
Yang Menyatakan,

Anie Viktaria
NIM. 10305141039

MOTTO

“Jika kau tanya kenapa aku memilihmu, itu karena Allah memberikan cinta yang ditujukan kepadamu “

(Asma Nadia)

“Ketika masalah datang, Allah tidak meminta kita memikirkan jalan keluar hingga penat. Allah hanya meminta kita sabar dan shalat.”

(Ade a.k.a Rindu)

“ Sejauh apapun jarak kita dengan mimpi-mimpi kita, tak ada satu teorema pun yang mampu mematahkan usaha dan meruntuhkan kemauan yang tinggi. Kelak, pundi-pundi semangat dan kerja keras akan membuahkan kecupan yang manis dari Allah. “

(Maulana Kafaby)

“Sesungguhnya bersama kesulitan ada kemudahan, maka apabila engkau telah selesai (dari suatu urusan), tetaplah bekerja keras (untuk urusan yang lain)”

(Qs. Al Insyirah 6-7)

PERSEMBAHAN

Skripsi ini saya persembahkan untuk

Suami tercinta, Aditya Alif Pradana. Terimakasih, terimakasih, dan terimakasih. Semoga Allah mengabulkan doa di tiap sujud kita, agar kita tidak hanya dilanggengkan di dunia, tapi juga diabadikan di taman surgaNya.

Keluarga tersayang, Mama dan Papa Purworejo, Ibu dan Papa Madiun, Mas Arie, Mba Achy, si kecil Abrisam, Dede Tafiv, Dek Onky, Dek Anggita. Terimakasih atas do'a dan semangat yang tak henti-hentinya hingga detik ini. Semoga Allah ridho menghimpun kita di surga-Nya.

Teman-teman Matematika angkatan 2010, keluarga Himatika, UKKI, dan FOSMA ESQ 165, yang selalu mencontohkan kebaikan dan tekad juangnya. Semoga mengalir lebih banyak pahala untuk kalian.

EFEKTIVITAS ALGORITMA *SIMULATED ANNEALING* DAN *LARGE NEIGHBORHOOD SEARCH*DALAM PENYELESAIAN *PICKUP AND DELIVERY VEHICLE ROUTING PROBLEM WITH TIME WINDOWS*

Oleh:
Anie Viktaria
NIM. 10305141039

ABSTRAK

Pickup and delivery vehicle routing problem with time windows(PDPTW) merupakan masalah penentuan rute optimal kendaraan untuk memenuhi permintaan pelanggan yang terdiri dari pelayanan jemput dan antar dengan kendala kapasitas, *time windows, precedence,* dan *pairing.* Permasalahan PDPTW dapat diselesaikan dengan menggunakan algoritma yang bersifat eksak dan heuristik. Dalam menyelesaikan masalah PDPTW, tidak banyak penelitian yang membandingkan efektivitas dari penggunaan algoritma-algoritma tersebut. Oleh karena itu, penelitian ini akan membandingkan efektivitaspenggunaan algoritma *simulated annealing* (SA) dan *large neighborhood search* (LNS) dalam menyelesaikan masalah PDPTW.

Pada penelitian ini akan dijelaskan mengenai penggunaan algoritma SA dan LNS dalam penyelesaian masalah PDPTW yang kemudian diimplementasikan pada data *benchmark* Benavent, dkk dan Li & Lim dengan menggunakan perangkat lunak. Selanjutnya akan dianalisa efektivitas kedua algoritma tersebut yang diukur dari banyaknya rute dan total jarak tempuh kendaraan berdasarkan hasil implementasi yang diperoleh. Algoritma SA dan LNS juga akan digunakan dalam menyelesaikan contoh permasalahan PDPTW untuk mengetahui efektivitasnya pada masalah nyata.

Berdasarkan analisis efektivitas jumlah rute dan total jarak tempuh kendaraan, diperoleh bahwa algoritma SA lebih efektif dalam mengurangi jumlah rute kendaraan pada 51 dari 100 tipe data *benchmark* yang diujicobakan. Di sisi lain, algoritma LNS lebih efektif dalam mengurangi total jarak tempuh kendaraan pada 90 dari 100 tipe data *benchmark* yang diujicobakan. Hasil yang sama juga ditunjukkan pada contoh permasalahan PDPTW yang diselesaikan dengan kedua algoritma tersebut.

Kata kunci: *pickup and delivery vehicle routing problem with time windows*(PDPTW), *simulated annealing ,large neighborhood search*

KATA PENGANTAR

Syukur Alhamdulillah penulis panjatkan kepada Allah SWT atas rahmat dan hidayah yang diberikan kepada penulis sehingga penulis dapat menyelesaikan tugas akhir skripsi ini. Shalawat dan salam senantiasa tercurah kepada manusia pilihan, Nabi Muhammad SAW, yang selalu kita harapkan syafa'atnya di hari akhir.

Skripsi yang berjudul "Efektivitas Algoritma *Simulated Annealing* dan *Large Neighborhood Search* dalam Penyelesaian *Pickup and Delivery Vehicle Routing Problem with Time Windows*" disusun untuk memenuhi salah satu syarat kelulusan guna meraih gelar Sarjana Sains pada Program Studi Matematika, Universitas Negeri Yogyakarata. Penyelesaian skripsi ini tidak lepas dari doa, dukungan, bantuan, dan bimbingan dari berbagai pihak. Oleh karena itu, penulis mengucapkan terimakasih kepada:

1. Bapak Dr. Hartono, selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Yogyakarta,
2. Bapak Dr. Sugiman, selaku Ketua Jurusan Pendidikan Matematika, Universitas Negeri Yogyakarta,
3. Bapak Dr. Agus Maman Abadi, selaku Ketua Program Studi Matematika, Universitas Negeri Yogyakarta dan Penasehat Akademik yang telah memberikan arahan, motivasi, dan dukungan akademik kepada penulis. Semoga Bapak terus dapat menjadi teladan bagi *civitas* akademika kampus,
4. Ibu Nur Insani, M.Sc, selaku dosen pembimbing yang telah meluangkan waktu dan pikiran di tengah kesibukannya dan selalu memberi semangat

kepada penulis untuk menyelesaikan skripsi ini. Semoga Allah memberikan balasan terbaik atas kebaikan dan bimbingan Ibu selama ini,

5. seluruh dosen, staf, dan karyawan Jurusan Pendidikan Matematika Universitas Negeri Yogyakarta. Terimakasih atas ilmu yang bermanfaat dan bantuan yang diberikan kepada penulis,
6. suami tercinta, Papa dan Mama Purworejo, Papa dan Ibu Madiun, kakak, dan adik yang telah meberikan doa, dukungan, serta semangat kepada penulis. Semoga Allah ridha menghimpun kita di surga-Nya kelak,
7. keluarga Matematika angkatan 2010 yang ikut memberikan arti dalam pembelajaran hidup penulis, dan
8. seluruh pihak yang tidak dapat disebutkan satu persatu, yang telah memberikan dukungan, bantuan, dan motivasi kepada penulis. Semoga semua selalu dalam jalan kebaikan.

Penulis menyadari adanya ketidaktelitian, kekurangan, dan kesalahan dalam penyusuan tugas akhir skripsi ini. Penulis menerima kritik dan saran yang bersifat membangun. Semoga penulisan tugas akhir skripsi ini dapat memberikan manfaat bagi penulis dan pembaca.

Yogyakarta, Januari 2015

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

A. Latar Belakang

Logistik merupakan suatu proses kegiatan yang berhubungan dengan pengadaan, penyimpanan, pengelolaan, persediaan dan pendistribusian barang atau jasa. Peran logistik dalam suatu perusahaan menjadi penting karena besarnya kebutuhan setiap unit operasional akan barang atau jasa dalam menjalankan kegiatannya. Logistik memberikan efesiensi dan efektivitas untuk mencapai keberhasilan dari tujuan suatu perusahaan. Oleh karena itu, perusahaan harus memiliki kemampuan untuk dapat mengelola logistik melalui manajemen logistik yang terarah, disiplin, dan penuh tanggungjawab sehingga dapat memberikan pelayanan yang baik kepada pelanggan.

Manajemen logistik dapat diartikan sebagai suatu sistem yang berfungsi untuk merencanakan, melaksanakan, dan mengendalikan efisiensi dan efektivitas pengadaan, penyimpanan dan aliran barang atau jasa dari produsen ke konsumen. Permasalahan distribusi barang atau jasa dalam manajemen logistik merupakan salah satu aspek yang harus diperhatikan karena permasalahan distribusi tersebut memiliki pengaruh yang cukup signifikan terhadap biaya dan tingkat pelayanan kepada pelanggan. Selain itu juga dikarenakan oleh banyaknya kendala yang harus dihadapi dalam proses distribusi, diantaranya adalah banyaknya permintaan yang fluktuatif dan berbeda-beda pada setiap pelanggan, keterbatasan jumlah dan kapasitas kendaraan yang dimiliki, adanya batasan waktu pengiriman, dan lokasi pelanggan yang tersebar secara geografis.

Berbagai usaha diperlukan untuk mengatasi kendala dalam pendistribusian barang atau jasa. Salah satu usaha yang dapat dilakukan adalah meminimalisasi biaya distribusi dengan menekan biaya transportasi melalui penentuan rute optimal kendaraan. Rute kendaraan yang optimal dapat meminimalkan biaya transportasi global terkait jarak dan biaya tetap yang berhubungan dengan kendaraan, meminimalkan banyaknya kendaraan yang dibutuhkan untuk melayani semua pelanggan, menyeimbangkan rute-rute dalam hal waktu perjalanan dan muatan kendaraan, serta meminimalkan pinalti akibat keterlambatan pengiriman (Toth dan Vigo, 2002). Masalah penentuan rute optimal kendaraan ini selanjutnya disebut sebagai *vehicle routing problem* (VRP).

Menurut Yeun dkk (2008), *vehicle routing problem* (VRP) merupakan masalah penentuan rute optimal kendaraan dalam pendistribusian barang atau jasa dari satu atau lebih depot ke sejumlah pelanggan di lokasi yang berbeda dengan permintaan yang telah diketahui dan memenuhi sejumlah kendala. VRP memiliki tujuan untuk meminimalkan biaya yang dapat direpresentasikan oleh total jarak tempuh atau banyaknya kendaraan yang digunakan ataupun keduanya. Aplikasi dari permasalahan VRP dapat ditemukan dalam kehidupan sehari-hari, seperti pada penentuan rute bus sekolah, distribusi surat kabar, pengumpulan sampah, pelayanan jasa kurir, dan lain sebagainya.

Beberapa jenis permasalahan utama pada VRP menurut Toth dan Vigo (2002) yaitu *capacitated vehicle roting problem* (CVRP), *distance constrained vehicle routing problem* (DCVRP), *multiple depots vehicle routing problem* (MDVRP), *vehicle routing problem with pickup and delivery* (VRPPD), *vehicle routing*

problem with backhauls (VRPB), *split delivery vehicle roting problem* (SDVRP), *periodic vehicle routing problem*, dan *vehicle routing problem with time windows* (VRPTW).

VRP yang mempunyai tujuan untuk membentuk rute optimal dalam memenuhi permintaan pelanggan yang terdiri dari pelayanan jemput dan pelayanan antar dengan kendala kapasitas, *time windows*, *precedence*, dan *pairing* dikenal dengan istilah *pickup and delivery vehicle routing problem with times windows* (PDPTW). *Precedence* merupakan kendala dimana dalam suatu permintaan pelanggan, pelayanan jemput harus dilakukan sebelum pelayanan antar, sedangkan *pairing* merupakan kendala dimana pelayanan jemput dan pelayanan antar pada setiap rutenya harus dilayani oleh kendaraan yang sama (Dumas dkk, 2001).

Permasalahan PDPTW dapat diselesaikan dengan menggunakana algoritma yang bersifat eksak dan heuristik. Pada algoritma yang bersifat eksak, dilakukan pendekatan dengan menghitung setiap solusi yang mungkin sampai satu solusi terbaik diperoleh, sedangkan algoritma heuristik merupakan algoritma untuk menyelesaikan permasalahan dengan lebih menekankan pada performa komputasi sederhana sehingga mampu memberikan solusi yang mendekati optimal lebih cepat jika dibandingkan dengan algoritma eksak. *Dynamic programming* dan *column generation* merupakan contoh algoritma yang bersifat eksak, sedangkan contoh algoritma yang bersifat heuristik adalah algoritma *insertion heuristic*, *cyclic transfer*, *tabu search*, *genetic algorithm*, *simulated annealing*, dan *large neighborhood search.*

Terdapat banyak penelitian mengenai penyelesaian masalah PDPTW yang telah dilakukan baik menggunakan algoritma yang bersifat eksak maupun heuristik. Beberapa penelitian tersebut antara lain, yaitu Bent R., Hentenryk, P.V. (2003) yang menggunakan algoritma *simulated annealing* dan *large neighborhood search* pada masalah PDPTW. Fajar D. W. (2006) meneliti tentang penyelesaian masalah PDPTW dalam bentuk pemartisian himpunan dengan menggunakan teknik pembangkitan kolom, S. Ropke dan Psinger (2005) menggunakan algoritma *large neighborhood search* dalam masalah PDPTW. Fitri Karunia R (2008) melakukan penelitian tentang PDPTW dinamis atau *dynamic* PDPTW untuk *city courier* dengan pengembangan algoritma heuristik *tabu search*, serta Risya P. (2012) yang mengaplikasikan algoritma hibrida dua tahap pada masalah PDPTW. Hasil dari setiap penelitian tersebut membuktikan bahwa algoritma yang digunakan dapat menghasilkan rute yang lebih efektif daripada rute sebelumnya. Tidak banyak penelitian yang membandingkan efektivitas dari penggunaan algoritma-algoritma tersebut dalam menyelesaikan masalah PDPTW. Pada umumnya, setiap peneliti hanya memberikan satu algoritma yang disajikan dalam penelitiannya, sehingga orang awam tidak mengetahui algoritma mana yang efektif dalam meminimalkan total jarak tempuh kendaraan, banyaknya kendaraan yang digunakan, waktu tempuh kendaraan ataupun ketiganya.

Berdasarkan uraian di atas, peneliti tertarik untuk membandingkan efektivitas dari penggunaan dua algoritma heuristik dalam menyelesaikan masalah PDPTW yaitu algoritma *simulated annealing* dan *large neighborhood search*. Algoritma *simulated annealing* merupakan algoritma yang berdasar pada metode

probabilisik, yaitu metode penerimaan solusi baru berdasarkan probabilitas tertentu yang dikembangkan oleh Kirkpatrick *et al.* (1983) untuk menemukan nilai minimum global dari sebuah fungsi yang memiliki beberapa nilai lokal minimum. Algoritma *simulated annealing* memiliki kelebihan dalam menemukan solusi, yaitu dengan bergerak menuju solusi yang biayanya lebih besar atau tidak lebih baik dengan harapan pergerakan ini dapat mengeluarkan keadaan dari lokal minimum. Algoritma *large neighborhood search* merupakan algoritma pencarian solusi terbaik yang memanfaatkan metode *local search* berdasarkan lingkungan yang dibentuk dari solusi awal dengan tujuan untuk memperbaiki solusi yang telah ada dengan mengevaluasi nilai fungsi tujuan. Adapun pembentukan solusi awal yang dilakukan pada algoritma *simulated annealing* dan *large neighborhood search* dalam penelitian ini adalah dengan menggunakan algoritma *insertion heuristic.* Algoritma *insertion heuristic* merupakan algoritma yang membentuk solusi dengan memilih pelanggan pertama untuk masuk ke dalam rute dan dilanjutkan dengan menyisipkan pelanggan yang belum dilayani ke dalam rute tersebut, hingga tidak ada lagi pelanggan yang dapat disisipkan.

Pada penelitian ini akan dijelaskan mengenai penggunaan algoritma *simulated annealing* dan *large neighborhood search* dalam penyelesaian masalah PDPTW yang kemudian diimplementasikan pada tipe data masalah penelitian optimasi (*benchmark instances)* dengan menggunakan Matlab (*Matrix Laboratory*). Selanjutnya akan dianalisa efektivitas kedua algoritma tersebut yang diukur dari banyaknya rute dan total jarak tempuh kendaraan berdasarkan hasil implementasi yang diperoleh. Pada akhir penelitian, diberikan sebuah contoh

permasalahan PDPTW yang akan diselesaikan dengan menggunakan algoritma *simulated annealing* dan *large neighborhood search* untuk mengetahui efektivitas kedua algoritma tersebut pada masalah nyata.

B. Pembatasan Masalah

Batasan masalah pada penelitian ini adalah sebagai berikut.

1. Permasalahan *pickup and delivery vehicle routing problem with time windows* (PDPTW) yang digunakan dalam penelitian ini adalah permasalahan PDPTW dengan satu depot.
2. Data uji coba algoritma yang digunakan dalam penelitian ini adalah 100 tipe data *benchmark* yang dipilih secara acak dari Li & Lim (http://www.sintef.no/Projectweb/TOP/PDPTW/Li--Lim-benchmark/.) dan tipe data *benchmark* dari Benavent, dkk (http://www.mat.ucm.es/~gregoriotd/PDPLT.htm).
3. Data yang digunakan sebagai contoh permasalahan *pickup and delivery vehicle routing problem with time windows* (PDPTW) dalam penelitian ini adalah data dari Jogja Kurir Express pada tangal 31 Agustus 2014.

C. Rumusan Masalah

Berdasarkan latar belakang tersebut, maka permasalahan dalam penelitian ini dapat dirumuskan sebagai berikut.

1. Bagaimana penggunaan algoritma *simulated annealing* dan algoritma *large neighborhood search* dalam penyelesaian masalah *pickup and delivery vehicle routing problem with time windows* (PDPTW)?

2. Bagaimana efektivitas penggunaan algoritma *simulated annealing* dan algoritma *large neighborhood search* dalam penyelesaian masalah *pickup and delivery vehicle routing problem with time windows* (PDPTW)?

D. Tujuan Penelitian

Tujuan dari penelitian ini adalah

1. menjelaskan penggunaan algoritma *simulated annealing* dan algoritma *large neighborhood search* dalam penyelesaian masalah *pickup and delivery vehicle routing problem with time windows* (PDPTW), dan
2. mengetahui efektivitas penggunaan algoritma *simulated annealing* dan algoritma *large neighborhood search* dalam penyelesaian masalah *pickup and delivery vehicle routing problem with time windows* (PDPTW).

E. Manfaat Penelitian

Hasil penelitian ini diharapkan memiliki manfaat sebagai berikut:

1. menambah pengetahuan di bidang penelitian operasional dan ilmu matematika,
2. menambah pengetahuan peneliti tentang penggunaan dan efektivitas algoritma *simulated annealing* dan *large neighborhood search* dalam penyelesaian masalah *pickup and delivery vehicle routing problem with time windows* (PDPTW),

3. sebagai salah satu alternatif penggunaan algoritma dalam penentuan rute kendaraan, khusunya pada masalah masalah *pickup and delivery vehicle routing problem with time windows* (PDPTW), dan
4. menambah koleksi bahan pustaka yang bermanfaat bagi UNY pada umumnya, dan mahasiswa Matematika dan Ilmu Pengetahuan Alam pada khususnya.

BAB II
KAJIAN TEORI

Pada bab II ini akan dijelaskan mengenai beberapa teori tentang efektivitas, graf, *vehicle routing problem* (VRP), *pickup and delivery vehicle routing problem with time windows* (PDPTW), metode penyelesaian PDPTW, algoritma *insertion heuristic*, *simulated annealing*, dan *large neighborhood search*, serta *benchmark*, yang akan digunakan sebagai landasan dalam pembahasan selanjutnya.

A. Efektivitas

Efektivitas berasal dari kata efektif yang memiliki arti berhasil, tepat guna, atau sesuatu yang dilakukan berhasil dengan baik. Pius A. Partanto dan M. Dahlan Bahri dalam bukunya Kamus Ilmiah Populer (1994) mendefinisikan efektivitas sebagai ketepatgunaan, hasil guna, dan menunjang tujuan. Hidayat (1986) mendefinisikan efektivitas sebagai suatu ukuran yang menyatakan seberapa jauh target telah tercapai, dimana semakin besar persentase target yang dicapai, maka semakin tinggi efektivitasnya.

Secara umum, efektivitas dapat diartikan sebagai hal yang berhubungan dengan keberhasilan atau suatu ukuran yang menyatakan sejauh mana tujuan yang dicapai sesuai dengan tujuan yang telah ditentukan sebelumnya. Dalam hal ini, efektivitas menjadi unsur penting karena mampu memberikan gambaran mengenai keberhasilan dalam mencapai tujuan.

Pada skripsi ini, efektivitas diukur dari total jarak tempuh kendaraan dan banyaknya rute yang dihasilkan oleh kedua algoritma heuristik yang digunakan. Algoritma dikatakan efektif dalam mengurangi total jarak tempuh jika dapat menghasilkan total jarak tempuh yang lebih minimal dan dikatakan efektif dalam mengurangi banyakya rute jika menghasilkan jumlah rute yang lebih minimal dari algoritma yang dibandingkan.

B. Graf

Masalah PDPTW dapat direpresentasikan dalam sebuah graf. Simpul pada graf merepresentasikan lokasi pelanggan yang dituju, sedangkan rusuk merepresentasikan ruas jalan penghubung antar pelanggan ataupun antar depot dengan pelanggan. Oleh karena itu, pada subbab berikut akan diberikan pengertian graf, jenis-enis graf, dan keterhubungan graf.

1. Pengertian

Pengertian graf menurut Edgar G. Goodaire dan Michael M. Parmenter (1997) adalah kumpulan simpul (*vertexs* atau *nodes*) yang dihubungkan satu sama lain melalui busur (*edges*). Secara matematis, suatu graf G didefinisikan sebagai pasangan himpunan (V,E), dimana V adalah himpunan tidak kosong dari simpul (*vertexs* atau *nodes*), $V = \{v_1, v_2, \dots, v_n\}$, dan E adalah himpunan busur(*edges* atau *arcs*), $E = \{e_1, e_2, \dots, e_n\}$, yang menghubungkan sepasang simpul pada graf tersebut.

2. Jenis-jenis Graf

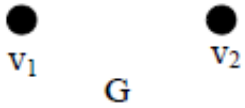
Graf dapat diklasifikasikan sesuai dengan kekhasan strukturnya (Edgar G. dan Michael M. P, 1997). Beberapa jenis graf disajikan sebagai berikut.

a. Graf sederhana (*simple graph*)

Graf sederhana adalah graf yang tidak memuat rusuk ganda dan gelang. Rusuk ganda adalah dua rusuk yang menghubungkan dua simpul yang sama. Gelang adalah rusuk yang menghubungkan suatu simpul dengan simpul itu sendiri. Beberapa graf sederhana dapat ditunjukkan sebagai berikut.

1) Graf nol

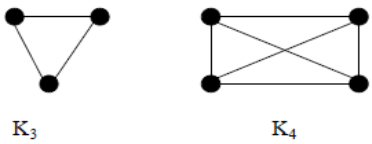
Graf nol adalah graf yang tidak memiliki rusuk atau himpunan rusuknya merupakan himpunan kosong. Gambar 2.1 berikut ini menunjukkan graf nol dengan dua buah simpul.



Gambar 2.1 Contoh Graf Nol dengan 2 Simpul

2) Graf lengkap

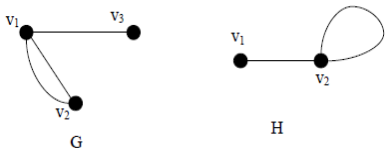
Graf lengkap adalah graf sederhana yang setiap pasang simpulnya saling berikatan. Notasi graf lengkap dengan n simpul adalah Kn. Contoh graf lengkap dengan n gasal (K_3) dan n genap (K_4) ditunjukkan oleh Gambar 2.2.



Gambar 2.2 Graf K_3 dan K_4

b. Graf tidak sederhana

Graf tidak sederhana adalah graf yang memiliki gelang dan rusuk ganda. Contoh graf tidak sederhana dapat dilihat pada Gambar 2. 3.

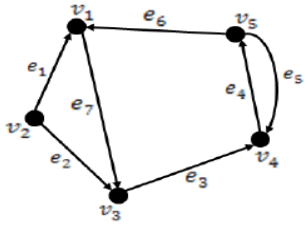


Gambar 2.3 Graf Tidak Sederhana G dan H

Pada Gambar 2.3, graf G memuat rusuk ganda dan pada graf H memuat gelang/ *loop*.

c. Graf berarah (*directed graph* atau *digraph*)

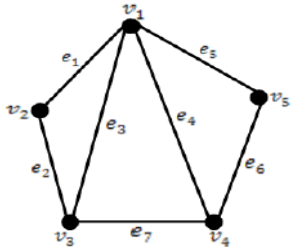
Graf berarah adalah graf yang setiap rusuknya memiliki orientasi arah. Contoh graf berarah dapat dilihat pada Gambar 2.4.



Gambar 2.4 Graf Berarah

d. Graf tidak berarah (*undirect graph*)

Graf tidak berarah adalah graf yang rusuknya tidak mempunyai orientasi arah. Contoh graf tidak berarah dapat dilihat pada Gambar 2.5.



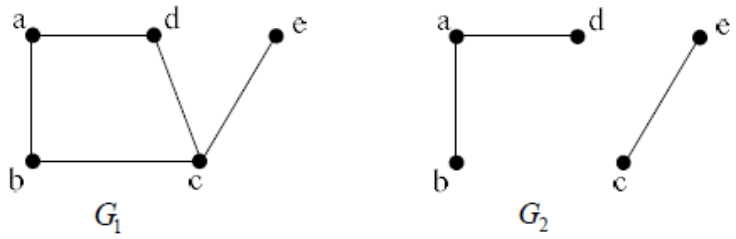
Gambar 2.5 Graf Tidak Berarah

3. Keterhubungan

Pada bagian ini akan dijelaskan keterhubungan suatu graf, pengertian jalan, lintasan, jalur/jejak, siklus dan sirkuit yang disertai dengan contoh-contoh untuk memperjelas definisi yang dimaksud.

Definisi 2.1

Misalkan *u* dan *v* adalah simpul-simpul dari suatu graf *G*, maka graf *G* dikatakan terhubung *(connected)* jika terdapat busur yang menghubungkan simpul *u* dan *v* di dalam *G*. Graf *G* dikatakan tidak terhubung *(disconnected)* jika simpul *u* dan *v* tidak terdapat busur yang menghubungkan.



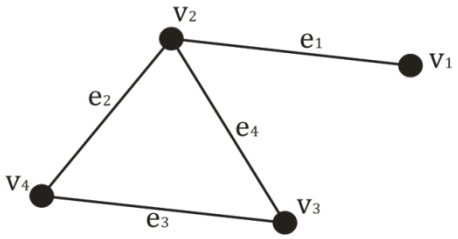
Gambar 2.6 Graf G_1 Terhubung dan G_2 Tidak Terhubung.

Definisi 2.2

Simpul-simpul v_1 dan v_2 disebut berdekatan (*adjacent*) atau v_1 berdekatan dengan v_2 jika ada suatu busur e, sedemikian sehingga $e = v_1v_2$. Busur e dikatakan menggabungkan (*join*) v_1 dan v_2.

Contoh:

Simpul v_3 pada graf berikut berdekatan *(adjacent)* dengan v_2 dan v_4, tetapi tidak berdekatan dengan v_1. Busur e_4 dikatakan menggabungkan *(join)* v_2 dan v_3, oleh karena $e_4 = v_2v_3$. Untuk hal yang sama $e_3 = v_3v_4$.



Gambar 2.7 Simpul v_1 Berdekatan *(Adjacent)* dengan v_2.

Definisi 2.3

Simpul v_1 dan v_2 insiden pada (*incident on*) busur e atau dikatakan busur e insiden dengan (*incident to)* simpul v_1 dan v_2 jika $e = v_1v_2$.

Contoh:

Pada Gambar 2.7, busur e_2 insiden dengan simpul v_2 dan v_4 karena $e_2 = v_2v_4$.

Definsi 2.4

Barisan simpul-simpul dan busur-busur pada graf *G* berselang-seling, yaitu:

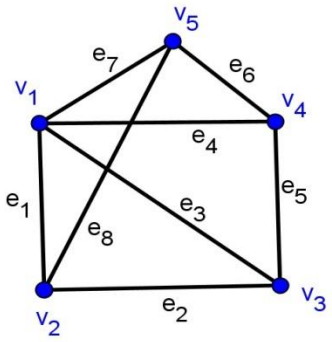
$$W: v_0, e_1, v_1, e_2, \dots, v_{n-1}, e_n, v_n \quad (n \geq 0)$$

yang dimulai dari suatu simpul dan berakhir pada suatu simpul sedemikian sehingga $e_i = v_{i-1}v_i$, untuk $i = 1,2,\dots,n$ (tiap busur insiden dengan tiap dua simpul pada barisan tersebut), dikatakan sebagai suatu jalan (*walk*) pada G.

Jalan pada graf dapat dinotasikan dengan $v_0v_1v_2,\dots,v_n$ dan dapat dikatakan sebagai jalan $v_0 - v_n$ yang bermakna barisan dimulai dari simpul awal v_0 dan berakhir di simpul akhir v_n. (Harary, 1969).

Contoh:

Diberikan Graf G_3 seperti pada gambar di bawah.



Gambar 2.8 Graf Sederhana G3

Misal diketahui bahwa W_1: $v_1, e_1, v_2, e_2, v_3, e_5, v_4, e_6, v_5$

berarti W_1 merupakan suatu jalan pada graf G. W_1 dapat ditulis lebih sederhana sebagai v_1, v_2, v_3, v_4, v_5.

Diberikan W_2: $v_2, v_5, v_4, v_3, v_4, v_1, v_4$

maka W_2 juga merupakan suatu jalan, walaupun busur e_5 dan e_4 dilewati lebih dari satu kali.

Definisi 2.5

Panjang (*length*) dari suatu jalan $v_0v_1, \ldots, v_n$ adalah banyaknya busur yang muncul (dilalui) sepanjang barisan.

Contoh:

Diberikan graf seperti pada Gambar 2.8.

Didefinisikan juga jalan W_1: v_1, v_2, v_3, v_4, v_5 dan W_2: $v_2, v_5, v_4, v_3, v_4, v_1, v_4$.

Jalan W_1 mempunyai panjang empat karena barisan melalui busur sebanyak empat buah. Jalan W_2 mempunyai panjang enam, karena melalui empat busur yang berbeda namun dua busur berulang sehingga panjangnya menjadi enam.

Definisi 2.6

Suatu barisan simpul-simpul dan rusuk-rusuk (jalan) dikatakan tertutup (*close*) apabila $v_0 = v_n$. Jika $v_0 \neq v_n$, maka dikatakan terbuka (*open*).

Contoh:

Diberikan graf seperti pada Gambar 2.8.

Jalan W_1: v_1, v_2, v_3, v_4, v_5 dan W_2: $v_2, v_5, v_4, v_3, v_4, v_1, v_4$.

Dimisalkan bahwa W_3: $v_1, v_2, v_3, v_1, v_5, v_4, v_1$, maka W_3 merupakan jalan tertutup, sedangkan W_1 dan W_2 merupakan jalan terbuka karena simpul awalnya tidak sama dengan simpul akhirnya.

Definisi 2.7

Lintasan (*path*) adalah jalan dengan semua simpul dalam barisannya berbeda.

Contoh:

Diberikan graf seperti pada Gambar 2.8.

Jalan W_1: v_1, v_2, v_3, v_4, v_5 dan W_2: $v_2, v_5, v_4, v_3, v_4, v_1, v_4$.

Berdasarkan definisi lintasan, maka W_1 merupakan suatu lintasan sedangkan W_2 bukan suatu lintasan. W_2 bukan lintasan karena ada simpul yang dilewati lebih dari satu kali, yaitu simpul v_4.

Definisi 2.8

Jalur/jejak (*trail*) adalah jalan dengan semua busur dalam barisannya berbeda.

Contoh:

Diberikan graf seperti pada Gambar 2.8, yaitu jalan $W_1: v_1, v_2, v_3, v_4, v_5$, $W_2: v_2, v_5, v_4, v_3, v_4, v_1, v_4$, dan $W_3: v_1, v_2, v_3, v_1, v_5, v_4, v_1$, maka W_1 dan W_3 merupakan suatu jalur sedangkan W_2 bukan merupakan suatu jalur karena barisan W_2 melewati busur yang sama, yaitu busur e_4 dan busur e_5.

Definisi 2.9

Siklus (*cycle*) adalah jalan tertutup (*closed walk*) dengan simpul tidak berulang, kecuali simpul awal sama dengan simpul akhir. Dengan kata lain, siklus adalah lintasan yang tertutup.

Contoh:

Diberikan graf seperti pada Gambar 2.8, maka v_1, v_2, v_3, v_4, v_5 dan $v_2, v_3, v_1, v_4, v_5, v_2$ merupakan siklus pada graf tersebut.

Definisi 2.10

Sirkuit (*circuit*) adalah jalan tertutup (*closed walk*) dengan busur tidak berulang atau dengan kata lain sirkuit adalah jalur yang tertutup.

Contoh:

Jika diberikan graf seperti pada Gambar 2.8, maka v_1, v_2, v_3, v_4, v_5 dan $v_2, v_3, v_1, v_4, v_5, v_2$ merupakan sirkuit graf tersebut.

C. *Vehicle Routing Problem*

Vehicle routing problem (VRP) merupakan masalah penentuan rute kendaraan yang memegang peranan penting dalam dunia industri yaitu pada masalah manajemen logistik dan transportasi. Yeun dkk (2008) mendefinisikan

VRP sebagai masalah penentuan rute optimal kendaraan dalam pendistribusian barang atau jasa dari satu atau lebih depot ke sejumlah pelanggan di lokasi yang berbeda dengan permintaan yang telah diketahui dan memenuhi sejumlah kendala. Tujuan umum VRP menurut Toth dan Vigo (2002) adalah

1. meminimalkan jarak dan biaya tetap yang berhubungan dengan penggunaan kendaraan,
2. meminimalkan jumlah kendaraan yang dibutuhkan untuk melayani permintaan seluruh pelanggan,
3. menyeimbangkan rute-rute dalam hal waktu perjalanan dan muatan kendaraan, dan
4. meminimalkan pinalti sebagai akibat dari pelayanan yang kurang memuaskan terhadap pelanggan, seperti keterlambatan pengiriman dan lain sebagainya.

Beberapa komponen beserta karakteristiknya yang terdapat dalam masalah VRP menurut Toth dan Vigo (2002), yaitu sebagai berikut.

1. Jaringan jalan

   Jaringan jalan biasanya direpresentasikan dalam sebuah graf. Jaringan jalan terdiri dari *edge* (rusuk) yang mempresentasikan bagian jalan yang digunakan, dan *vertex* (titik) yang mempresentasikan konsumen dan depot.

2. Konsumen

   Konsumen atau pelanggan direpresentasikan dengan *vertex* (titik). Setiap konsumen memiliki jumlah permintaan yang berbeda-beda yang dapat mempengaruhi lamanya waktu bongkar muat (*loading unloading*) barang.

Pada beberapa konsumen, biasanya terdapat *time windows* atau rentang waktu kapan konsumen tersebut dapat dilayani.

3. Depot

   Depot direpresentasikan oleh *vertex* (titik). Depot merupakan tempat awal dan akhir dari suatu rute kendaraan. Depot memiliki sejumlah kendaraan dengan jenis dan kapasitas tertentu yang dapat digunakan dalam mendistribusikan barang atau jasa pada jam operasional depot yang telah ditentukan (*time windows depot*).

4. Kendaraan

   Kendaraan yang digunakan dalam proses distribusi memiliki kapasitas yang membatasi permintaan konsumen, yaitu dimana jumlah permintaan konsumen tidak boleh melebihi kapasitas kendaraan tersebut. Selain itu, kendaraan juga memiliki biaya yang berhubungan dengan penggunaan kendaraan, baik yang meliputi biaya pengeluaran untuk bahan bakar maupun sewa kendaraan.

5. Pengemudi

   Pengemudi memiliki kendala seperti jam kerja harian, tambahan waktu lembur apabila diperlukan, jumlah dan jam istirahat, serta durasi maksimum perjalanan.

Dalam masalah penentuan rute kendaraan agar sesuai dengan tujuan yang telah ditentukan, ada beberapa kendala atau batasan yang harus dipenuhi VRP. Batasan-batasan yang harus dipenuhi menurut Kallehauge (2001), yaitu sebagai berikut.

1. Setiap konsumen atau pelanggan hanya dikunjungi satu kali oleh satu kendaraan.
2. Semua pelanggan harus dilayani sesuai dengan permintaannya masing-masing yang telah diketahui sebelumnya.
3. Kendaraan yang digunakan adalah seragam/homogen dan memiliki kapasitas tertentu, sehingga permintaan pelanggan pada setiap rute yang dilalui tidak boleh melebihi kapasitas kendaraan.
4. Setiap rute kendaraan berawal dari depot dan pada akhirnya juga harus kembali ke depot.

Penelitian mengenai VRP terus mengalami perkembangan sejak VRP pertama kali diperkenalkan oleh Dantzig dan Ramser (1959) melalui makalah mereka yang berjudul "*The Truck Dispatching Problem*". Dantzig dan Ramser meneliti bagaimana memperoleh rute optimal untuk truk tangki distribusi bensin dengan mengunakan pendekatan program linear. Pada tahun 1964, Clark dan Wright melakukan penelitian lanjutan dengan mengenalkan istilah depot sebagat tempat awal keberangkatan dan kembalinya kendaraan. Sejak itulah, VRP mulai dikenal dan terus berkembang dengan berbagai metode yang dipakai untuk memecahkan masalah VRP dan variasinya.

Berikut ini terdapat beberapa jenis atau variasi masalah utama dalam VRP menurut Toth dan Vigo (2002).

1. *Capacitated Vehicle Routing Problem* (CVRP)

   CVRP merupakan jenis VRP yang setiap kendaraannya memiliki kapasitas terbatas.

2. *Distance Constrained Vehicle Routing Problem* (DCVRP)

   DCVRP merupakan jenis VRP dengan kendala batasan panjang rute.

3. *Vehicle Routing Problem with Pick up and Delivery* (VRPPD)

   VRPPD merupakan jenis VRP dengan pelayanan jemput dan pelayanan antar dalam setiap permintaan pelanggan.

4. *Vehicle Routing Problem with Multiple Depot* (MDVRP)

   MDVRP merupakan jenis VRP yang memiliki banyak depot dalam melakukan pelayanan terhadap pelanggan.

5. *Split Delivery Vehicle Routing Problem* (SDVRP)

   SDVRP merupakan jenis VRP dimana pelayanan terhadap pelanggan dilakukan dengan menggunakan kendaraan yang berbeda-beda.

6. *Vehicle Routing Problem with Time Windows* (VRPTW)

   VRPTW merupakan jenis VRP dengan kendala kapasitas kendaraan dan batasan waktu (*time windows*) pada setiap pelanggan dan depot.

D. *Pickup and Delivery Vehicle Routing Problem with Time Windows* (PDPTW)

1. Pengertian

Pickup and delivery vehicle routing problem with time windows (PDPTW) adalah salah satu jenis VRP yang merupakan bentuk umum dari *vehicle routing problem with time windows (VRPTW).* PDPTW mempunyai tujuan membentuk rute optimal untuk memenuhi permintaan pelanggan, dimana setiap permintaan terdiri dari pelayanan jemput dan pelayanan antar dengan kendala kapasitas, *time windows, precedence,* dan *pairing.*

Kendala pertama pada PDPTW adalah kendala kapasitas. Kendala kapasitas yang dimaksud adalah bahwa setiap kendaraan memiliki kapasitas tertentu dan jika kapasitas kendaraan sudah penuh, maka kendaraan tidak dapat melayani pelanggan jemput selanjutnya. Namun, kendaraan dapat melakukan pelayanan jemput yang lain setelah melakukan pelayanan antar selama muatan yang ada pada kendaraan tersebut belum mencapai kapasitas maksimal.

Kendala berikutnya adalah kendala *time window* pada masing-masing pelanggan dan *time windows* pada depot. *Time window* pada masing-masing pelanggan $[a_i , b_i]$ adalah interval waktu yang ditentukan oleh masing-masing pelanggan bagi setiap kendaraan untuk dapat memulai pelayanan. Kendaraan dapat melakukan pelayanan di antara waktu awal pelanggan (a_i) dan waktu akhir pelanggan (b_i). Akan tetapi, kendaraan harus menunggu sampai waktu awal pelanggan dapat dilayani apabila kendaraan tersebut datang sebelum waktu awal pelanggan, sedangkan *time windows* pada depot $[a_0 , b_0]$ didefinisikan sebagai interval waktu yang menunjukkan waktu awal keberangkatan kendaraan dari depot dan waktu kembalinya kendaraan ke depot. Hal ini menunjukkan bahwa setiap kendaraan tidak boleh meninggalkan depot sebelum waktu awal depot (a_0) dimulai dan harus kembali ke depot sebelum waktu akhir (b_0) depot selesai. Terdapat dua jenis *time windows* pada PDPTW, yaitu *hard time windows* dan *soft time windows*. Pada *hard time windows*, kendaraan harus tiba di lokasi pelanggan sebelum waktu akhir pelanggan, sedangkan pada *soft time windows*, kendaraan dapat tiba di lokasi pelanggan setelah waktu akhir pelanggan dan dikenakan biaya tambahan (Solomon dkk, 2005).

Selanjutnya adalah kendala *precedence* dan kendala *pairing.* Kendala *precedence* adalah kendala yang dalam suatu permintaan pelanggannya, pelayanan jemput harus dilakukan sebelum pelayanan antar, sedangkan kendala *pairin*g adalah kendala dengan pelayanan jemput dan pelayanan antar harus dilayani oleh kendaraan yang sama (Dumas dkk, 2001). Sebagai contoh, pelanggan *i* menginginkan barangnya diantar ke pelanggan *j*, maka kendaraan harus datang ke pelanggan *i* terlebih dahulu untuk mengambil barang yang akan diantar. Kemudian setelah itu kendaraan dapat menuju ke pelanggan *j* sebagai tempat tujuan pengiriman barang (*precedence).* Kendaraan yang menjemput barang di pelanggan *i* dan mengantar barang ke pelanggan *j* adalah kendaraan yang sama (*pairing*).

2. Formulasi PDPTW

Berikut akan diberikan pendefinisian variabel dan model matematika untuk PDPTW yang mengacu pada Dumas Y., dkk (1991), Bent, R., dan Hentenryck P. V. (2003).

Masalah PDPTW dapat direpresentasikan dalam sebuah graf berarah. Simpul mewakili tiap lokasi pelanggan dan rusuk atau garis berarah sebagai ruas jalan penghubung antar pelanggan ataupun antar depot dengan pelanggan. Didefinisikan $G(N, A)$ adalah graf berarah yang mempresentasikan masalah PDPTW. Himpunan $P^+ = \{1, 2,3,...,n\}$ adalah himpunan simpul yang mewakili tiap lokasi pelanggan jemput. Himpunan $P^- = \{@1, @2, @3, \dots, @n\}$ adalah himpunan simpul yang mewakili tiap lokasi pelanggan antar, sedangakan $P = P^+ \cup P^-$ merupakan himpunan pelanggan jemput dan pelanggan antar. $N =$

$\{0, 1, 2, 3, \ldots, n, @1, @2, @3, \ldots, @n, @0\}$ merupakan himpunan yang anggotanya adalah himpunan *P* ditambah simpul 0 dan simpul $@0$ sebagai simpul depot. $A = \{(i, j) | i, j \in N, i \neq j\}$ merupakan himpunan rusuk atau garis berarah yang menghubungkan dua simpul yaitu ruas jalan penghubung antar pelanggan ataupun antara depot dengan pelanggan.

Misalkan terdapat n pelanggan dan jika i adalah simpul untuk pelanggan jemput, maka $@i$ adalah simpul untuk pelanggan antar, sedangkan 0 dan $@0$ adalah simpul untuk depot. Setiap pelanggan i memiliki sejumlah permintaan q_i, yang harus diantar dari pelanggan i ke pelanggan $@i$, dan pelanggan $@i$ menerima permintaan dari pelanggan i yang dinyatakan dengan $q_{@i}$ yang nilainya adalah -$q_{i..}$ Selanjutnya $[a_i, b_i]$ adalah *time windows* pelayanan jemput ke pelanggan i, $[a_{@i}, b_{@i}]$ adalah *time windows* pelayanan antar untuk pelanggan i, dan $[a_0, b_0]$ adalah *time windows* kendaraan berangkat dari depot, sedangkan $[a_{@0}, b_{@0}]$ adalah *time windows* kendaraan untuk kembali ke depot. Himpunan $K = \{1, 2, 3, \ldots, k\}$ adalah himpunan kendaraan yang digunakan pada rute untuk melayani pelanggan dengan kendaraan sebanyak *k*.

Diasumsikan nilai kapasitas untuk setiap kendaraan $k \in K$ adalah homogen yang dinyatakan dengan Q. Kemudian, untuk setiap rusuk berarah $(i, j) \in N$ memiliki waktu tempuh t_{ij}^k yang menyatakan waktu tempuh perjalanan dari pelanggan i ke pelanggan j oleh kendaraan k, c_{ij}^k menyatakan biaya perjalanan dari pelanggan i ke pelanggan j oleh kendaraan k, s_i^k menyatakan waktu pelayanan pada pelanggan i oleh kendaraan k, d_{ij}^k yang menyatakan jarak tempuh

kendaraan *k* dari pelanggan *i* ke pelanggan *j*, dan w_i^k yang menyatakan waktu sampainya kendaraan *k* ke pelanggan *i*.

Pada penelitian ini, permasalahan PDPTW dalam menentukan sejumlah rute kendaraan memenuhi kondisi sebagai berikut, (1) terdapat satu depot dan sejumlah kendaraan yang seragam dengan kapasitas Q, (2) setiap kendaraan memulai rute perjalanan dari depot dan kembali lagi ke depot dengan tidak melanggar *time windows* depot, (3) dalam setiap rute, pelayanan jemput harus dilakukan sebelum pelayanan antar, (4) setiap pelanggan hanya dikunjungi satu kali oleh satu kendaraan yang sama, (5) *time windows* yang digunakan adalah *hard time windows*, yaitu dimana kendaraan harus tiba di lokasi pelanggan sebelum waktu akhir pelanggan, (6) muatan atau kapasitas total pelanggan pada setiap rute tidak boleh melebihi kapasitas kendaraan Q, (7) kecepatan kendaraan adalah konstan dan masalah kemacetan ataupun gangguan lainnya selama perjalanan diabaikan.

Permasalahan PDPTW tersebut kemudian diformulasikan ke dalam model matematika dengan tujuan meminimumkan biaya total perjalanan kendaraan untuk melayani seluruh pelanggan. jika Z merupakan fungsi tujuan maka,

meminimumkan

$$Z = \sum_{k \in K} \sum_{i,j \in V} c_{ij} \; X_{ij}^k \tag{2.1}$$

dengan variabel keputusan,

1. Variabel X_{ij}^{k} , $k \in K, i, j \in N, i \neq j$.

   Variabel X_{ij}^{k} mempresentasikan ada atau tidaknya perjalanan dari pelanggan i ke pelanggan j oleh kendaraan *k.*

$$X_{ij}^{k} = \begin{cases} 1, & jika\ terdapat\ perjalanan\ kendaraan\ \ dari\ i\ ke\ j\ oleh\ kendaraan\ k \\ 0, & jika\ tidak\ ada\ perjalanan\ kendaraan\ dari\ i\ ke\ j\ oleh\ kendaraan\ k \end{cases} \quad (2.2)$$

2. Variabel $T_i^k, i \in P$, $T_0^k dan\, T_{@0}^k$, $k \in K$

   Variabel T_i^k menyatakan waktu dimulainya pelayanan pada pelanggan *i* oleh kendaraan *k,* sedangkan T_0^k adalah waktu saat kendaraan *k* meninggalkan depot, dan $T_{@0}^k$ adalah waktu saat kendaraan *k* kembali ke depot.

3. Variabel Y_i^k, $i \in P$, $k \in K$

   Variabel Y_i^k menyatakan muatan atau kapasitas total dalam kendaraan *k* setelah meninggalkan pelanggan *i*. Diasumsikan kendaraan memiliki muatan atau kapasitas kosong saat berangkat dari depot atau $Y_0^k = 0$.

Kendala permasalahan PDPTW adalah sebagai berikut.

1. Setiap kendaraan memulai rute perjalanan dari depot dan akan kembali lagi ke depot.

$$\sum_{j \in P} X_{o,j}^{k} = 1\,, \qquad \forall k \in K. \quad (2.3)$$

$$\sum_{i \in P} X_{i,@0}^{k} = 1\,, \qquad \forall k \in K\,. \quad (2.4)$$

2. Setiap pelanggan hanya dilayani tepat satu kali dengan kendaraan yang sama.

$$\sum_{k\in K}\sum_{j\in P\cup\{@0\}} X_{ij}^{k} = 1, \qquad \forall i \in P. \tag{2.5}$$

$$\sum_{j\in N} X_{ij}^{k} - \sum_{j\in N} X_{j,@i}^{k} = 0, \qquad \forall i \in P^{+}, \forall k \in K. \tag{2.6}$$

3. Kendaraan yang telah mengunjungi seorang pelanggan harus meninggalkan pelanggan tersebut menuju pelanggan lain.

$$\sum_{i\in P\cup\{0\}} X_{ij}^{k} - \sum_{i\in P\cup\{@0\}} X_{ji}^{k} = 0, \qquad \forall j \in P\ , \forall k \in K. \tag{2.7}$$

4. Kendala *precedence*, yaitu kendala dimana dalam suatu permintaan pelanggan, pelanggan jemput dikunjungi terlebih dahulu sebelum pelanggan antar.

$$T_i^k + s_i^k + t_{i,@i}^k \le T_{@i}^k\ , \qquad \forall i \in P^{+}. \tag{2.8}$$

5. Jika terdapat perjalanan dari pelanggan *i* ke pelanggan *j*, maka waktu dimulainya pelayanan di pelanggan *j* pasti lebih dari atau sama dengan waktu kendaraan *k* untuk memulai pelayanan pada pelanggan *i* ditambah waktu pelayanan pelanggan *i* ditambah dan waktu tempuh perjalanan dari pelanggan *i* ke pelanggan *j*.

$$X_{ij}^{k} = 1 \rightarrow\ T_i^k + s_i^k + t_{i,j}^k \le T_j^k\ , \qquad \forall i,j\ \in N, \forall k \in K. \tag{2.9}$$

6. Kendala *time windows* pelanggan (2.10) dan *time windows* depot (2.11), yaitu waktu kendaraan *k* untuk memulai pelayanan harus berada pada selang waktu atau *time windows* yang telah ditentukan.

$$a_i\ \le\ T_i^k\ \le b_i\ , \qquad i \in P, \qquad k \in K. \tag{2.10}$$

$$a_0\ \le\ T_0^k\ \le b_0\ , \qquad k \in K. \tag{2.11}$$

7. Muatan kendaraan k setelah meninggalkan pelanggan j adalah muatan kendaraan k setelah meninggalkan pelanggan i ditambah dengan muatan yang diambil pada pelanggan j apabila j adalah pelanggan jemput (2.12). Jika j adalah pelanggan antar, maka muatan kendaraan k setelah meninggalkan pelanggan i dikurangi dengan muatan yang diantar pada pelanggan j (2.13).

$$X_{ij}^{k} = 1 \rightarrow Y_i^k + q_j = Y_j^k \ , \quad i \in P \ , \ j \in P^+, \ k \in K. \qquad (2.12)$$

$$X_{ij}^{k} = 1 \rightarrow Y_i^k - q_j = Y_j^k \ , \quad i \in P \ , j \in P^-, k \in K. \qquad (2.13)$$

8. Muatan atau kapasitas kendaraan k setelah meninggalkan pelanggan antar $@i$ lebih besar atau sama dengan 0 dan lebih kecil atau sama dengan kapasitas kendaraan k setelah dikurangi muatan yang harus diantar dari pelanggan i ke pelanggan $@i$.

$$0 \leq Y_{@i}^{k} \leq Q - q_i \ , \quad @i \in P^-, \qquad k \in K. \qquad (2.14)$$

9. Kendala kapasitas kendaraan, yaitu kapasitas yang ada di dalam kendaraan k setelah meninggalkan pelanggan tidak boleh melebihi kapasitas kendaraan k.

$$Y_0^k = 0, \qquad q_i \leq Y_i^k \leq Q \ , \qquad i \in P^+ \ . \qquad (2.15)$$

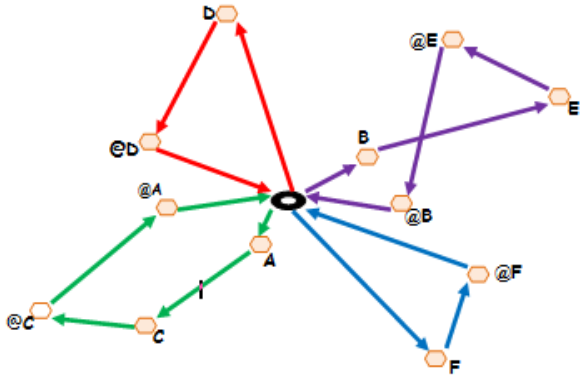
10. Variabel keputusan yang digunakan merupakan bilangan biner

$$X_{ij}^{k} \in \{0,1\} \qquad \forall \, i,j \in N, \qquad \forall \, k \in K. \qquad (2.16)$$

3. Representasi Solusi

Solusi layak formulasi PDPTW yang telah dijelaskan sebelumnya di atas dinotasikan dengan σ yaitu himpunan rute kendaraan yang memiliki total biaya perjalanan minimal dengan memenuhi semua kendala yang diberikan. Himpunan σ dapat ditulis sebagai berikut, σ = { rute 1, rute 2, ... , rute n}.

Solusi PDPTW dapat digambarkan dalam bentuk graf yang setiap rute perjalanan merupakan lintasan yang berawal dari satu simpul (depot), kemudian menghubungkan simpul-simpul pelanggan dalam rute sesuai dengan urutan kunjungan hingga kembali lagi ke simpul awal. Ilustrasi mengenai contoh solusi layak dari masalah PDPTW dapat dilihat pada Gambar 2.9.



Gambar 2.9 Contoh Solusi Layak PDPTW

Pada Gambar 2.9, terdapat 6 pelanggan jemput yaitu A, B, C, D, E, dan F serta 6 pelanggan antar yaitu, $@A$, $@B$, $@C$, $@D$, $@E$, dan $@F$. Dengan penggunaan algoritma eksak ataupun heuristik didapatkan sebuah solusi yang terdiri dari empat rute perjalanan kendaraan yang berawal dan berakhir di depot serta memenuhi semua kendala pada masalah PDPTW. Rute pertama terdiri atas pelanggan B, E, $@E$, dan $@B$, rute kedua terdiri atas pelanggan F dan $@F$, rute ketiga terdiri atas pelanggan A, C, $@C$, dan $@A$, serta rute keempat yang terdiri atas pelanggan D dan $@D$. Dengan demikian, representasi solusi dari masalah PDPTW tersebut adalah $\sigma = \{0 - B - E - @E - @B - 0, 0 - F - @F - 0, 0 - A - C - @C - @A - 0, 0 - D - @D - 0\}$, dengan total jarak tempuh kendaraan = $d_{0B} + d_{BE} + d_{E@E} + d_{@E@B} + d_{@B0} + d_{0F} + d_{F@F} + d_{@F0} + d_{0A} + d_{AC} + d_{C@C} + d_{C@C} + d_{@C@A} + d_{@A0} + d_{0D} + d_{D@D} + d_{@D0}$.

E. Metode Penyelesaian PDPTW

Berdasarkan Mitrovic Minic (1998) dan Dridi dkk (2011), algoritma yang digunakan untuk menyelesaikan masalah PDPTW terdiri atas algoritma yang bersifat eksak dan heuristik.

1. Algoritma Eksak

Penyelesaian masalah PDPTW menggunakan algoritma eksak dilakukan dengan menghitung setiap solusi yang mungkin sampai ditemukan solusi terbaik. Hal ini akan menghabiskan waktu yang cukup lama karena PDPTW termasuk dalam permasalahan NP-hard (*non polynomial-hard*), yaitu waktu yang dibutuhkan untuk mencari solusi permasalahan akan bergerak secara eksponensial dengan semakin rumitnya permasalahan. Contoh algoritma yang bersifat eksak yaitu, *dinamic programming* dan *column generation.*

2. Algoritma Heuristik

Heuristik adalah suatu metode untuk menyelesaikan permasalahan dengan lebih menekankan pada performa komputasi sederhana. Algoritma heuristik merupakan algoritma yang mampu memberikan solusi yang mendekati optimal lebih cepat jika dibandingkan dengan algoritma eksak. Algoritma heuristik dibagi ke dalam tiga macam, yaitu *construction heuristic*, *improvement heuristic*, dan *metaheuristic*. Algoritma yang termasuk *construction heuristic* yaitu *clustering, mini-clustering,* dan *insertion*. Algoritma yang termasuk *improvement heuristic* yaitu *local search, variable dept arc exchange*, dan *cyclyc transfer*, sedangkan algoritma yang termasuk *metaheuristic*, yaitu *large neighborhood search, tabu*

search, ant colony system, differential evolution, genetic algorithm, dan *simulated annealing*.

Algoritma eksak secara konsisten dapat menghasilkan kualitas solusi yang lebih baik jika dibandingkan dengan algoritma heuristik meskipun lebih memakan waktu yang lebih lama. Namun, kesederhanaan algoritma heuristik dalam penggunaannya, membuat algoritma tersebut tetap menjadi algoritma populer untuk digunakan dalam penyelesaian masalah.

Pada penelitian ini, penyelesaian masalah PDPTW dilakukan dengan menggunakan algoritma heuristik yaitu algoritma *simulated annealing* dan *large neighborhood search* dengan *insertion heuristic* sebagai solusi awal. Ketiga algoritma tersebut akan dijelaskan pada subbab berikutnya.

F. Algoritma *Insertion Heuristic*, *Simulated Annealing*, dan *Large Neighborhood Search*

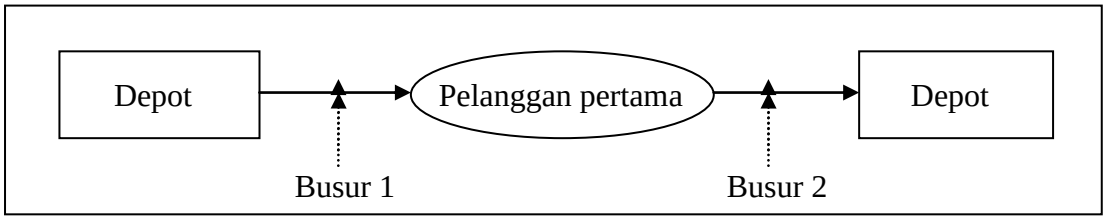
1. *Insertion Heuristic*

Algoritma *insertion heuristic* merupakan algoritma yang populer dalam menyelesaikan masalah penentuan rute kendaraan (Rozenkrantz, 1977). Prinsip dasar dari algoritma *insertion heuristic* adalah dengan menyisipkan pelanggan di antara busur penyisipan berupa lintasan penghubung pada rute yang dibentuk agar diperoleh hasil maksimal.

Algoritma *insertion heuristic* membentuk solusi layak berupa himpunan rute yang dimulai dengan memilih pelanggan pertama (*seed customer*) untuk masuk ke dalam rute. Pemilihan pelanggan pertama dapat dilakukan dengan berdasar pada jarak terjauh pelanggan dari depot atau pelanggan mana yang mendesak harus

segera dilayani (Joubert dan Classen, 2006). Selanjutnya dilakukan penyisipan satu pelanggan berikutnya atau pelanggan yang belum masuk ke dalam rute diantara dua pelanggan lain pada setiap iterasinya.

Dua hal yang harus ditentukan dalam setiap perulangan (iterasi) algoritma *insertion heuristic* adalah pelanggan mana yang akan disisipkan dan pada bagian mana pelanggan tersebut akan disisipkan. Penyisipan pelanggan berikutnya atau pelanggan yang belum masuk ke dalam rute dapat dilihat pada Gambar 2.4 berikut.



Gambar 2.10 Penyisipan Pelanggan Berikutnya pada *Insertion Heuristic*

Pada Gambar 2.10, terlihat bahwa pelanggan berikutnya dapat disisipkan pada busur 1 dan busur 2 yang ada dalam rute saat itu. Selanjutnya kelayakan diperiksa untuk semua kendala yang ada. Pelanggan pada rute yang memberikan biaya penyisipan paling kecil dan layak yang akan dipilih. Prosedur ini terus berulang hingga tidak ada lagi pelanggan yang dapat disisipkan ke dalam rute.

Berdasarkan Solomon (1987), terdapat tiga kriteria dalam menghitung biaya penyisipan pelanggan pada algoritma *insertion heuristic*, yaitu sebagai berikut.

1. Kriteria pertama

$$c_1 (i, u, j) = \alpha_1 c_{11} (i, u, j) + \alpha_2 c_{12} (i, u, j) , \quad (2.17)$$

$$\alpha_1 + \alpha_2 = 1 ; \alpha_1 \geq 0, \alpha_2 \geq 0.$$

$$c_2 (i, u, j) = \lambda d_{0u} - c_1 (i, u, j) , \lambda \geq 0. \quad (2.18)$$

Nilai $c_1\ (i,\ u,\ j)$ merupakan besarnya biaya penyisipan saat pealnggan *u* disisipkan diantara pelanggan *i* dan j. Nilai $c_2\ (i,\ u,\ j)$ merupakan penghematan jarak tempuh ketika menempatkan pelanggan *u* ke dalam rute baru. Nilai $c_{11}\ (i,\ u,\ j)$ merupakan penambahan jarak yang dihasilkan jika pelanggan *u* disisipkan di antara pelanggan *i* dan pelanggan *j*. Nilai $c_{11}(i,u,j)$ dapat dihitung menggunakan rumus berikut.

$$c_{11}\ (i,\ u,\ j\ ) = d_{iu} + d_{uj} - \mu\ d_{ij}\ . \tag{2.19}$$

dengan $\mu \geq 0$, dan d_{iu} , d_{uj} , dan d_{ij} masing-masing adalah jarak antara pelanggan *i* dengan pelanggan *u*, jarak antar pelanggan *u* dengan pelanggan *j*, dan jarak antar pelanggan *i* dengan pelanggan *j*.

Nilai $c_{12}\ (i,\ u,\ j)$ merupakan pergeseran waktu dimulainya pelayanan pada pelanggan *j* saat pelanggan *u* disisipkan antara pelanggan *i* dan pelanggan *j*. Nilai $c_{12}\ (i,\ u,\ j)$ dapat dihitung menggunakan rumus berikut.

$$c_{12}\ (i,\ u,\ j\ ) = T'_j\ - T_{j.} \tag{2.20}$$

dengan T'_j adalah waktu dimulainya pelayanan pada *j* saat pelanggan *u* berada dalam rute, dan T_j adalah waktu dimulainya pelayanan pada pelanggan *j*.

Kriteria pertama ini memiliki tujuan untuk mendapatkan keuntungan maksimum dengan menyisipkan pelanggan ke dalam rute yang telah ada daripada menyisipkannya ke dalam rute yang baru.

2. Kriteria kedua

$$c_1\ (i,\ u,\ j\ ) = \alpha_1\ c_{11}\ (i,\ u,\ j\ ) + \alpha_2\ c_{12}\ (i,\ u,\ j\ )\ , \tag{2.21}$$

$$\alpha_1 + \alpha_2 = 1\ ;\ \alpha_1 \geq 0,\ \alpha_2 \geq 0.$$

$$c_2\ (i,\ u,\ j\ ) = \beta_1 R_d(u) + \beta_2 R_t(u),\ \beta_1 + \beta_2 = 1,\ \beta_1 \geq 0,\ \beta_2 > 0. \tag{2.22}$$

Pada kriteria kedua, nilai $c_{11}(i, u, j)$ dan $c_1(i, u, j)$ memiliki definisi yang sama dengan kriteria pertama. Akan tetapi, nilai $c_{12}(i, u, j)$ pada kriteria ini diperoleh dari $R_d(u)$ yang menyatakan total jarak pada rute dan dan $Rt(u)$ yang menyatakan total waktu tempuh kendaraan pada rute setelah pelanggan u masuk ke dalam rute.

Kriteria kedua ini memiliki tujuan untuk memilih pelanggan yang biaya penyisipannya dapat meminimalkan total jarak dan waktu.

3. Kriteria ketiga

$$c_1(i, u, j) = \alpha_1 c_{11}(i, u, j) + \alpha_2 c_{12}(i, u, j) + \alpha_3 c_{13}(i, u, j), \quad (2.23)$$

$$\alpha_1 + \alpha_2 + \alpha_3 = 1 ; \alpha_1 \geq 0, \alpha_2 \geq 0, \alpha_3 \geq 0.$$

$$c_2(i, u, j) = c_1(i, u, j). \quad (2.24)$$

Pada kriteria ketiga, nilai $c_{11}(i, u, j)$ dan $c_{12}(i, u, j)$ memiliki definisi yang sama seperti yang telah didefenisikan sebelumnya di kriteria pertama. Sedangkan, nilai $c_{13}(i, u, j)$ merupakan interval waktu antara dimulainya pelayanan pada pelanggan u dan waktu terakhir kendaraan dapat melakukan pelayanan. Nilai $c_{13}(i, u, j)$ dapat dihitung dengan menggunakan rumus berikut.

$$c_{13}(i, u, j) = b_u - T_u. \quad (2.25)$$

dengan b_u adalah waktu akhir pelayanan pada pelanggan u dan T_u dalah waktu dimulainya pelayanan pada pelanggan u.

Kriteria ketiga ini menimbang aspek pelanggan jemput yang memiliki jarak terjauh dari depot dan keadaan dari pelanggan mana yang mendesak untuk segera dilayani.

2. *Simulated Annealing*

Algortima *simulated annealing* pertama kali diperkenalkan oleh Metropolis *et al*. pada tahun 1959. Algoritma ini diadaptasi dari proses *annealing* pada pembuatan kristal suatu material, yaitu proses pendinginan suatu benda padat sehingga strukturnya membeku pada suatu energi minimum (Betsimas dan Tsiklis, 1993). Pada proses pembuatan kristal suatu material, dilakukan pemanasan hingga mencapai satu titik tertentu. Dalam keadaan ini, atom-atom akan bergerak bebas dengan tingkat energi yang tinggi. Kemudian, secara perlahan suhu diturunkan dengan harapan energi dapat berkurang menuju ke suatu tingkatan yang relatif rendah. Semakin lambat laju pendinginan, akan semakin rendah pula energi yang dicapai oleh sistem. Dengan demikian, atom-atom tersebut diharapkan akan berada dalam posisi optimum dengan energi yang minimum.

Algoritma *simulated annealing* dapat dipandang sebagai algoritma *local search* yang terkadang bergerak menuju solusi dengan biaya lebih besar atau solusi yang tidak lebih baik dengan harapan pergerakan ini dapat mengeluarkan keadaan dari titik mimimum lokal (Betsimas dan Tsiklis, 1993). Kemampuan untuk menerima solusi yang buruk atau tidak lebih baik pada waktu-waktu tertentu inilah yang membedakan algoritma *simulated annealing* dari algoritma *local search* biasa. Penerimaan solusi pada keadaan tersebut didasarkan atas sebuah metode probabilisik yang dikembangkan oleh Kirkpatrick *et al.* pada tahun 1983, yaitu penerimaan solusi baru berdasarkan probabilitas tertentu untuk menemukan nilai minimum global dari sebuah fungsi yang memiliki beberapa nilai lokal minimal.

Tiga komponen yang perlu diperhatikan dalam algoritma *simulated annealing* menurut Tospornsampan (2007) adalah sebagai berikut.

1. Proses *Annealaing*

   Proses *annealing* merupakan proses utama pada algoritma *simulated annealing* yang bertujuan agar sistem tidak terjebak dalam keadaan lokal minimum. Proses ini bergantung pada parameter berikut.

   a. Suhu awal

      Suhu awal H_0 dipilih setinggi mungkin untuk memperluas penerimaan terhadap solusi baru.

   b. Jumlah iterasi pada tiap suhu

      Penurunan suhu akan dilakukan apabila telah mencapai jumlah iterasi tertentu, L_t. Jumlah iterasi pada setiap suhu tersebut dapat bernilai konstan ataupun berubah-ubah.

   c. Pemilihan parameter untuk menurunkan suhu

      Penurunan suhu dilakukan dengan menggunakan parameter α yang disebut *cooling rate* atau parameter penurunan suhu yang nilainya berkisar antara 0 dan 1. Proses penurunan suhu secara perlahan dapat dituliskan sebagai berikut.

$$H_t = \alpha H_{t-1}, \tag{2.26}$$

      dengan H_t adalah suhu saat t sedangkan H_{t-1} adalah suhu saat $t-1$.

2. Penyusunan Ulang

   Penyusunan ulang atau pembentukan solusi lingkungan dilakukan secara acak yaitu dengan mengubah solusi yang ada dengan solusi yang baru. Prosedur ini dilakukan dengan berbagai cara tergantung pada jenis permasalahnnya.

3. Penghentian Algoritma

   Dalam proses penghentian algoritma, diperlukan suatu kriteria yang ditentukan sejak awal proses. Kriteria tersebut dapat berupa suhu minimum, yaitu dimana proses akan berhenti ketika suhu mencapai suhu minimum tersebut. Kriteria yang lain dapat berupa banyaknya iterasi, yaitu dimana proses akan berhenti apabila tidak ada solusi baru yang dapat diterima hingga mencapai iterasi.

Langkah-langkah algoritma *simulated annealing* secara umum, yaitu sebagai berikut.

1. Tentukan solusi awal σ.
2. Tentukan suhu awal H_0, suhu akhir H_t, *cooling rate* α, dan jumlah iterasi L_t.
3. Tetapkan suhu awal H_0 sebagai pengontrol apakah solusi baru diterima atau tidak diterima.
4. Bentuk solusi lingkungan, σ'.

   Pembentukan solusi lingkungan σ' dilakukan dengan menyusun ulang solusi awal yang telah ditentukan sebelumnya secara acak.

5. Hitung selisih nilai fungsi objektif $(\Delta\sigma)$ menggunakan rumus berikut:

$$\Delta\sigma = e(\sigma') - e(\sigma) \qquad (2.27)$$

dengan $e(\sigma')$ adalah nilai fungsi dari solusi lingkungan σ', dan $e(\sigma)$ adalah nilai fungsi dari solusi sekarang σ.

a. Jika $\Delta\sigma < 0$, maka solusi lingkungan diterima sebagai solusi baru

b. Jika $\Delta\sigma > 0$, maka ada dua kemungkinan.

1) Solusi lingkungan diterima

Solusi lingkungan diterima jika nilai dari suatu bilangan random P antar 0 dan 1 lebih kecil dari $exp(-\Delta\sigma/H)$, dengan H adalah suhu pada saat ini atau suhu awal yang menjadi pengontrol apakah solusi baru akan diterima atau tidak.

2) Solusi lingkungan tidak diterima

Jika nilai dari suatu bilangan random P antar 0 dan 1 tidak lebih kecil dari $exp(-\Delta\sigma/H)$, maka solusi sekarang tidak berubah atau dengan kata lain solusi lingkungan tidak diterima sebagai solusi baru.

6. Lakukan pengecekkan apakah iterasi sudah mencapai iterasi L_t .

a. Jika belum mencapai iterasi L_t, maka lakukan penyusunan ulang solusi lingkungan layak, yaitu dengan mengulangi langkah 4 dan 5 hingga mencapai iterasi L_t yang ditentukan.

b. Jika telah mencapai iterasi L_t, lanjut ke langkah 7.

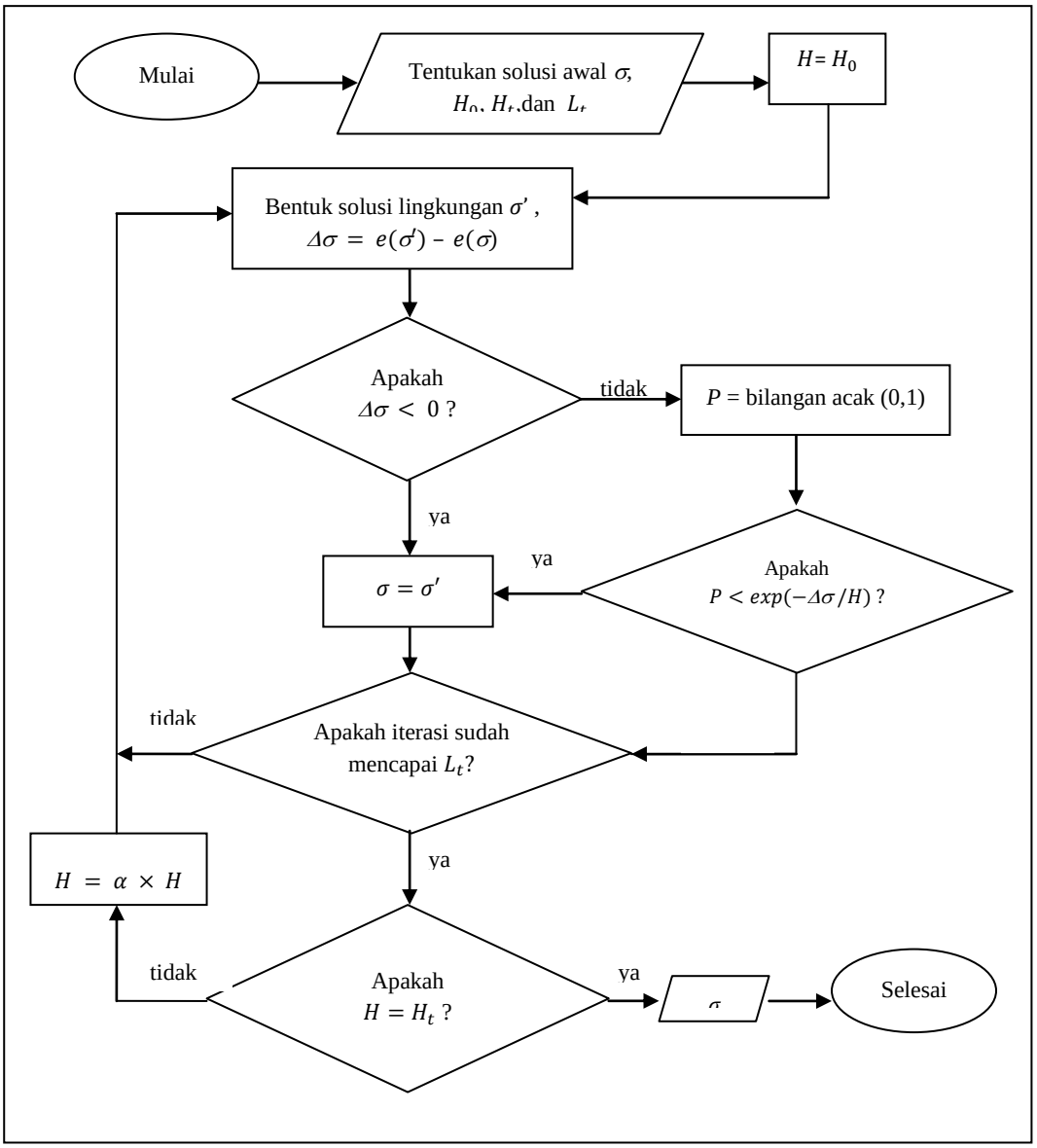
7. Lakukan pengecekkan apakah suhu telah mencapai suhu minimum.

a. Jika suhu belum mencapai suhu minimum, lakukan penurunan suhu H menggunakan suatu parameter *cooling rate* α dan ulangi langkah 4, 5, dan 6 secara berulang hingga mencapai suhu minimum.

b. Jika telah mencapai suhu minimum, maka lanjut ke langkah 8.

8. Proses dari algoritma *simulated annealing* telah selesai dan didapatkan solusi akhir σ.

Langkah-langkah dari algoritma *simulated annealing* secara umum di atas, dapat disajikan dalam digram alir berikut.



Keterangan : σ = solusi, H_0 = suhu awal, H_t = suhu akhir, dan L_t = iterasi tiap suhu, α = *cooling rate*
σ' = solusi lingkungan dari σ, $e(\sigma)$ = nilai fungsi objektif σ

Gambar 2.11 Diagram Alir Algoritma *Simulated Annealing*

3. *Large Neighborhood Search*

Algoritma *large neighborhood search* pertama kali diperkenalkan oleh Shaw pada tahun 1998 (Pisinger dan Ropke, 2010). Algoritma ini merupakan algoritma pencarian solusi terbaik yang memanfaatkan metode *local search*, yaitu metode pencarian solusi bedasarkan lingkungan dari suatu solusi awal.

Pada algoritma *large neighborhood search*, terdapat tiga tahapan utama dalam penyelesaian masalah VRP.

1. Pembentukan Solusi Awal

   Pembentukan solusi awal dapat dilakukan secara acak ataupun dengan menggunakan algoritma heuristik.

2. Pembentukan Solusi Lingkungan

   Pembentukan solusi lingkungan dilakukan dengan metode penghapusan dan metode perbaikan (Pisinger dan Ropke, 2010). Metode penghapusan akan merusak atau mengubah solusi yang telah ada, sedangkan metode perbaikan akan membangun kembali solusi yang telah diubah oleh metode penghapusan. Solusi lingkungan $N(\sigma')$ dari σ merupakan himpunan solusi yang didapatkan dengan dengan metode penghapusan dan perbaikan.

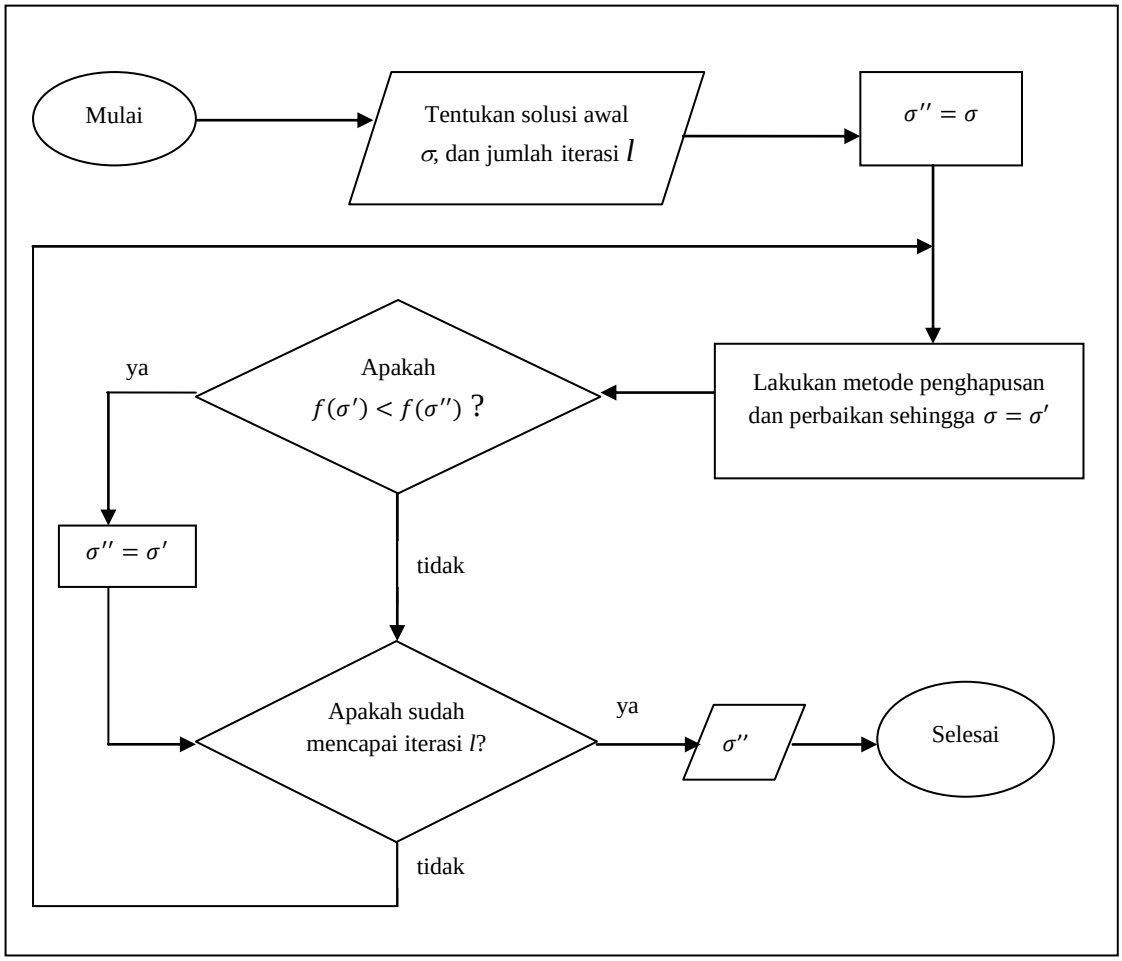
3. Penghitungan Nilai Fungsi Evaluasi

   Penghitungan nilai fungsi evaluasi dilakukan dengan membandingkan solusi awal dengan solusi lingkungan yang telah dievaluasi pada fungsi tujuan hingga mencapai solusi terbaik.

Langkah-langkah algoritma *large neighborhood search* secara umum adalah sebagai berikut.

1. Tentukan solusi awal σ.
2. Tentukan jumlah iterasi l yang akan digunakan.
3. Inisialisasi solusi awal (σ) sebagai solusi terbaik global (σ'').
4. Bentuk solusi lingkungan dengan melakukan metode penghapusan dan perbaikan sehingga didapat solusi lingkungan σ' .
5. Hitung nilai fungsi objektif dari solusi awal $f(\sigma'')$ dan solusi lingkungan $f(\sigma')$.
6. Tentukan apakah solusi lingkungan σ' tersebut akan diterima sebagai solusi baru menggantikan solusi sekarang atau tidak.
    a. Jika $f(\sigma') < f(\sigma'')$, maka solusi akan diperbarui yaitu solusi lingkungan σ' diterima sebagai solusi baru
    b. Jika $f(\sigma') > f(\sigma'')$, maka solusi lingkungan σ' tidak diterima sebagai solusi baru, dan lanjut ke langkah 7.
7. Lakukan pengecekkan apakah iterasi sudah mencapai itersi l.
    a. Jika belum mencapai itersi l , maka lakukan penyusunan ulang solusi lingkungan layak, yaitu dengan mengulangi langkah 4, 5, dan 6 hingga mencapai itersi l.
    b. Jika telah mencapai itersi l, maka lanjut ke langkah 8.
8. Proses dari algoritma *Large Neighborhood Search* telah selesai dan didapatkan solusi terbaik σ'' sebagai solusi akhir.

Langkah-langkah algoritma *large neighborhood search* secara umum di atas, dapat disajikan dalam digram alir berikut.



Keterangan : σ = solusi sekarang, σ'' = solusi terbaik, σ' = solusi lingkungan,
$f(\sigma)$ = nilai fungsi objektif σ l = banyaknya iterasi

Gambar 2.12 Diagram Alir Algoritma *Large Neighborhood Search*

G. *Benchmark*

Benchmark memilik pengertian yang berbeda-beda pada setiap bidang ilmu. Dalam ilmu komputer dan teknologi, *benchmark* merupakan suatu metode atau tindakan pengujian yang dilakukan dengan cara menjalankan beberapa program atau operasi lain yang bertujuan untuk mengetahui performansi dari komputer tersebut. Dalam ilmu ukur tanah, *benchmark* merupakan titik tetap yang diketahui ketinggiannya terhadap suatu bidang referensi tertentu, sedangkan dalam ilmu

ekonomi, *benchmark* adalah proses membandingkan kinerja suatu bisnis termasuk matriks biaya, siklus waktu, produktivitas, atau kualitas lain secara luas yang dianggap sebagai tolak ukur standar industri atau praktik terbaik.

Dari pengertian yang telah diuraikan di atas, secara umum *benchmark* dapat diartikan sebagai sebuah ukuran kuantitatif yang digunakan untuk memberikan penilaian atau membandingkan suatu kinerja dengan tujuan untuk meningkatkan kualitas kinerja tersebut. Sebuah *benchmark* setidaknya memiliki 3 kriteria, yaitu sebagai berikut.

1. Konfigurasi

   Sebuah *benchmark* yang baik memungkinkan penguji atau yang menjalankan *benchmark* melakukan konfiguarasi sehingga faktor-faktor penentu kinerja dapat ditentukan dan diisolasi.

2. Konsistensi

   Benchmark yang baik memberikan hasil konsisten tiap kali dijalankan. Pengujian berulang perlu dilakukan untuk melihat apakah hasil *benchmark* memang benar. Pengujian ynag berulang juga akan memperlihatkan tingkat deviasi atau penyimpangan untuk *benchmark* tersebut. Standar tingkat deviasi 3% hingga 5% dari hasil pengulangan.

3. Kemampuan untuk bisa diulang kembali

   Karakteristik ini memungkinkan orang lain dengan konfigurasi yang sama (*hardware* atau *software*) mendapatkan hasil yang sama, konsisten dengan pengujian awak dengan standar deviasi 5 %.

Data *benchmark* adalah satu bentuk data komparatif yang menjadi standar atau dasar penilaian yang dijalankan pihak penguji, baik itu *reviewer*, pengguna maupun produsen untuk melihat kinerja sebuah produk, bagaimana dan seberapa baik atau cepat produk tersebut menjalankan test hingga selesai.

Pada penelitian ini, untuk melihat efektivitas kinerja dari dua algoritma yaitu algoritma *simulated annealing* dan *large neighborhood search,* digunakan data *benchmark* Li dan Lim dan data *benchmark* Benavent, dkk. yang akan diselesaikan. Data *benchmark* Li dan Lim merupakan himpunan data yang dibuat oleh Haibing Li dan Andrew Lim sebagai hasil pengembangan dari 56 contoh masalah (*instances)* pada data *benchmark* Solomon. Pengembangan dilakukan dengan memasangkan lokasi pelanggan dalam rute secara acak pada solusi yang didapatkan dengan pendekatan heuristik. Sama halnya dengan data *benchmark* Solomon yang terdiri dari 6 kelas tipe data, yaitu C1, C2, R1, R2, RC1, dan RC2, data *benchmark* Li dan Lim juga memiliki 6 kelas tipe data yaitu, LC1, LC2, LR1, LR2, LRC1, dan LRC2. Semua tipe data *benchmark* Li dan Lim tersebut memiliki 100 pelanggan dengan beberapa tambahan simpul *dummy* jika dibutuhkan sebagai pasangan, depot utama, kendala kapasitas, *time windows, precedence*, dan kendala *pairing*. Seiring berjalannya waktu, jumlah pelanggan pada tipe data *benchmark* Li dan Lim semakin berkembang, tidak hanya terbatas pada 100 pelanggan, tetapi juga 200, 400, 600, 800, dan 1000 pelanggan.

Pelanggan yang terdapat pada tipe LC adalah pelanggan yang membentuk kelompok atau *cluster*. Pada tipe LR, pelanggan tersebar secara acak. Sedangkan pada tipe LRC, sebagian pelanggan berkelompok dan sebagian pelanggan lainnya

tersebar secara acak. Tipe data LC1, LR1, dan LRC1 memiliki rentang waktu (*time windows)* depot yang pendek, sedangkan LC2, LR2, dan LRC2 mempunyai *time windows* depot yang lebih panjang. Data *benchmark* Li dan Lim dapat diakses di http://www.sintef.no/Projectweb/TOP/PDPTW/Li--Lim-benchmark/.

Selain data *benchmark* dari Li dan Lim, penelitian ini juga menggunakan data *benchmark* dari Benavent, dkk yaitu himpunan data yang dibuat oleh Benavent, Landete, Mota, dan Tirado sebagai hasil pengembangan dari data *benchmark* Li dan Lim 100 pelanggan. Data *benchmark* Benavent, dkk mengembangkan 5 himpunan data dari 56 contoh masalah (*instances)* data *benchmark* Li dan Lim yaitu himpunan data dengan 20, 30, 40, 50, dan 60 pelanggan. Data b*enchmark* Benavent, dkk memiliki tipe dan format data yang sama dengan data *benchmark* Li dan Lim. Data *benchmark* Benavent, dkk. dapat diakses di http://www.mat.ucm.es/~gregoriotd/PDPLT.htm.

BAB III
PEMBAHASAN

Pada bab III ini akan dijelaskan mengenai penggunaan algoritma *simulated annealing* dan *large neighborhood search* dalam penyelesaian masalah *pickup and delivery vehicle routing problem with time windows* (PDPTW) dan implementasinya pada data masalah penelitian optimasi (*benchmark instances*) dengan menggunakan Matlab (*Matrix Laboratory*). Selanjutnya akan dianalisa efektivitas dari kedua algoritma tersebut berdasarkan hasil implementasi yang telah diperoleh. Di akhir penelitian, diberikan sebuah contoh permasalahan PDPTW yang diselesaikan dengan menggunakan algoritma *simulated annealing* dan *large neighborhood search.*

A. Penggunaan Algoritma *Simulated Annealing* dan *Large Neighborhood Search* dalam Penyelesaian PDPTW

Penggunaan algoritma *simulated annealing* dan *large neighborhood search* dalam menyelesaikan masalah *pickup and delivery vehicle routing problem with time windows* (PDPTW) dimulai dengan membentuk solusi awal menggunakan algoritma *insertion heuristic* yang akan dijelaskan pada subbab a.1. kemudian dilanjutkan dengan penggunaan algoritma *simulated annealing* yang akan dijelaskan pada subbab A.2, dan penggunaan algoritma *large neighborhood search* pada subbab A.3.

1. Algoritma *Insertion Heuristik* dalam Pembentukan Solusi Awal

Algoritma *insertion heuristic* merupakan algoritma yang membentuk solusi dengan memilih pelanggan pertama (*seed customer*) untuk masuk ke dalam rute yang kemudian dilanjutkan dengan menyisipkan pelanggan yang belum dilayani ke dalam rute tersebut, hingga tidak ada lagi pelanggan yang dapat disisipkan. Langkah-langkah algoritma *insertion heuristic* dalam pembentukan solusi awal pada masalah PDPTW yang diadopsi dari Risya P. (2012) adalah sebagai berikut.

1. Tentukan parameter yang akan digunakan.

   Pada penelitian ini, digunakan parameter yang mengacu pada Solomon (1987) yaitu $\mu = 1$, $\alpha_1 = 0.5$, $\alpha_2 = 0.5$, dan $\alpha_3 = 0$.
2. Buat daftar pelanggan yang akan dilayani perdasarkan keterurutan.
3. Pilih pelanggan jemput pertama untuk masuk ke dalam rute *R.*

   Pemilihan pelanggan jemput pertama berdasarkan jarak terjauh dari depot ke pelanggan jemput. Kemudian, masukkan pelanggan jemput tersebut ke dalam rute beserta pasangannya. Misalkan, pelanggan i adalah pelanggan jemput yang memiliki jarak terjauh dari depot, maka rute awal yang terbentuk adalah $R = \{0 - i - @i - @0\}$, dengan 0 dan @0 adalah depot.
4. Hapus pelanggan i dan $@i$ dari daftar pelanggan.
5. Pilih pelanggan jemput j yang belum masuk ke dalam rute berdasarkan keterurutan pada daftar pelanggan, kemudian sisipkan ke dalam rute yang terbentuk.

Misalkan pelanggan *j* akan disisipkan ke dalam rute $R = \{0 - i - @i - @0\}$, maka kemungkinan rute yang terbentuk adalah $R = \{0 - j - i - @i - @0,\ 0 - i - j - @i - @0,\ 0 - i - @i - j - @0\}$.

6. Lakukan pengecekkan apakah rute yang terbentuk memenuhi solusi layak.

   Pada langkah ini, tiap kemungkinan rute yang terbentuk akan diperiksa apakah memenuhi solusi layak atau tidak. Solusi dikatakan layak jika memenuhi kendala kapasitas dan *time windows*. Sedangkan kendala *precedence* dan *pairing* akan dilengkapi saat pelanggan antar $@j$ mulai disisipkan ke dalam rute.

   a. Jika memenuhi solusi layak, maka hitung biaya penyisipan pelanggan *j* menggunakan kriteria ketiga (persamaan (2.23) – (2.25)). Jika terdapat lebih dari satu rute yang memenuhi solusi layak, maka pilih rute dengan biaya penyisipan terkecil.

   b. Jika tidak memenuhi solusi layak, maka lakukan pengecekkan apakah pelanggan *j* merupakan pelanggan terakhir dalam daftar pelanggan.

      1) Jika pelanggan *j* adalah pelanggan terakhir, maka kembali lagi ke langkah 3.

      2) Jika pelanggan *j* bukan pelanggan terakhir, maka kembali lagi ke langkah 5.

7. Pilih pelanggan antar $@j$ yang merupakan pasangan pelanggan jemput *j*, kemudian sisipkan ke dalam rute yang telah terbentuk.

   Misalkan rute yang memenuhi solusi layak dengan penyisipan pelanggan *j* adalah $R = \{0 - i - j - @i - @0\}$, maka kemungkinan rute yang terbentuk

dengan penyisipan pelanggan antar $@j$ adalah adalah $R = \{0 - i - j - @j - @i - @0,\ 0 - i - j - @i - @j - @0\}$.

8. Lakukan pengecekkan apakah rute yang terbentuk memenuhi solusi layak.

   Pada langkah ini, tiap kemungkinan rute yang terbentuk akan diperiksa apakah memenuhi solusi layak atau tidak. Solusi dikatakan layak jika memenuhi kendala kapasitas, *time windows*, *precedence,* dan *pairing.*

   a. Jika rute memenuhi solusi layak, maka hitung biaya penyisipan pelanggan $@j$ menggunakan kriteria ketiga (persamaan (2.23) – (2.25)). Jika terdapat lebih dari satu rute yang memenuhi solusi layak, maka pilih rute dengan biaya penyisipan terkecil.

   b. Jika rute tdak memenuhi solusi layak, maka lanjut ke langkah 10.

9. Hapus pelanggan j dan $@j$ dari daftar pelanggan.

10. Lakukan pengecekkan apakah ada pelanggan yang masih dapat disisipkan ke dalam rute tersebut.

    a. Jika masih ada pelanggan yang dapat disisipkan ke dalam rute, maka ulangi langkah 5 dan seterusnya secara terurut.

    b. Jika tidak ada lagi pelanggan yang dapat disisipkan ke dalam rute tersebut, maka bentuk rute baru dengan kembali ke langkah 3 dan seterusnya secara terurut.

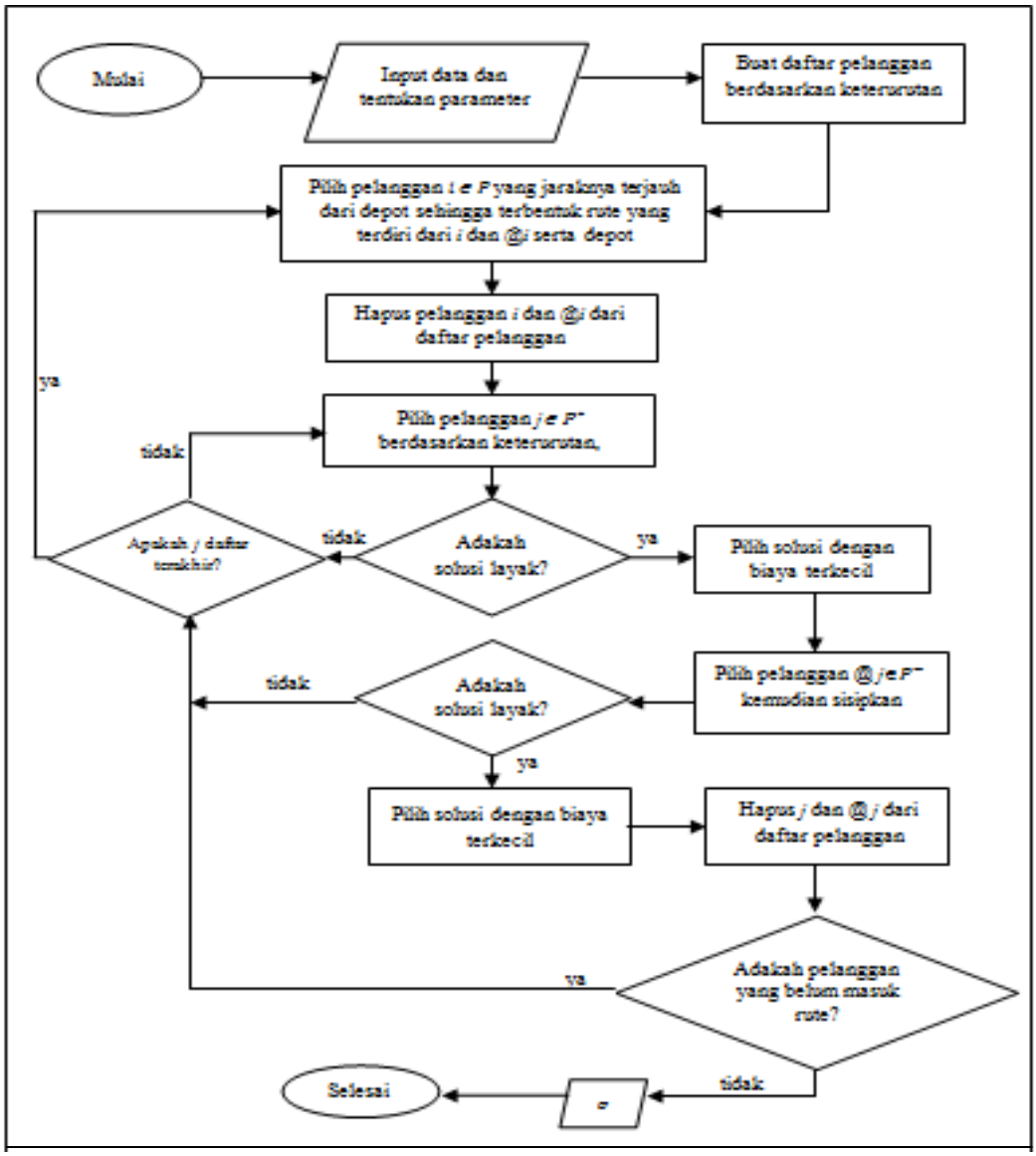
11. Lakukan pengecekkan apakah masih ada pelanggan yang belum masuk ke dalam rute.

    a. Jika masih ada pelanggan yang belum masuk ke dalam rute, maka ulangi langkah 3 dan seterusnya secara terurut.

b. Jika semua pelanggan telah masuk ke dalam rute, maka lanjut ke langkah 12.

12. Proses dari algoritma *insertion heuristic* telah selesai dan didapatkan solusi σ, yaitu berupa sekumpulan rute yang memenuhi kendala.

Penjelasan mengenai langkah-langkah algoritma *insertion heuristic* dalam pembentukan solusi awal pada masalah PDPTW di atas, dapat ditampilkan dalam diagram alir seperti pada Gambar 3.1.



Gambar 3.1 Diagram Alir Algoritma *Insertion Heuristic*

2. Penggunaan Algoritma *Simulated Annealing*

Algoritma *simulated annealing* merupakan algoritma yang berdasar pada metode probabilisik, yaitu metode penerimaan solusi baru berdasarkan probabilitas tertentu yang dikembangkan oleh Kirkpatrick *et al.* (1983) untuk menemukan nilai minimum global dari sebuah fungsi yang memiliki beberapa nilai lokal minimal.

Pada algoritma *simulated annealing*, terdapat empat tahapan utama dalam penyelesaian masalah PDPTW.

1. Pembentukan Solusi Awal

   Tahap pembentukan solusi awal pada algoritma *simulated annealing* dalam penelitian ini dilakukan dengan menggunakan algoritma *insertion heuristic* seperti yang telah dijelaskan pada subbab sebelumnya.

2. Pembentukan Solusi Sublingkungan

   Sublingkungan dibentuk untuk mencari solusi yang lebih baik dari solusi sebelumnya, yaitu melalui eksplorasi lingkungan dengan memilih secara acak pelanggan jemput i dan pelanggan antar $@i$ dari N yang merupakan himpunan pelanggan. Kemudian dilakukan *pair relocation* terhadap pelanggan i dan $@i$ tersebut. *Pair relocation* yaitu pemindahan sepasang pelanggan terpilih dari satu rute ke rute lain. Himpunan solusi sublingkungan dinotasikan dengan $N(i, \sigma)$ yaitu himpunan solusi lingkungan layak yang terbentuk setelah dilakukan *pair relocation* terhadap pelanggan i dan $@i$. Misalkan terdapat solusi awal $\sigma = \{0 - i - @i - j - @j, 0 - m - @m\}$ dan

secara acak terpilih pelanggan *j*, maka dilakukakn pair relocation terhadap pelanggan *j* dan *@j*. Sublingkungan yang mungkin terbentuk adalah

1. $\{0-i-@i-0, 0-j-@j-m-@m-0\}$,
2. $\{0-i-@i-0, 0-j-m-@j-@m-0\}$,
3. $\{0-i-@i-0, 0-j-m-@m-@j-0\}$,
4. $\{0-i-@i-0, 0-m-j-@j-@m-0\}$,
5. $\{0-i-@i-0, 0-m-j-@m-@j-0\}$, dan
6. $\{0-i-@i-0, 0-m-@m-j-@j-0\}$

Dari enam kemungkinan di atas, yang menjadi solusi sublingkungan adalah yang memenuhi kendala.

3. Penghitungan Fungsi Evaluasi

Fungsi evaluasi $e(\sigma)$ yang digunakan pada penelitian ini adalah fungsi evaluasi berdasarkan *lexicographic ordering* (Bent dan Hentenryck, 2003), yaitu

$$e(\sigma) = \langle |\sigma|, -\sum_{r\in\sigma} |r|^2, \sum_{r\epsilon\sigma} t(r)\rangle \qquad (3.1)$$

dengan

$|\sigma|$: jumlah rute pada solusi σ,

$|r|$: jumlah pelanggan di setiap rute pada solusi σ,

$t(r)$: biaya perjalanan dari himpunan rute pada solusi σ yang dapat dilihat dari total jarak tempuh seluruh kendaraan yang melalui himpunan rute tersebut.

4. Penghitungan Selisi Nilai Fungsi Evaluasi

   Selisih nilai fungsi evaluasi atau $\Delta\sigma$ akan dihitung jika solusi lingkungan tidak lebih baik dari solusi sebelumnya. Dengan fungsi evaluasi yang berdasarkan *lexicographic ordering*, maka nilai $\Delta\sigma$ adalah selisih nilai pertama yang membedakan fungsi evaluasi antara dua solusi (Bent dan Hentenryck, 2003).

   Contoh: $e(\sigma_i) = \langle 2, -40, 160 \rangle$ dan $e(\sigma_j) = \langle 2, -45, 160 \rangle$, maka

   $$\Delta\sigma = -40 - (-45) = 5.$$

Langkah-langkah penggunaan algoritma *simulated annealing* dalam penyelesaian masalah PDPTW yang diadopsi dari Risya Priwarnela (2012) adalah sebagai berikut.

1. Tentukan solusi awal σ.

   Solusi awal σ merupakan solusi yang didapatkan dari solusi akhir algoritma *insertion heuristic.*

2. Tentukan suhu awal H_0, suhu akhir H_t , *cooling rate* α, dan jumlah iterasi L_t.

3. Tetapkan suhu awal H_0 sebagai pengontrol apakah solusi baru diterima atau tidak.

4. Bentuk solusi sublingkungan σ'.

   Pembentukan solusi sublingkungan σ' dilakukan dengan melakukan *pair relocation* terhadap pelanggan i dan $@i$ yang dipilih secara acak dari N yang merupakan himpunan pelanggan, sehingga terbentuk himpunan solusi sub-lingkungan $N(i, \sigma)$.

5. Lakukan pengecekkan apakah sublingkungan terbentuk.
   a. Jika sublingkungan tebentuk, maka hitung nilai fungsi evalusi sub-lingkungan $e(\sigma'_i)$, menggunakan persamaan (3.1) dan urutkan solusinya dari nilai fungsi evaluasi yang terkecil, $\{\sigma'_1 \leq \cdots \leq \sigma'_s\}$, dengan *s* adalah banyaknya solusi dalam sublingkungan.
   b. Jika sublingkungan tidak terbentuk, maka lanjut ke langkah 8 dan seterusnya secara terurut.
6. Hitung nilai fungsi evaluasi solusi saat ini $e(\sigma)$ berdasarkan persamaan (3.1). Kemudian, bandingkan hasilnya dengan solusi sublingkungan yang memiliki nilai fungsi evalusi sublingkungan terkecil, $e(\sigma'_1)$. Lakukan pengecekkan apakah nilai $e(\sigma'_1) < e(\sigma)$.
   a. Jika $e(\sigma'_1) < e(\sigma)$, maka solusi sublingkungan diterima sebagai solusi baru dan lanjut ke langkah 8.
   b. Jika $e(\sigma'_1) > e(\sigma)$, maka tentukan bilangan random *r* menggunakan rumus berikut.

$$r = [random\ (0,1)^{\beta} \times s] \tag{3.2}$$

   dengan β adalah parameter yang telah ditentukan dan *s* adalah banyaknya solusi dalam sublingkungan. Kemudian lanjut ke langkah 7.
7. Hitung selisih nilai fungsi evaluasi $\Delta\sigma$ berdasarkan persamaan (2.27), yaitu selisih antara nilai fungsi evaluasi sublingkungan bilangan random *r*, $e(\sigma'_r)$, dengan nilai fungsi evaluasi solusi saat ini, $e(\sigma)$. Kemudian lakukan pengecekkan apakah selisih nilai fungsi $\Delta\sigma \leq 0$.

a. Jika $\Delta\sigma \leq 0$, maka solusi sublingkungan bilangan random r , σ'_r diterima sebagai solusi baru.

b. Jika $\Delta\sigma > 0$, tentukan bilangan random P antara 0 dan 1. Kemudian, lakukan pengecekkan apakah $P \leq exp(-\Delta\sigma/H)$.

1) Jika $P \leq exp(-\Delta\sigma/H)$, maka solusi sublingkungan bilangan random r , σ'_r diterima sebagai solusi baru.

2) Jika $P > exp(-\Delta\,\sigma/H)$, maka lanjut ke langkah 8.

8. Lakukan pengecekkan apakah iterasi sudah mencapai iterasi L_t.

a. Jika iterasi sudah mencapai iterasi L_t, maka cek apakah suhu saat ini sama dengan suhu pada saat t, $H = H_t$.

1) Jika $H = H_t$, maka lanjut ke langkah 10.

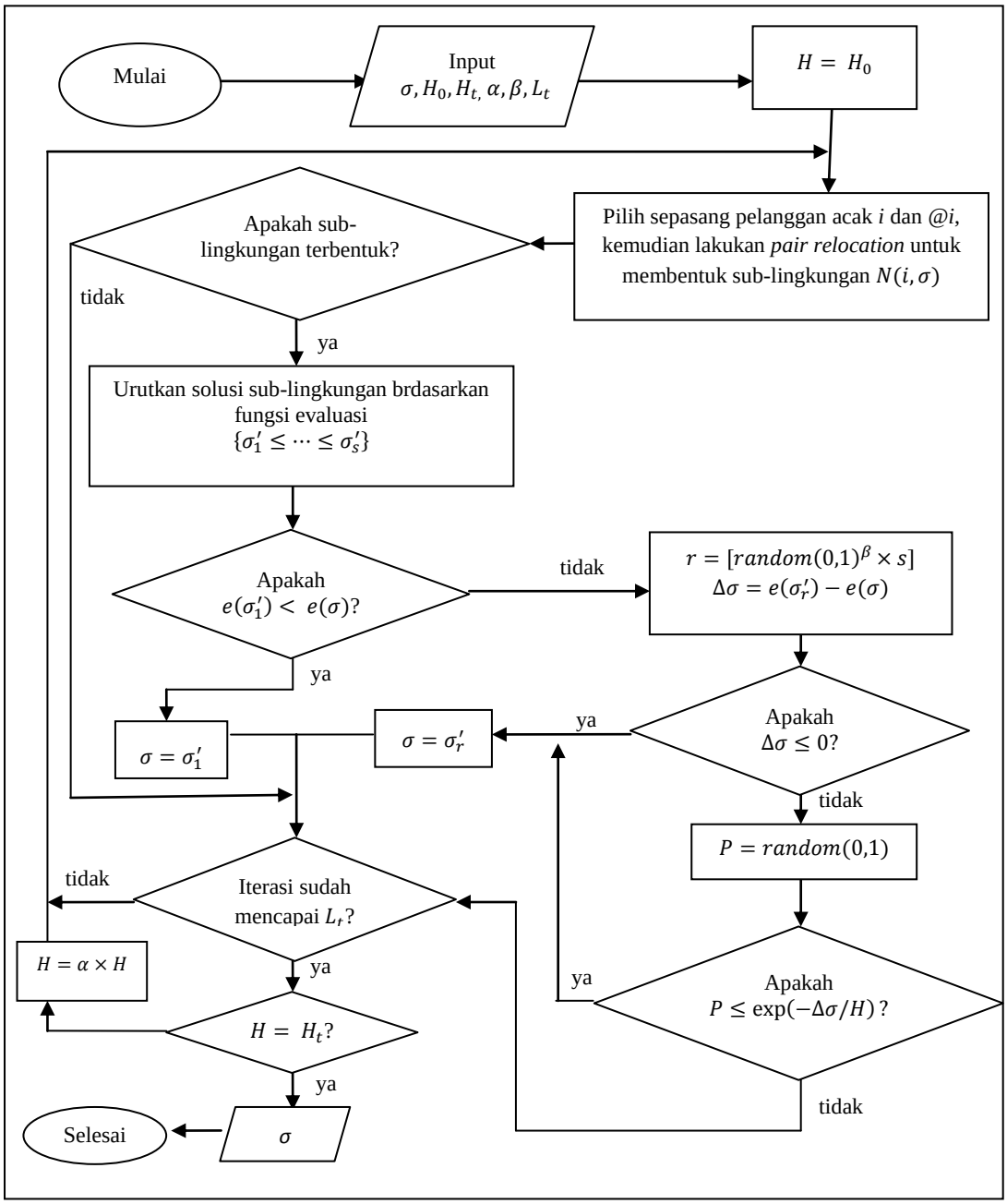
2) Jika $H \neq H_t$, maka lanjut ke langkah 9.

b. Jika iterasi belum mencapai iterasi L_t, maka kembali ke langkah 4. Ulangi hingga iterasi mencapai iterasi L_t.

9. Lakukan penurunan suhu H secara perlahan menggunakan suatu parameter cooling rate α , lalu kembali ke langkah 4 dan seterusnya secara terurut.

10. Proses dari algoritma *simulated annealing* telah selesai dan didapatkan solusi akhir σ.

Penjelasan mengenai langkah-langkah algoritma *simulated annealing* dalam penyelesaian masalah PDPTW di atas, dapat ditampilkan dalam diagram alir seperti pada Gambar 3.2.



Gambar 3.2 Diagram Alir Algoritma *Simulated Annealing*

3. Penggunaan Algoritma *Large Neighborhood Search*

Algoritma *large neighborhood search* merupakan algoritma pencarian solusi terbaik yang memanfaatkan metode *local search*, yaitu metode pencarian solusi

bedasarkan lingkungan dari suatu solusi awal. Pada algoritma *large neighborhood search*, terdapat tiga tahapan utama dalam penyelesaian masalah PDPTW.

1. Pembentukan Solusi Awal

   Tahap pembentukan solusi awal pada algoritma *large neighborhood search* dilakukan dengan menggunakan algoritma *insertion heuristic* seperti yang telah dijelaskan pada subbab sebelumnya.

2. Pembentukan Solusi Sublingkungan

   Solusi sublingkungan dibentuk dengan menggunakan metode penghapusan dan perbaikan terhadap solusi yang ada. Berikut akan dijelaskan metode penghapusan dan perbaikan yang digunakan dalam penelitian ini.

   a. Metode Penghapusan

      Metode penghapusan yang digunakan dalam penelitian ini adalah dengan memilih sebanyak x pasang pelanggan yang akan dihapus dari himpunan rute yang menjadi solusi awal σ.

   b. Metode Perbaikan

      Metode perbaikan dilakukan untuk membentuk kembali solusi yang sebelumnya telah diubah oleh metode penghapusan. Metode perbaikan yang digunakan dalam penelitian ini adalah dengan menyisipkan kembali satu per satu x pasang pelanggan yang telah dihapus dari rute dengan urutan acak. Penyisipan ini akan menghasilkan solusi sublingkungan yang layak.

Himpunan solusi sublingkungan dinotasikan dengan $N(\sigma, x)$ yaitu himpunan yang terdiri dari solusi lingkungan layak yang terbentuk setelah menyisipkan kembali x pasang pelanggan yang telah dihapus dari rute.

3. Penghitungan Fungsi Evaluasi

   Fungsi evaluasi $f(\sigma)$ digunakan untuk menentukan apakah solusi sub-lingkungan yang terpilih dapat menggantikan solusi sebelumnya atau tidak. Solusi sub-lingkungan yang terpilih dapat diterima sebagai solusi baru menggantikan solusi sebelumnya apabila solusi tersebut lebih baik dari solusi sebelumnya berdasarkan *lexicographic ordering*. Fungsi evaluasi yang berdasarkan *lexicographic ordering* (Bent dan Hentenryck, 2003), yaitu

$$f(\sigma) = \langle(|\sigma|, \sum_{r\epsilon\sigma} t(r)\rangle, \tag{3.3}$$

   dengan

   $|\sigma|$: banyaknya rute pada solusi σ,

   $t(r)$: biaya perjalanan dari himpunan rute pada solusi σ yang dapat dilihat dari total jarak tempuh dari seluruh kendaraan yang melalui himpunan rute tersebut.

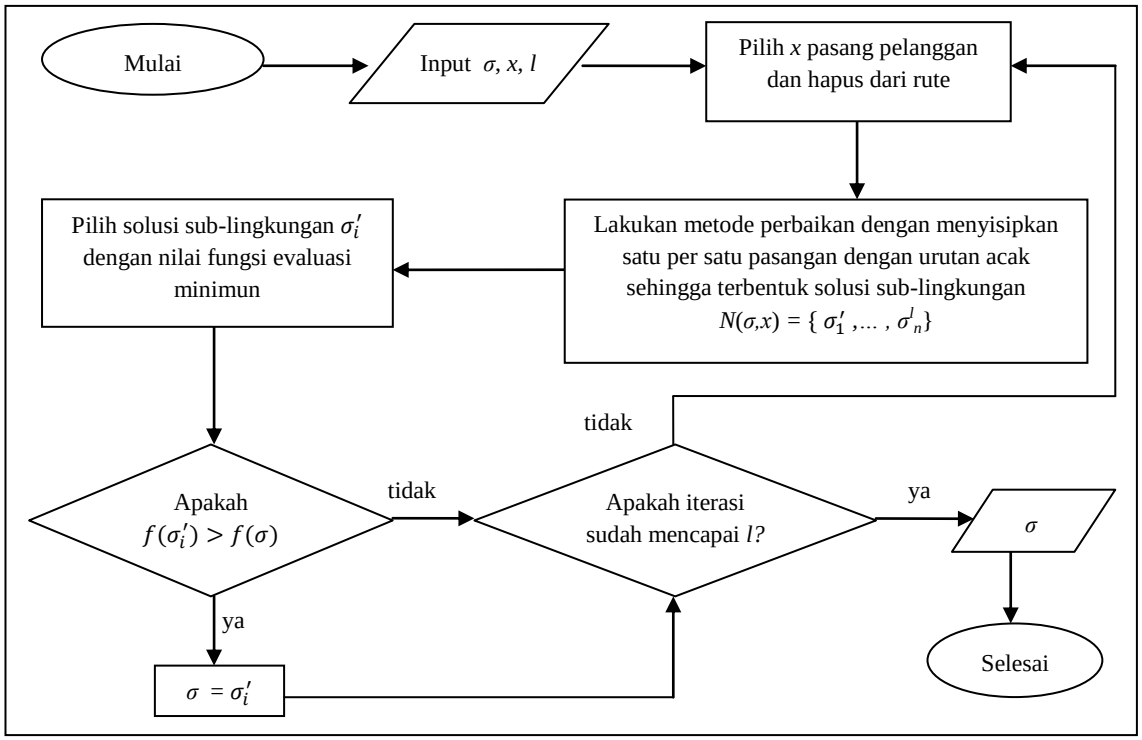
Langkah-langkah penyelesaian masalah PDPTW dengan menggunakan algoritma *large neighborhood search* yang diadopsi dari Risya Priwarnela (2012), adalah sebagai berikut.

1. Tentukan solusi awal σ.

   Solusi awal σ merupakan solusi yang didapatkan dari solusi akhir algoritma *insertion heuristic*.

2. Tentukan banyaknya pasangan pelanggan x, dan banyaknya iterasi l.
3. Bentuk solusi sub-lingkungan σ' menggunakan metode penghapusan, yaitu dengan memilih x pasangan pelanggan secara acak dan kemudian hapus x pasangan pelanggan tersebut dari rute.
4. Bentuk solusi sublingkungan σ' menggunakan metode perbaikan, yaitu dengan menyisipkan kembali satu per satu x pasangan pelanggan yang telah dihapus sebelumnya dengan urutan acak sehingga terbentuk himpunan solusi sub-lingkungan $N(\sigma, x) = \{\sigma'_1, \dots, \sigma'_n\}$, dimana n adalah banyaknya solusi dalam sub-lingkungan.
5. Hitung nilai fungsi evaluasi solusi sublingkungan $f(\sigma'_i)$ dan nilai fungsi evaluasi solusi saat ini $f(\sigma)$ berdasarkan persamaan (3.3). Lalu pilih solusi sub-lingkungan σ'_i dengan nilai fungsi evaluasi minimum untuk dibandingkan hasilnya dengan $f(\sigma)$.
6. Lakukan pengecekan apakah nilai $f(\sigma'_i) < f(\sigma)$.
    a. Jika $f(\sigma'_i) < f(\sigma)$, maka solusi sub-lingkungan σ'_i diterima sebagai solusi baru.
    b. Jika $f(\sigma'_i) > f(\sigma)$, maka lanjut ke langkah 7
7. Lakukan pengecekkan apakah iterasi sudah mencapai iterasi *l*.
    a. Jika iterasi sudah mencapai iterasi *l*, maka lanjut ke langkah 8.
    b. Jika iterasi belum mencapai iterasi *l*, maka kembali lagi ke langkah 3 dan seterusnya secara terurut. Ulangi sampai iterasi mencapai iterasi *l*.
8. Proses dari algoritma *large neighborhood search* telah selesai dan didapatkan solusi akhir σ.

Penjelasan mengenai langkah-langkah algoritma *large neighborhood search* di atas dapat ditampilkan dalam diagram alir seperti pada Gambar 3.3.



3.3 Diagram Alir Algoritma *Large Neighborhood Search*

B. Implementasi Algoritma *Simulated Annealing* dan *Large Neighborhood Search* pada Data *Benchmark*

Pada subbab sebelumnya telah dijelaskan mengenai penggunaan algoritma *simulated annealing* dan *large neighborhood search* dalam penyelesaian masalah PDPTW. Pada subbab ini, algoritma *simulated annealing* dan *large neighborhood search* tersebut akan diimplementasikan pada data *benchmark* ke dalam perangkat lunak, yaitu menggunakan Matlab 7.8 (R2009a) dengan spesifikasi sebagai berikut.

Sistem Operasi : Windows 7 *Home Basic*

Processor : Intel (R) Atom (TM) CPU N550 @1.50 GHz 1.50 GHz

RAM : 2,00 GB

System Type : 32-bit *Operating System*

1. Data *Benchmark*

Algoritma *simulated annealing* dan *large neighborhood search* akan diimplementasikan pada data *benchmark* dari Benavent, dkk dengan 40 pelanggan yaitu tipe LR104_40. Selain itu, implementasi dilakukan juga pada 100 data *benchmark* lainnya yang diambil secara acak dari data *benchmark* Benavent, dkk. dengan 20, 30, 40, 50, dan 60 pelanggan, serta data *benchmark* Li dan Lim dengan 100 pelanggan.

2. Penetapan Parameter

Parameter-parameter yang digunakan pada setiap algoritma dalam penelitian ini adalah sebagai berikut.

a. *Insertion Heuristic*

Pada algoritma *insertion heuristic,* digunakan variabel μ , α_1, α_2, dan α_3 untuk menghitung fungsi biaya atau biaya penyisipan. Nilai setiap variabel tersebut mengacu pada Solomon (1987) yaitu $\mu = 1$, $\alpha_1 = 0.5$, $\alpha_2 = 0.5$, dan $\alpha_3 = 0$.

b. *Simulated Annealing*

Pada algoritma *simulated annealing,* digunakan variabel suhu awal H_0, suhu akhir H_t, iterasi tiap suhu L_t, *cooling rate* α, dan parameter β. Nilai setiap

variabel tersebut mengacu pada Bent dan Hentenryck (2003), yaitu $H_0 = 2000$, $H_t = 0.01$, $L_t = 2500$, α = 0.01 , dan parameter $\beta = 10$.

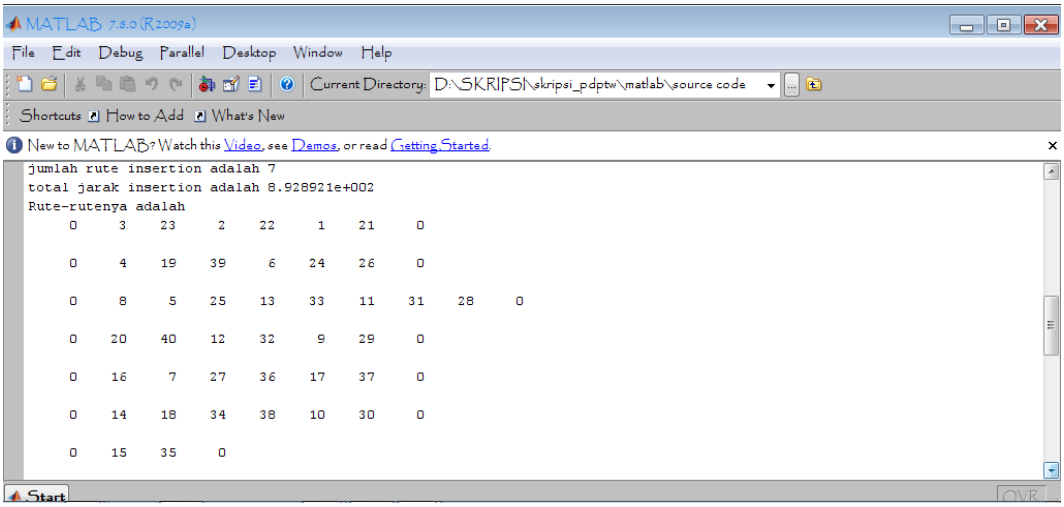
c. *Large Neighborhood Search*

Pada algoritma *large neighborhood search*, digunakan variabel *x* yang menyatakan jumlah pasangan pelanggan yang akan dihapus dan disisipkan kembali pada rute, dan variabel *l* yang menyatakan banyaknya iterasi yang ditetapkan untuk melakukan tahap *large neighborhood search.* Pada penelitian ini, ditentukan *x* = 3 dan *l* = 500.

3. Implementasi Algoritma *Simulated Annealing* dan *Large Neighborhood Search* pada Data LR104_40

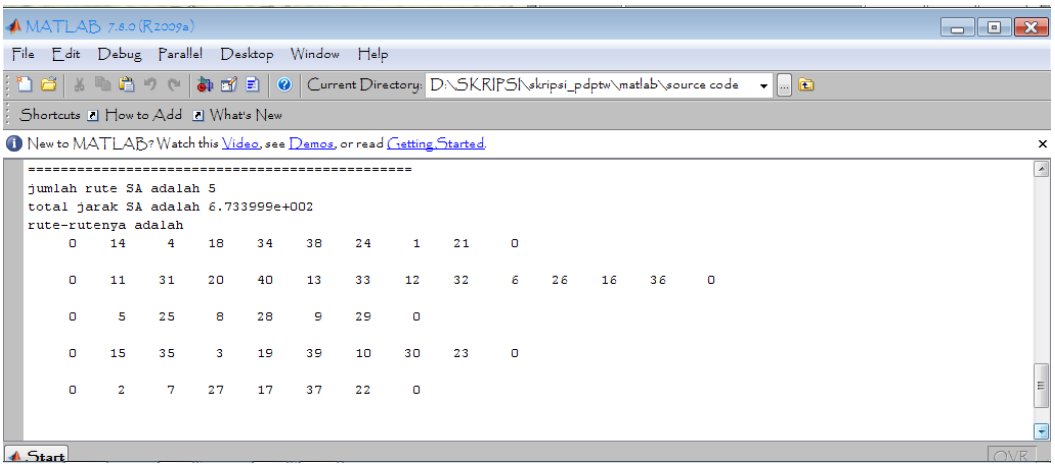
Agar dapat melihat hasil implementasi dari algoritma *simulated annealing* dan *large neighborhood search*, maka peneliti mengambil salah satu kasus *benchmark instances*, yaitu pada data *benchmark* dari Benavent, dkk tipe LR104_40. Pada tipe ini, terdapat 40 pelanggan yang tersebar secara acak dan memiliki rentang waktu (*time windows)* depot yang pendek. Adapun implementasi tersebut dapat dijelaskan sebagai berikut.

Implementasi algoritma *simulated annealing* dan *large neighborhood search* pada data *benchmark* tipe LR104_40 dimulai dengan pembentukan solusi awal menggunakan algoritma yang sama, yaitu algoritma *insertion heuristic*. *Source code* algoritma *insertion heuristic* dalam penelitian ini diadopsi dari Risya P. (2012) yang dapat dilihat pada lampiran 1. Dari penggunaan algoritma *insertion heuristic*, didapatkan sebuah solusi yang terdiri dari 7 rute dengan total jarak 892,8921 km. Berikut disajikan *output* Matlab untuk algoritma *insertion heuristic* .



Gambar 3.4 *Output* Matlab untuk Algoritma *Insertion Heuristic*

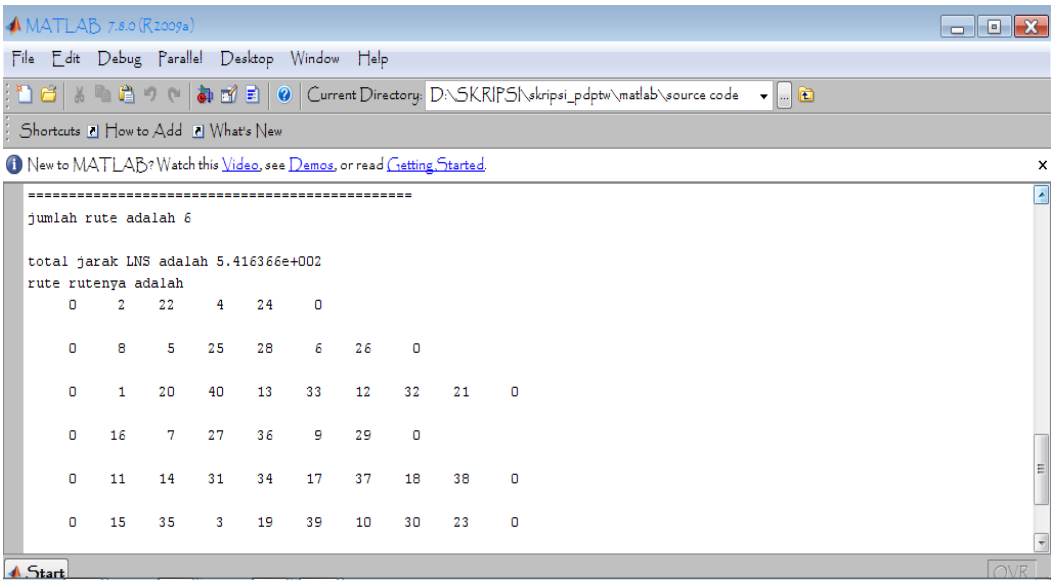
Selanjutnya solusi yang didapat dari penggunaan algoritma *insertion heuristic* tersebut digunakan sebagai solusi awal pada algoritma *simulated annealing* dan *large neighborhood search. Source code* algoritma *simulated annealing* dan *large neighborhood search* dalam penelitian ini diadopsi dari Risya P. (2012) yang dapat dilihat pada lampiran 1. Adapun solusi yang didapat dari penggunaan algoritma *simulated annealing* dengan solusi awal *insertion heuristic* yaitu berupa sebuah solusi yang terdiri dari 5 rute dengan total jarak 692,2026 km. Berikut disajikan *output* Matlab untuk algoritma *simulated annealing*.



Gambar 3.5 *Output* Matlab untuk Algoritma *Simulated Annealing*.

Pada Gambar 3.4 dan Gambar 3.5, terlihat bahwa penggunaan algoritma *simulated annealing* dengan *insertion heuristic* sebagai solusi awal telah berhasil meminimalkan jumlah rute dari 7 rute menjadi 5 rute dan total jarak dari 892,8921 km menjadi 692,2026 km.

Berikut ini disajikan pula *output* Matlab untuk algoritma *large neighborhood search* dengan solusi awal *insertion heuristic*. Pada penggunaan algoritma *large neighborhood search*, diperoleh sebuah solusi yang terdiri dari 6 rute dengan total jarak 541,6366 km.



Gambar 3.6 *Output* Matlab untuk Algoritma *Large Neighborhood Search*

Pada Gambar 3.4 dan Gambar 3.6, terlihat bahwa penggunaan algoritma *large neighborhood search* dengan *insertion heuristic* sebagai solusi awal telah berhasil meminimalkan jumlah rute dari 7 rute menjadi 6 rute dan total jarak dari 892,8921 km menjadi 541,6366 km. Adapun rekapitulasi hasil implementasi algoritma *simulated annealing* dan *large neighborhood search* pada data *benchmark* tipe LR104_40 adalah sebagai berikut.

Tabel 3.1 Rekapitulasi Hasil Implementasi
pada Data LR104_40

Algoritma	Jumlah rute	Total jarak (km)
Insertion Heuristic	7	892,8921
Simulated Annealing	5	692,2016
Large Neeighborhood Search	6	541,6366

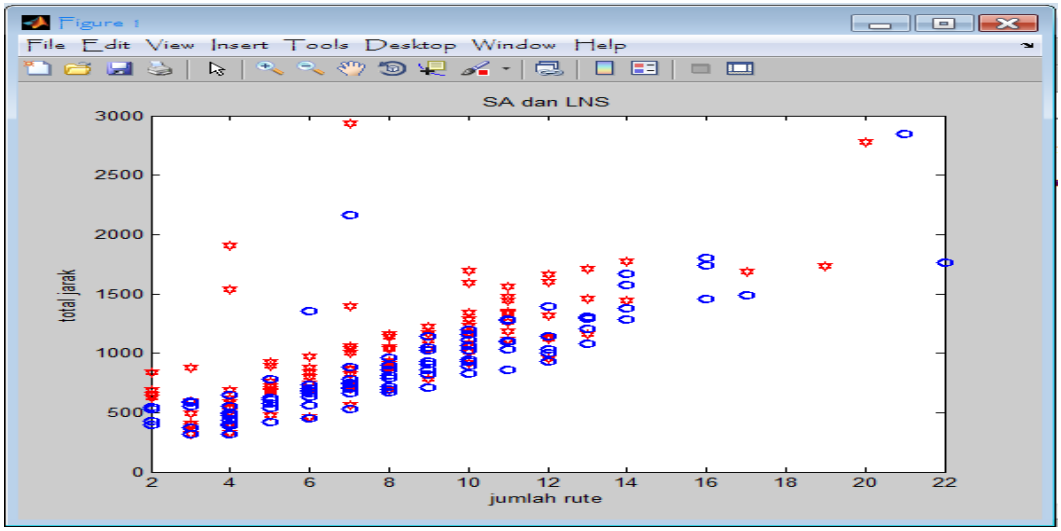
Pada Tabel 3.1, terlihat bahwa penggunaan algoritma *simulated annealing* dan *large neighborhood search* dalam penyelesaian masalah PDPTW efektif untuk menghasilkan solusi terbaik pada data *benchmark* tipe LR104_40. Algoritma *simulated annealing* lebih efektif dalam meminimalkan jumlah rute, sedangkan algoritma *large neighborhood search* lebih efektif dalam meminimalkan total jarak tempuh kendaraan. Hal ini terlihat dari algoritma *simulated annealing* yang dapat lebih meminimalkan banyaknya rute dari 7 rute menjadi 5 rute, sedangkan algoritma *large neighbborhood search* hanya dapat meminimalkan jumlah rute dari 7 rute menjadi 6 rute. Demikian juga dengan total jarak tempuh kendaraan, dimana algoritma *large neighborhood search* dapat lebih meminimalkan total jarak dari 892,8921 km menjadi 541,6366 km, sedangkan algoritma *simulated annealing* hanya dapat meminimalkan total jarak dari 892,8921 km menjadi 692,2026 km.

4. Implementasi Algoritma *Simulated Annealing* dan *Large Neighborhood Search* pada Data *Benchmark* Lainnya

Untuk dapat melihat hasil implementasi dari algoritma *simulated annealing* dan *large neighborhood search* secara lebih jelas, maka peneliti mengambil kasus *benchmark instances* pada data *benchmark* dari Benavent, dkk dengan 20, 30, 40,

50, dan 60 pelanggan serta data *benchmark* dari Li dan Lim dengan 100 pelanggan. Implementasi algoritma *simulated annealing* dan *large neighborhood search* dengan algoritma *insertion heuristic* sebagai solusi awal dilakukan pada 100 tipe data *benchmark* yang dipilih secara acak dari kedua data *benchmark* tersebut. Adapun hasil implementasinya dapat dilihat pada lampiran 2.

Tabel pada lampiran 2 memperlihatkan solusi yang dihasilkan dari penggunaan algoritma *simulated annealing* dan *large neighborhood search* dalam penyelesaian PDPTW pada setiap tahapnya dengan satu kali percobaan, dimana |K| adalah banyaknya rute yang terbentuk atau ekuivalen dengan banyaknya kendaraan dan TD adalah total jarak tempuh kendaraan (km). Secara umum, terlihat bahwa algoritma *simulated annealing* lebih efektif dalam mengurangi jumlah rute dan algoritma *large neighborhood search* lebih efektif dalam mengurangi total jarak tempuh kendaraan. Hasil yang terdapat pada lampiran 2 dapat disajikan dalam bentuk diagram seperti pada Gambar 3.7.



Keterangan : *** (warna merah) = *simulated annealing*
ooo (warna biru) = *large neighborhood search*

Gambar 3.7 Hasil Penggunaan Algoritma *Simulated Annealing* dan *Large Neighborhood Search* pada 100 Tipe Data *Benchmark*

Hasil yang sama dapat dilihat pada Gambar 3.7. Pada gambar tersebut dapat dilihat bahwa tanda berwarna merah banyak berada di ruas sebelah kiri dari pada tanda berwarna biru, dan begitu juga dengan tanda berwarna biru yang lebih banyak berada di ruas bawah dibandingkan dengan tanda berwarna merah. Hal ini juga menunjukan bahwa algoritma *simulated annealing* memberikan hasil yang lebih efektif dalam mengurangi jumlah rute dan algoritma *large neighborhood search* memberikan hasil yang lebih efektif dalam mengurangi total jarak tempuh kendaraan. Untuk lebih jelasnya, efektivitas dari kedua algoritma tersebut akan dijelaskan pada subbab selanjutnya.

C. Efektivitas Algoritma *Simulated Annealing* dan *Large Neighborhood Search* dalam Penyelesaian PDPTW

Dari hasil implementasi yang telah dilakukan dengan menggunakan algoritma *simulated annealing* dan *large neighborhood search* pada tipe data *benchmark* di atas, terlihat bahwa kedua algoritma tersebut berhasil dalam menyelesaikan masalah *pickup and delivery vehicle routing problem with time windows* (PDPTW), yaitu dalam mengurangi jumlah rute dan total jarak tempuh kendaraan. Penggunaan algoritma *simulated annealing* berhasil mengurangi jumlah rute dengan baik dan penggunaan algoritma *large neighborhood searc*h juga berhasil mengurangi total jarak tempuh kendaraan dengan baik.

1. Efektivitas Jumlah Rute

Penentuan efektivitas jumlah rute kendaraan dapat dilihat dari berapa banyaknya kendaraan yang digunakan untuk melakukan pelayanan atau jumlah rute yang terbentuk. Berdasarkan tabel pada lampiran 2, dapat dilihat bahwa

secara keseluruhan, algoritma *simulated annealing* menghasilkan jumlah rute yang minimal dibandingkan dengan algoritma *large neighborhood search*. Algoritma *simulated annealing* dapat mengurangi jumlah rute kendaraan pada 51 dari 100 tipe data *benchmark* yang diujicobakan, sedangkan algoritma *large neighborhood search* hanya dapat mengurangi jumlah rute pada 5 dari 100 tipe data *benchmark* yang diujicobakan. Dengan demikian, dapat disimpulkan bahwa berdasarkan jumlah rute kendaraan, algoritma *simulated annealing* merupakan algoritma yang efektif dalam mengurangi jumlah rute kendaraan.

2. Efektivitas Total Jarak Tempuh Kendaraan

Penentuan efektivitas total jarak tempuh kendaraan dapat dilihat dari seberapa jauh kendaraan melakukan perjalanan. Berdasarkan tabel pada lampiran 2, dapat dilihat bahwa secara keseluruhan, algoritma *large neighborhood search* menghasilkan total jarak tempuh kendaraan yang minimal dibandingkan dengan agoritma *simulated annealing.* Algoritma *large neighborhood search* dapat mengurangi total jarak tempuh kendaraan pada 90 dari 100 tipe data *benchmark* yang diujicobakan. Dengan demikian, dapat disimpulkan bahwa berdasarkan total jarak tempuh kendaraan, algoritma *large neighborhood search* merupakan algoritma yang efektif dalam mengurangi total jarak tempuh kendaraan .

Berdasarkan perbandingan efektivitas terhadap jumlah rute dan total jarak tempuh di atas, dapat disimpulkan bahwa algoritma *simulated annealing* efektif dalam mengurangi jumlah rute dan algoritma *large neighborhood search* efektif dalam mengurangi total jarak tempuh kendaraan. Selanjutnya, untuk melihat bagaimana efektivitas dari algoritma *simulated annealing* dan *large neighborhood*

search dalam kasus yang terjadi di kehidupan sehari-hari, maka akan diberikan sebuah contoh permasalahan PDPTW yang akan diselesaikan dengan kedua algoritma tersebut pada subbab berikutnya.

D. Penyelesaian Contoh Permasalahan PDPTW Menggunakan Algoritma *Simulated Annealing* dan *Large Neighborhood Search*

Pada subbab sebelumnya telah dijelaskan mengenai penggunaan algoritma *simulated annealing* dan *large neighborhood search* dalam penyelesaian masalah PDPTW dan implementasinya pada data *benchmark*, serta efektivitas dari kedua algoritma berdasarkan hasil implementasi tersebut. Pada subbab ini, untuk memperlihatkan bagaimana aplikasi dan efektivitas dari algorima *simulated annealing* dan *large neighborhood search* dalam kasus di kehidupan sehari-hari, maka berikut akan diberikan sebuah contoh permasalahan PDPTW yang akan diselesaikan dengan menggunakan algoritma *simulated annealing* dan *large neighborhood search*. Contoh permasalahan PDPTW yang diambil peneliti adalah masalah penentuan rute di Jogja Kurir Express.

1. Deskripsi Masalah

Jogja Kurir Express merupakan sebuah usaha yang bergerak di bidang jasa pengiriman barang yang berada di kota Yogyakarta. Jogja Kurir Express menerapkan sistem layanan jemput antar barang (*pickup and delivery service*) terhadap pelanggannya. Petugas/kurir dari Jogja Kurir Express akan menjemput/mengambil barang dari pelanggan jemput untuk kemudian diantar ke pelanggan antar sebagai tempat tujuan pengiriman barang. Wilayah operasi Jogja Kurir Express adalah di sekitaran kota Yogyakarta dengan tujuan pengiriman

Yogyakarta, Sleman, Bantul, Kulonprogo, Gunung Kidul, Purworejo, dan Magelang.

Jogja Kurir Express memiliki tiga jenis layanan pengiriman barang berdasarkan permintaan pelanggannya, yaitu *urgent service, one day service,* dan 2 – 3 *days service.* Jenis layanan *urgent service,* apabila pelanggan menginginkan barang untuk segera dijemput/diambil dan langsung diantar ke tempat tujuan pada saat itu juga. *One day service*, apabila pelanggan menginginkan pelayanan jemput dan antar barang dapat dilakukan dalam satu hari dan dengan rentang waktu yang telah ditentukan sebelumnya oleh pelanggan., sedangkan pada jenis layanan 2 – 3 *days service,* pelanggan menginginkan barang dapat sampai ke tempat tujuan dalam rentang waktu 2 – 3 hari. Pada penelitian ini, dipilih jenis layananan *one day sevice* yang sesuai dengan permasalahan PDPTW.

Untuk satu hari kerja, Jogja Kurir Express memiliki *transportation request* yang hendak dilayani. Setiap *transportation request* terdiri dari informasi mengenai nama pelanggan, alamat lokasi penjemputan/pengambilan dan tujuan pengiriman barang, *contact person* yang dapat dihubungi, beban yang diangkut/permintaan pelanggan, jenis barang, *time windows* pelanggan, dan jenis layanan.

Dalam melakukan pelayanan jemput dan antar barang, Jogja Kurir Express menentukan rute berdasarkan perkiraan saja tanpa mengetahui apakah jarak tempuh rute yang dipilih sudah minimal atau belum, sehingga mengakibatkan biaya bahan bakar yang dikeluarkan pun belum tentu minimal. Selain itu, karena banyaknya pelanggan yang harus dilayani dengan batasan waktu dan jumlah

permintaan/muatan yang tidak selalu sama, mengakibatkan ada kalanya pengambilan/pengiriman barang tidak tepat waktu, kendaraan dalam keadaan tidak berfungsi secara penuh, serta ada kalanya kapasitas kendaraan tidak mencukupi sehingga diperlukan biaya tambahan untuk menyewa kendaraan tambahan. Hal tersebut juga dapat mengakibatkan rute dilalui lebih dari satu kali sehingga membuat jarak tempuh dan biaya perjalanan meningkat.

Berdasarkan uraian di atas, penentuan rute yang saat ini digunakan oleh Jogja Kurir Express masih belum efektif karena dengan adanya beberapa kendala tersebut, mengakibatkan kurang maksimalnya pihak Jogja Kurir Express dalam melakukan proses pelayanan jemput antar barang pelanggan. Oleh karena itu, diperlukan suatu metode dalam penentuan rute yang efektif sehingga menghasilkan total biaya perjalanan yang minimal dengan mempertimbangkan kendala yang ada. Pada penelitian ini, peneliti akan menggunakan algoritma *simulated annealing* dan *large neighborhood search* dalam menyelesaikan permasalahan di Jogja Kurir Express.

2. Pengumpulan Data

Dalam menyelesaikan permasalahan PDPTW, dibutuhkan beberapa data yang digunakan untuk mendapatkan solusi rute transportasi yang optimal. Data yang digunakan merupakan data yang berkaitan dengan aktivitas jemput antar barang pelanggan. Data diperoleh langsung dari bagian operasional tempat dilakukannya penelitian. Adapun data-data tersebut adalah data mengenai lokasi pelanggan jemput dan pelanggan antar, jumlah permintaan pelanggan, jenis dan kapasitas kendaraan, *time windows*, *service time*, data jarak antar pelanggan dalam bentuk

matriks jarak, dan waktu tempuh kendaraan dalam bentuk matriks waktu. Pada penelitian ini, digunakan data yang diambil pada tangal 31 Agustus 2014 .

a. Lokasi Pelanggan

Terdapat beberapa lokasi pelanggan jemput dan pelanggan antar yang dikunjungi oleh kendaraan tiap harinya. Berikut disajikan data lokasi pelanggan jemput dan pelanggan antar berdasarkan *transportation request* (lampiran 3). Untuk mengefisienkan penulisan, lokasi pelanggan diberi penomeran seperti pada Tabel 3.2 berikut.

Tabel 3.2 Lokasi Pelanggan Jemput dan Pelanggan Antar

No.	**Pelanggan Jemput**	**Alamat Pelanggan Jemput**
1	Mbak Lia	Jl. Rejosari No. 20, Sardonoharjo, Ngaglik, Sleman
2	Tk "Kusuka"	Jl. Wijilan No. 117, Yogyakarta
3	Fitriani P.	Jl. Gatot Subroto No.10, Mandingan, Ringinharjo, Bantul
4	Ummu Azka	Mejing Kidul RT 01/RW 08, Ambarketawang, Gamping,
5	Mama Moza	Perum Griya Taman Asri, Blok I No.334, Donoharjo, Sleman
6	Diah Retno	Baturan, Jambon, Sleman
7	Melanie S.	Tegalrejo No. 252, Yogyakarta
8	Iya Ratisa	Jl. Raya Pleret Km. 2,5, Perum Puri Sakinah No. C 5
No.	**Pelanggan Antar**	**Alamat Pelanggan Antar**
9	Bpk. Fauzan	Perum Mutiara Asri, Kecamatan Banguntapan
10	Larasati	Jl. Krangan No. 245,Yogyakarta
11	Apri	Perum Janti Buana Asri B 11, Banguntapan, Bantul
12	Retno S. W.	Warak Kidul, Sumberadi, M lati, Sleman
13	Ibu Henny	Perum Pemda DIY No. P 48, Banjardadap, Potorono, Bantul
14	Liza D. A.	Perum Griya Arga Permai H 9, Nogotirto, Sleman
15	Nurhayati	Salam, Dukuharjo, Salam, Magelang
16	Ika	Jl.Nangka 1, No.167, Karangnangka, Maguwoharjo, Sleman

b. Kendaraan

Dalam melakukan pelayanan jemput dan antar barang pelanggan, Jogja Kurir Express menggunakan kendaraan berupa sepeda motor. Jumlah kendaraan yang tersedia adalah 4 unit sepeda motor. Setiap kendaraan memiliki box tempat meletakkan barang pelanggan yang diletakkan di bagian belakang sepeda motor dengan kapasitas muatan 10 kg. Dengan demikian, kapasitas kendaraan yang digunakan dalam penelitian ini adalah $Q = 10$ kg.

c. Jumlah Permintaan Pelanggan

Jumlah permintaan atau muatan setiap pelanggan (q_i) berbeda-beda dan satuan yang digunakan untuk jumlah permintaan ini adalah kilogram (kg). q_i yang bernilai positif menyatakan banyaknya muatan yang harus dijemput dari pelanggan i, sedangkan q_i yang bernilai negatif menyatakan banyaknya muatan yang harus diantar ke pelanggan i. Berikut disajikan Tabel 3.3 mengenai jumlah permintaan dari setiap pelanggan berdasarkan *transportation request* (lampiran 3).

Tabel 3.3 Data Jumlah Permintaan Pelanggan

Pelanggan Jemput	**Jumlah Permintaan**	**Pelanggan Antar**	**Jumlah Permintaan**
1	2	9	–2
2	8	10	–8
3	5	11	–5
4	4	12	–4
5	8	13	–8
6	5	14	–5
7	6	15	–6
8	7	16	–7

d. Jarak

Data jarak yang diperlukan adalah jarak antara lokasi Jogja Kurir Express sebagai depot dengan lokasi pelanggan yang tersebar di D. I. Yogyakarta dan sekitarnya, serta jarak antar pelanggan tersebut. Perhitungan jarak dilakukan dengan menggunakan bantuan peta digital, yaitu dengan alat bantu *distance measurement tool* pada *google maps*. Alat bantu ini dapat dgunakan untuk menghitung jarak antar dua titik yang berada di peta.

Perhitungan jarak antar dua titik dilakukan dengan mengikuti alur jalan yang ada pada peta sehingga data jarak yang diperoleh dapat mendekati jarak aktual yang ditempuh kendaraan. Jarak antara dua titik tujuan ditentukan dengan pertimbangan jarak terdekat dan kondisi atau karakteristik jalan (tingkat kemacetan). Data jarak dari depot ke pelanggan dan jarak antar pelanggan ditampilkan dalam bentuk matriks. Selanjutnya, jarak tempuh dari titik A ke titik B diasumsikan sama dengan jarak tempuh dari titik B ke titik A. Matriks jarak antar depot dengan lokasi pelanggan dan jarak antar pelanggan dapat dilihat pada lampiran 4.

e. Waktu

Data waktu yang dibutuhkan dalam penelitian ini adalah *time windows* depot, *time windows* masing-masing pelanggan, *service time*, dan waktu tempuh kendaraan. Waktu operasional Jogja Kurir Express untuk melakukan pelayanan jemput antar barang pelanggan dimulai dari pukul 08.00 WIB s.d. pukul 19.00 WIB, maka *time windows* depot dalam satuan menit adalah $[0, 660]$. Waktu setiap

pelanggan berdasarkan *transportation request* (lampiran 3), dapat dilihat dalam Tabel 3.4.

Tabel 3.4 Waktu Pelanggan Jemput dan Pelanggan Antar

Pelanggan Jemput	Waktu Pelanggan Jemput	Pelanggan Antar	Waktu Pelanggan Antar
1	09.00 – 10.00	9	10.00 – 12.00
2	08.30 – 10.00	10	08.30 – 11.00
3	09.30 – 11.00	11	11.00 – 12.00
4	08.00 – 09.00	12	08.00 – 09.00
5	09.00 – 12.00	13	09.00 – 13.00
6	10.00 – 11.30	14	11.00 – 13.00
7	09.00 – 10.00	15	09.00 – 10.00
8	08.30 – 11.30	16	08.30 – 11.30

Service time atau waktu pelayanan pelanggan dapat dibagi menjadi waktu pemuatan barang (*loading*), waktu penurunan barang (*unloading*), dan waktu untuk urusan administrasi. Dari hasil wawancara pihak Jogja Kurir Express, diketahui bahwa rata-rata waktu pelayanan pelanggan adalah 10 menit.

Selanjutnya, untuk mengetahui waktu tempuh kendaraan, dilakukan penghitungan dengan cara membagi jarak tempuh kendaraan dengan rata-rata kecepatan kendaraan. Dari hasil wawancara dengan kurir/petugas Jogja Kurir Express, diperoleh kecepatan rata-rata kendaraan sebesar 40 km/jam. Berikut penghitungan waktu tempuh kendaraan :

$$\text{Waktu tempuh} = \left(\frac{Jarak\ (km)}{kecepatan\ rata{-}rata}\right) \times 60 \text{ menit.} \qquad (3.4)$$

Contoh :

Diketahui jarak antara lokasi pelanggan A dengan pelanggan B adalah 35,8 km. Jika kecepatan rata-rata kendaraan adalah 40 km/jam, maka waktu tempuh kendaraan adalah sebagai berikut,

Waktu tempuh = $(\frac{35{,}8\ km}{40\ km/jam})\ \times$ 60 menit = 53,7 menit = 54 menit (pembulatan)

Adapun hasil penghitungan waktu tempuh kendaraan disajikan dalam bentuk matrik waktu tempuh kendaraan yang dapat dilihat pada lampiran 5.

3. Pengolahan Data

Penyelesaian permasalahan PDPTW untuk mendapatkan solusi rute transportasi optimal pada Jogja Kurir Express dilakukan dengan mengolah data yang telah diperoleh dan dijelaskan sebelumnya pada subbab D.2. Permasalahan PDPTW tersebut akan diselesaikan dengan menggunakan Algoritma *simulated annealing* dan *large neighborhood search*.

Penggunaan algoritma *simulated annealing* dan *large neighborhood search* dalam menyelesaikan masalah pdptw dimulai dengan pembentukan solusi awal menggunakan algoritma *insertion heuristic* yang akan dijelaskan pada bagian D.3.a. Selanjutnya penyelesaian masalah menggunakan algoritma *simulated annealing* akan dijelaskan pada bagian D.3.b, dan penyelesaian masalah menggunakan algoritma *large neighborhood search* pada bagian D.3.c.

Namun sebelumnya, berikut akan disajikan Tabel 3.5 mengenai data pelanggan berdasarkan perolehan data yang telah dijelaskan pada subbab D.2 untuk membantu proses penyelesaian masalah PDPTW pada Jogja Kurir Express.

Tabel 3.5 Data Pelanggan

i	q_i	*Time Windows*		s_i	Jenis Pelayanan	
		a_i	b_i		p	d
0	0	0	660	0	0	0
1	2	60	120	10	0	9
2	8	30	120	10	0	10
3	5	90	180	10	0	11
4	4	0	60	10	0	12
5	8	60	240	10	0	13
6	5	120	210	10	0	14
7	6	60	120	10	0	15
8	7	30	210	10	0	16
9	–2	120	240	10	1	0
10	–8	30	180	10	2	0
11	–5	180	240	10	3	0
12	–4	0	60	10	4	0
13	–8	60	300	10	5	0
14	–5	180	360	10	6	0
15	–6	60	120	10	7	0
16	–7	30	210	10	8	0

Keterangan:

a. i menyatakan pelanggan ke *i* dengan 0 sebagai depot.

b. q_i adalah banyaknya permintaan dari pelanggan *i*.

q_i yang bernilai positif menyatakan banyaknya muatan yang harus dijemput dari pelanggan *i*, sedangkan q_i yang bernilai negatif menyatakan banyaknya muatan yang harus diantar ke pelanggan *i*.

c. a_i dan b_i adalah *time windows* pelanggan dalam satuan menit.

d. s_i adalah lamanya waktu pelayanan pada pelanggan *i*.

e. p dan d memberikan keterangan apakah pelanggan i merupakan pelanggan jemput atau pelanggan antar.

Jika $p = 0$, maka pelanggan i adalah pelanggan jemput dan permintaan/muatan harus diantar ke pelanggan d. Hal ini berlaku juga sebaliknya.

Berikut akan disajikan pula daftar urutan pasangan pelanggan jemput dan pelanggan antar berdasarkan *transportation request* (lampiran 3) dan matriks jarak (lampiran 4) pada Tabel 3.6.

Tabel 3.6 Daftar Urutan Pelanggan Jemput dan Pelanggan Antar

No.	Pelanggan Jemput	Pelanggan Antar
1	1	9
2	2	10
3	3	11
4	4	12
5	5	13
6	6	14
7	7	15
8	8	16

a. Pembentukan Solusi Awal Menggunakan Algoritma *Insertion Heuristic*

Pembentukan solusi awal dalam penyelesaian masalah PDPTW di Jogja Kurir Express dilakukan dengan menggunakan algoritma *insertion heuristic*. Pada algoritma ini, pemilihan pelanggan pertama untuk masuk ke dalam rute dilakukan berdasar pada jarak terjauh pelanggan jemput dari depot. Selanjutnya dilakukan pemilihan pasangan pelanggan untuk disisipkan ke dalam rute berdasar keterurutan dalam daftar pelanggan. Untuk setiap pasangan pelanggan tersebut, akan dilakukan pencarian tempat penyisipan yang layak diantara busur penyisipan

pada rute. Selanjutnya akan dilakukan penghitungan biaya penyisipannya. Pasangan pelanggan pada rute yang memberikan biaya penyisipan paling kecil dan layak yang akan dipilih. Berikut langkah-langkah pembentukan solusi awal menggunakan algoritma *insertion heuristic.*

1. Pembentukan Rute Pertama

Pada pembentukan rute pertama, kendaraan mengawali perjalanan dari Jogja Kurir Express sebagai depot (0). Kemudian pelanggan antar berturut-turut dilayani setelah mengunjungi pelanggan jemput sebelumnya, hingga mengakhiri perjalanan kembali ke depot. Kendaraan dalam keadaan kosong, sehingga kapasitas kendaraan (Q) = 0 kg. Adapun langkah-langkah pembentukan rute pertama adalah sebagai berikut.

a. Pada langkah ini, kendaraan mengawali perjalanan dari Jogja Kurir Express sebagai depot (0). Kemudian dipilih pelanggan jemput yang memiliki jarak terjauh dari 0. Berdasarkan matriks jarak (lampiran 4), pelanggan jemput yang paling jauh dari 0 adalah pelanggan 1, yaitu dengan jarak 20,3 km. Maka, pelanggan 1 tersebut dimasukkan ke dalam rute beserta pasangannya yaitu pelanggan 9, sehingga rute yang terbentuk adalah 0 – 1 – 9 – 0. Selanjutnya dilakukan uji kelayakan sebagai berikut.

   1) Kendaraan meninggalkan 0 dalam keadaan kosong tanpa muatan, $Y_0 = 0$. Setelah mengunjungi pelanggan 1, muatan kendaraan adalah 2 kg, $Y_1 = q_1 = 2$ kg. Kemudian kendaraan menuju pelanggan 9 untuk mengantar muatan dari pelanggan 1, sehingga $Y_9 = q_1 - q_9 =$

$2 - 2 = 0$, dan kendaraan kembali ke 0 dalam keadaan kosong tanpa muatan.

Terlihat bahwa rute $0 - 1 - 9 - 0$ memenuhi kendala kapasitas karena muatan total kendaraan setelah mengunjungi pelanggan 1 dan 9 lebih kecil dari kapasitas kendaraan, yaitu $Y_9 = 0$ kg < $Q = 10$ kg.

2) Pelayanan pada pelanggan 1 dimulai pada waktu $T_1 = 60$ menit karena $a_1 = 60$. Jadi, meskipun $w_1 = 30$, kendaraan harus menunggu hingga waktu $T_1 = 60$ untuk memulai pelayanan. Berdasarkan matriks waktu tempuh kendaraan (lampiran 5) dan persamaan 2.9, kendaraan akan meninggalkan pelanggan 1 menuju pelanggan 9 pada waktu $T_1 + s_1 + t_{1,9} = 60 + 10 + 40 = 110$ menit, dan akan memulai pelayanan pada pelanggan 9 pada waktu $T_9 = 120$ karena $a_9 = 120$. Setelah itu, kendaraan meninggalkan pelanggan 9 menuju 0 pada waktu $T_9 + s_9 + t_{9,0} = 120 + 10 + 10 = 140$ menit, maka $T_0 = 140$ menit

Terlihat bahwa rute $0 - 1 - 9 - 0$ memenuhi kendala *time windows* karena kendaraan tiba dan memulai pelayanan pada pelanggan 1 dan 9 sebelum waktu akhir pelanggan 1 dan 9 yaitu $b_1 = 120$ dan $b_9 = 240$, serta kembali ke depot sebelum waktu akhir depot yaitu $b_0 = 660$.

3) Pada rute $0 - 1 - 9 - 0$, dapat dilihat bahwa pelayanan jemput pada pelanggan 1 dilakukan terlebih dahulu sebelum pelayanan antar pada pelanggan 9 (*precedence*) dengan kendaraan yang sama (*pairing*).

Berdasarkan 1), 2), dan 3), rute $0-1-9-0$ memenuhi kendala kapasitas, *time windows*, *precedence* dan *pairing*, maka rute $0-1-9-0$ dianggap layak. Dari rute $0-1-9-0$, diperoleh total jarak tempuh kendaraan $d_{01}+d_{19}+d_{90}=20{,}3+26{,}8+6{,}5=53{,}6$ km.

b. Pada langkah ini, dilakukan penyisipan pasangan pelanggan yang belum dilayani untuk masuk ke dalam rute $0-1-9-0$, yaitu pasangan pelanggan 2 dan 10, 3 dan 11, 4 dan 12, 5 dan 13, 6 dan 14, 7 dan 15, serta 8 dan 16. Berdasarkan urutan pada daftar pelanggan (Tabel 3.6), maka dipilih pelanggan 2 dan 10 untuk disisipkan ke dalam rute. Kemudian akan dilakukan pencarian tempat penyisipan yang layak untuk pelanggan 2 dan 10, yaitu yang memenuhi kendala kapasitas, *time windows*, *precedence* dan *pairing*. Selanjutnya dilakukan penghitungan biaya penyisipan berdasarkan kriteria ketiga, persamaan (2.23) – (2.25), dengan menggunakan parameter $\mu=1$, $\alpha_1=0.5$, $\alpha_2=0.5$, dan $\alpha_3=0$ (Solomon, 1987). Langkah-langkah penyisipan pelanggan 2 dan 10 ke dalam rute $0-1-9-0$ adalah sebagai berikut.

   1) Pertama, dilakukan penyisipan pelanggan 2 ke dalam rute $0-1-9-0$. Rute yang mungkin terbentuk dengan penyisipan pelanggan 2 adalah $0-2-1-9-0, 0-1-2-9-0$, dan $0-1-9-2-0$. Pada setiap rute tersebut akan dilakukan uji kelayakan untuk kendala kapasitas dan *time windows*, sedangkan

kendala *precedence* dan *pairing* akan dipenuhi saat pelanggan 10 disisipkan ke dalam rute.

Rute 0 – 2 – 1 – 9 – 0

(a) Kendaraan meninggalkan 0 dalam keadaan kosong tanpa muatan, $Y_0 = 0$. Setelah mengunjungi pelanggan 2, muatan kendaraan adalah 8 kg, $Y_2 = q_2 = 8$ kg. Kemudian kendaraan menuju pelanggan 1 untuk mengambil muatan, sehingga $Y_1 = q_2 + q_1 = 8 + 2 = 10$ kg, lalu kendaraan menuju pelanggan 9 untuk mengantar muatan dari pelanggan 1 sehingga $Y_9 = Y_1 - q_9 = 10 - 2 = 8$ kg, dan kendaraan kembali ke 0 dengan kapasitas 8 kg.

Terlihat bahwa rute 0 – 2 – 1 – 9 – 0 memenuhi kendala kapasitas karena muatan total kendaraan setelah meninggalkan pelanggan 2, 1, dan 9 lebih kecil dari kapasitas kendaraan, yaitu $Y_9 = 8$ kg $< Q = 10$ kg.

(b) Pelayanan pada pelanggan 2 dimulai pada waktu $T_2 = 30$ karena $a_2 = 30$. Jadi, meskipun $w_2 = 5$, kendaraan harus menunggu hingga waktu $T_2 = 30$ untuk memulai pelayanan. Berdasarkan matriks waktu tempuh kendaraan (Lampiran 4) dan persamaan 2.9, kendaraan akan meninggalkan pelanggan 2 menuju pelanggan 1 pada waktu $T_2 + s_2 + t_{21} = 30 + 10 + 26 = 66$ menit, dan langsung memulai pelayanan pada pelanggan 1 pada waktu $T_1 = 66$ karena $a_6 = 60$. Setelah itu, kendaraan

meninggalkan pelanggan 1 menuju pelanggan 9 pada waktu $T_1 + s_1 + t_{19} = 66 + 10 + 40 = 116$ menit, dan memulai pelayanan pada pelanggan 9 pada waktu $T_9 = 120$ karena $a_9 = 120$. Kendaraan akan meninggalkan pelanggan 9 menuju 0 pada waktu $T_9 + s_9 + t_{90} = 120 + 10 + 10 = 140$ menit , maka $T_0 = 140$ menit

Terlihat bahwa rute 0 – 2 – 1 – 9 – 0 memenuhi kendala *time windows* karena kendaraan tiba dan memulai pelayanan pada pelanggan 2, 1 dan 9 sebelum waktu akhir pelanggan 2, 1 dan 9 yaitu $b_2 = 120$, $b_1 = 120$, dan $b_9 = 240$, serta kembali ke depot sebelum waktu akhir depot yaitu $b_0 = 660$.

Berdasarkan (a) dan (b), rute 0 – 2 – 1 – 9 – 0 memenuhi kendala kapasitas dan *time windows*, maka rute 0 – 2 – 1 – 9 – 0 masuk dalam kategori layak. Selanjutnya dilakukan perhitungan biaya penyisipan pelanggan 2 diantara 0 dan pelanggan 1 sebagai berikut.

$c_{11}(0,2,1) = d_{02} + d_{21} - \mu\, d_{01}$

$= 3{,}1 + 17{,}5 - 1.\ 20{,}3 = 0{,}3.$

$c_{12}(0,2,1) = T'_1 - T_1$

$= 66 - 60 = 6.$

$c_{13}(0,2,1) = b_2 - T_2$

$= 120 - 30 = 90.$

c_1 (0,2,1) = $\alpha_1 c_{11}$(0,2,1) + $\alpha_2 c_{12}$(0,2,1) + $\alpha_3 c_{13}$(0,2,1)

= 0,5 . 0,3 + 0,5 . 6 + 0 . 90 = 3,15.

c_2 (0,2,1) = c_1(0,2,1) = 3,15.

Dari perhitungan di atas, diperoleh biaya penyisipan pelanggan 2 diantara 0 dan pelanggan 1 adalah 3,15. Dengan demikian, rute 0 – 2 – 1 – 9 – 0 adalah layak dan memiliki biaya penyisipan 3,15.

Rute 0 – 1– 2 – 9 – 0

(a) Kendaraan meninggalkan 0 dalam keadaan kosong tanpa muatan, $Y_0 = 0$. Setelah mengunjungi pelanggan 1, muatan kendaraan adalah 2 kg, $Y_1 = q_1 = 2$ kg. Kemudian kendaraan menuju pelanggan 2 untuk mengambil muatan dari pelanggan 2, sehingga $Y_2 = Y_1 + q_2 = 2 + 8 = 10$ kg, lalu kendaraan menuju pelanggan 9 untuk megantar muatan dari pelanggan 1 sehingga $Y_9 = Y_2 - q_9 = 10 - 2 = 8$ kg, dan kendaraan kembali ke 0 dengan kapasitas 8 kg.

Terlihat bahwa rute 0 – 1 – 2 – 9 – 0 memenuhi kendala kapasitas karena muatan total kendaraan setelah meninggalkan pelanggan 1, 2, dan 9 lebih kecil dari kapasitas kendaraan, yaitu yaitu $Y_9 = 8$ kg $< Q = 10$ kg.

(b) Pelayanan pada pelanggan 1 dimulai pada waktu $T_1 = 60$ karena $a_1 = 60$. Jadi, meskipun $w_1 = 30$, kendaraan harus menunggu hingga waktu $T_1 = 60$ untuk memulai pelayanan. Berdasarkan

matriks waktu tempuh kendaraan (lampiran 5) dan persamaan 2.9, kendaraan akan meninggalkan pelanggan 1 menuju pelanggan 2 pada waktu $T_1 + s_1 + t_{12} = 60 + 10 + 26 = 96$ menit, dan langsung memulai pelayanan pada pelanggan 2 pada waktu $T_2 = 96$ karena $a_2 = 30$. Setelah itu, kendaraan meninggalkan pelanggan 2 menuju pelanggan 9 pada waktu $T_2 + s_2 + t_{29} = 96 + 10 + 14 = 120$ menit, dan memulai pelayanan pada pelanggan 9 pada waktu $T_9 = 120$. Kendaraan akan meninggalkan pelanggan 9 menuju 0 pada waktu $T_9 + s_9 + t_{90} = 120 + 10 + 10 = 140$ menit, maka $T_0 = 140$ menit.

Terlihat bahwa rute 0 – 1 – 2 – 9 – 0 memenuhi kendala *time windows* karena kendaraan tiba dan memulai pelayanan pada pelanggan 1, 2, dan 9 sebelum waktu akhir pelanggan 1, 2, dan 9 yaitu $b_1 = 120$, $b_2 = 120$, dan $b_9 = 240$, serta kembali ke depot sebelum waktu akhir depot yaitu $b_0 = 660$.

Berdasarkan (a) dan (b), rute 0 – 1 – 2 – 9 – 0 memenuhi kendala kapasitas dan *time windows*, maka rute 0 – 1 – 2 – 9 – 0 masuk dalam kategori layak. Selanjutnya dilakukan penghitungan biaya penyisipan pelanggan 2 diantara pelanggan 1 dan pelanggan 9 sebagai berikut.

c_{11} (1,2,9) = $d_{02} + d_{21} - \mu\, d_{01}$

= 17,5 + 9,4 – 1. 26,8 = 0,1.

c_{12} (1,2,9) = $T'_9 - T_9$

= 120 – 120 = 0.

c_{13} (1,2,9) = $b_2 - T_2$

= 120 – 96 = 24.

c_1 (1,2,9) = $\alpha_1\, c_{11}$(1,2,9) + $\alpha_2\, c_{12}$(1,2,9) + $\alpha_3\, c_{13}$(1,2,9)

= 0,5 . 0,1 + 0,5 . 0 + 0 . 24 = 0,05.

c_2 (1,2,9) = c_1(1,2,9) = 0,05.

Dari perhitungan di atas, diperoeh biaya penyisipan pelanggan 2 diantara pelanggan 1 dan 9 adalah 0,05. Dengan demikian, rute 0 – 1 – 2 – 9 – 0 adalah layak dan memiliki biaya penyisipan 0,05.

Rute 0 – 1 – 9 – 2 – 0

(a) Kendaraan meninggalkan 0 dalam keadaan kosong tanpa muatan, $Y_0 = 0$. Setelah mengunjungi pelanggan 1, muatan kendaraan adalah 2 kg, $Y_1 = q_1 = 2$ kg. Kemudian kendaraan menuju pelanggan 9 untuk mengantar muatan dari pelanggan 1 sehingga $Y_9 = Y_1 - q_9 = 2 - 2 = 0$ kg, lalu kendaraan menuju pelanggan 2 untuk mengambil muatan dari pelanggan 2 sehingga $Y_2 = Y_2 + q_2 = 0 + 8 = 8$ kg, dan kendaraan kembali ke 0 dengan muatan 8 kg.

Terlihat bahwa rute 0 – 1 – 9 – 2 – 0 memenuhi kendala kapasitas karena muatan total kendaraan setelah meninggalkan

pelanggan 1, 9, dan 2 lebih kecil dari kapasitas kendaraan, yaitu yaitu $Y_2 = 8$ kg < $Q = 10$ kg.

(b) Pelayanan pada pelanggan 1 dimulai pada waktu $T_1 = 60$ karena $a_1 = 60$. Jadi, meskipun $w_1 = 30$, kendaraan harus menunggu hingga waktu $T_1 = 60$ untuk memulai pelayanan. Berdasarkan matriks waktu tempuh kendaraan (lampiran 5) dan persamaan 2.9, kendaraan akan meninggalkan pelanggan 1 menuju pelanggan 9 pada waktu $T_1 + s_1 + t_{19} = 60 + 10 + 40 = 110$ menit, dan memulai pelayanan pada pelanggan 9 pada waktu $T_9 = 120$ karena $a_9 = 120$. Setelah itu, kendaraan meninggalkan pelanggan 9 menuju pelanggan 2 pada waktu $T_9 + s_9 + t_{92} = 120 + 10 + 14 = 144$ menit, dan memulai pelayanan pada pelanggan 2 pada waktu $T_2 = 144$. Namun, karena $T_2 = 144$ melebihi waktu akhir pelanggan 2, yaitu $b_2 = 120$, maka penyisipan pelanggan 2 pada rute 0 – 1 – 9 – 2 – 0 tidak layak.

Berdasarkan (a) dan (b), rute 0 – 1 – 9 – 2 – 0 memenuhi kendala kapasitas, tetapi tidak memenuhi kendala *time windows*, sehingga rute 0 – 1 – 9 – 2 – 0 tidak termasuk dalam kategori layak.

Berdasarkan uji kelayakan yang telah dilakukan untuk penyisipan pelanggan 2 ke dalam rute 0 – 1 – 9 – 0 di atas, rute yang memenuhi kendala kapasitas dan *time windows* dari ketiga rute yang mungkin adalah rute 0 – 2 – 1 – 9 – 0 dan 0 – 1 – 2 – 9 – 0.

Diantara kedua rute tersebut, biaya penyisipan terkecil diperoleh dari rute 0 – 1 – 2 – 9 – 0. Dengan demikian, rute yang terbentuk saat ini adalah adalah rute $0-1-2-9-0$. Selanjutnya akan disisipkan pelanggan 10 ke dalam rute $0-1-2-9-0$ yang akan dijelaskan pada langkah berikut ini.

2) Kedua, akan dilakukan penyisipan pelanggan 10 ke dalam rute $0-1-2-9-0$ untuk melengkapi kendala *precedence* dan *pairing*. Rute yang mungkin terbentuk dengan penyisipan pelanggan 10 adalah $0-1-2-10-9-0$ dan $0-1-2-9-10-0$. Pada setiap rute tersebut akan dilakukan uji kelayakan untuk kendala kapasitas, *time windows*, *precedence,* dan *pairing.*

Rute 0 – 1 – 2 – 10 – 9 – 0

(a) Kendaraan meninggalkan 0 dalam keadaan kosong tanpa muatan, $Y_0 = 0$. Setelah mengunjungi pelanggan 1, muatan kendaraan adalah 2 kg, $Y_1 = q_1 = 2$ kg. Kemudian kendaraan menuju pelanggan 2 untuk mengambil muatan dari pelanggan 2 sehingga $Y_2 = Y_1 + q_2 = 2 + 8 = 10$ kg, lalu kendaraan menuju pelanggan 10 untuk megantar muatan dari pelanggan 2 sehingga $Y_{10} = Y_2 - q_2 = 10 - 8 = 2$ kg, dan kendaraan menuju pelanggan 9 untuk megantar muatan dari pelanggan 1 sehingga $Y_9 = Y_{10} - q_1 = 2 - 2 = 0$ kg, hingga akhirnya kendaraan kembali lagi ke 0 dengan tanpa muatan.

Terlihat bahwa rute $0 - 1 - 2 - 10 - 9 - 0$ memenuhi kendala kapasitas karena muatan total kendaraan setelah meninggalkan pelanggan 1, 2, 10, dan 9 lebih kecil dari kapasitas kendaraan, yaitu $Y_9 = 0$ kg $< Q = 10$ kg.

(b) Pelayanan pada pelanggan 1 dimulai pada waktu $T_1 = 60$ karena $a_1 = 60$. Jadi, meskipun $w_1 = 30$, kendaraan harus menunggu hingga waktu $T_1 = 60$ untuk memulai pelayanan. Berdasarkan matriks waktu tempuh kendaraan (lampiran 5) dan persamaan 2.9, kendaraan akan meninggalkan pelanggan 1 menuju pelanggan 2 pada waktu $T_1 + s_1 + t_{12} = 60 + 10 + 26 = 96$ menit, dan langsung memulai pelayanan pada pelanggan 2 pada waktu $T_2 = 96$ karena $a_2 = 30$. Setelah itu, kendaraan meninggalkan pelanggan 2 menuju pelanggan 10 pada waktu $T_2 + s_2 + t_{210} = 96 + 10 + 6 = 112$ menit, dan langsung memulai pelayanan pada pelanggan 10 pada waktu $T_{10} = 112$ karena $a_{10} = 30$. Kendaraan akan meninggalkan pelanggan 9 menuju pelanggan 10 pada waktu $T_9 + s_9 + t_{910} = 112 + 10 + 19 = 141$ menit, dan langsung memulai pelayanan pada pelanggan 9 pada waktu $T_9 = 141$ karena $a_9 = 120$. Kendaraan akan meninggalkan pelanggan 9 menuju depot pada waktu $T_9 + s_9 + t_{90} = 141 + 10 + 10 = 161$ menit, maka $T_0 = 161$ menit.

Terlihat bahwa rute $0-1-2-10-9-0$ memenuhi kendala *time windows* karena kendaraan tiba dan memulai pelayanan pada pelanggan 1, 2, 10, dan 9 sebelum waktu akhir pelanggan 1, 2, 10, dan 9 yaitu $b_1 = 120$, $b_2 = 120$, $b_{10} = 180$ dan $b_9 = 240$, serta kembali ke depot sebelum waktu akhir depot yaitu $b_0 = 660$.

(c) Pada rute $0-1-2-10-9-0$, dapat dilihat bahwa pelayanan jemput pada pelanggan 1 dilakukan terlebih dahulu sebelum pelayanan antar pada pelanggan 9 (*precedence*) dengan kendaraan yang sama (*pairing*), begitu juga dengan pelanggan 2 dan 10.

Berdasarkan (a), (b), dan (c), rute $0-1-2-10-9-0$ memenuhi kendala kapasitas, *time windows*, *precedence* dan *pairing*, maka rute $0-1-2-10-9-0$ masuk dalam kategori layak. Selanjutnya dilakukan penghitungan biaya penyisipan pelanggan 10 diantara pelanggan 2 dan 9 sebagai berikut.

$c_{11}(2,10,9) = d_{210} + d_{109} - \mu\, d_{29}$

$= 4{,}3 + 11{,}8 - 1.\ 9{,}4 = 6{,}7.$

$c_{12}(2,10,9) = T'_9 - T_9$

$= 141 - 120 = 21.$

$c_{13}(2,10,9) = b_{10} - T_{10} = 180 - 112$

$= 68.$

$c_1(2,10,9) = \alpha_1 c_{11}(2,10,9) + \alpha_2 c_{12}(2,10,9) + \alpha_3 c_{13}(2,10,9)$

$= 0{,}5 \,.\, 6{,}7 + 0{,}5 \,.\, 21 + 0 \,.\, 68 = 13{,}85.$

$c_2(2,10,9) = c_1(2,10,9) = 13{,}85.$

Dari perhitungan di atas, diperoleh biaya penyisipan pelanggan 10 diantara pelanggan 2 dan 9 adalah 13,85. Dengan demikian, rute 0 – 1 – 2 – 10 – 9 – 0 adalah layak dan memiliki biaya penyisipan 13,85.

Rute 0 – 1 – 2 – 9 – 10 – 0

(a) Kendaraan meninggalkan 0 dalam keadaan kosong tanpa muatan, $Y_0 = 0$. Setelah mengunjungi pelanggan 1, muatan kendaraan adalah 2 kg, $Y_1 = q_1 = 2$ kg. Kemudian kendaraan menuju pelanggan 2 untuk mengambil muatan dari pelanggan 2, sehingga $Y_2 = Y_1 + q_2 = 2 + 8 = 10$kg, lalu kendaraan menuju pelanggan 9 untuk mengantar muatan dari pelanggan 1, sehingga $Y_9 = Y_2 - q_1 = 10 - 2 = 8$ kg. Kendaraan menuju pelanggan 10 untuk mengantar muatan dari pelanggan 2, sehingga $Y_{10} = Y_9 - q_2 = 8 - 8 = 0$ kg, dan kendaraan kembali ke 0 dengan tanpa muatan.

Terlihat bahwa rute 0 – 1 – 2 – 9 – 10 – 0 memenuhi kendala kapasitas karena muatan total kendaraan setelah meninggalkan pelanggan 1, 2, 9, dan 10 lebih kecil dari kapasitas kendaraan, yaitu $Y_{10} = 0$ kg < $Q = 10$ kg.

(b) Pelayanan pada pelanggan 1 dimulai pada waktu $T_1 = 60$ karena $a_1 = 60$. Jadi, meskipun $w_1 = 30$, kendaraan harus menunggu hingga waktu $T_1 = 60$ untuk memulai pelayanan. Berdasarkan matriks waktu tempuh kendaraan (lampiran 5) dan persamaan 2.9, kendaraan akan meninggalkan pelanggan 1 menuju pelanggan 2 pada waktu $T_1 + s_1 + t_{12} = 60 + 10 + 26 = 96$ menit, dan langsung memulai pelayanan pada pelanggan 2 pada waktu $T_2 = 96$ karena $a_2 = 30$. Setelah itu, kendaraan meninggalkan pelanggan 2 menuju pelanggan 9 pada waktu $T_2 + s_2 + t_{29} = 96 + 10 + 14 = 120$ menit, dan langsung memulai pelayanan pada pelanggan 9 pada waktu $T_9 = 120$ karena $a_9 = 120$. Kendaraan akan meninggalkan pelanggan 9 menuju pelanggan 10 pada waktu $T_9 + s_9 + t_{910} = 120 + 10 + 19 = 149$ menit, dan langsung memulai pelayanan pada pelanggan 10 pada waktu $T_{10} = 149$ karena $a_{10} = 149$. Kendaraan akan meninggalkan pelanggan 10 menuju depot pada waktu $T_{10} + s_{10} + t_{100} = 149 + 10 + 10 = 169$ menit, maka $T_0 = 169$ menit.

Terlihat bahwa rute 0 – 1 – 2 – 9 – 10 – 0 memenuhi kendala *time windows* karena kendaraan tiba dan memulai pelayanan pada pelanggan 1, 2, 9, dan 10 sebelum waktu akhir pelanggan 1, 2, 9, dan 10 yaitu $b_1 = 120$, $b_2 = 120$, $b_9 = 240$,

dan $b_{10} = 180$, serta kembali ke depot sebelum waktu akhir depot yaitu $b_0 = 660$.

(c) Pada rute 0 – 1 – 2 – 9 – 10 – 0, dapat dilihat bahwa pelayanan jemput pada pelanggan 1 dilakukan terlebih dahulu sebelum pelayanan antar pada pelanggan 9 (*precedence*) dengan kendaraan yang sama (*pairing*), begitu juga dengan pelanggan 2 dan 10.

Berdasarkan (a), (b), dan (c), rute 0 – 1 – 2 – 9 – 10 – 0 memenuhi kendala kapasitas, *time windows*, *precedence* dan *pairing*, maka rute 0 – 1 – 2 – 9 – 10 – 0 masuk dalam kategori layak. Selanjutnya dilakukan penghitungan biaya penyisipan pelanggan 10 diantara pelanggan 9 dan 0 sebagai berikut..

$c_{11}(9,10,0) = d_{910} + d_{100} - \mu\, d_{90}$

$= 11{,}8 + 6{,}6 - 1.\ 6{,}5 = 11{,}9.$

$c_{12}(9,10,0) = T'_0 - T_0$

$= 169 - 140 = 29.$

$c_{13}(9,10,0) = b_{10} - T_{10} = 180 - 149 = 31.$

$c_1(9,10,0) = \alpha_1\, c_{11}(9,10,0) + \alpha_2\, c_{12}(9,10,0) + \alpha_3\, c_{13}(9,10,0)$

$= 0{,}5 \,.\, 11{,}9 + 0{,}5 \,.\, 29 + 0 \,.\, 31 = 20{,}45.$

$c_2(9,10,0) = c_1(9,10,0) = 20{,}45.$

Dari perhitungan di atas, diperoleh biaya penyisipan pelanggan 10 diantara pelanggan 9 dan 0 adalah 20,45. Dengan demikian, rute

$0 – 1 – 2 – 9 – 10 – 0$ adalah layak dan memiliki biaya penyisipan 20,45.

Berdasarkan uji kelayakan yang telah dilakukan untuk penyisipan pelanggan 10 ke dalam rute $0 – 1 – 2 – 9 – 0$ di atas, didapatkan rute $0 – 1 – 2 – 10 – 9 – 0$ dan $0 – 1 – 2 – 9 – 10 – 0$ memenuhi solusi layak, yaitu memenuhi kendala kapasitas, *time windows*, *precedence* dan *pairing*. Diantara kedua rute tersebut, biaya penyisipan terkecil diperoleh dari rute $0 – 1 – 2 – 10 – 9 – 0$.

Dengan demikian, rute baru yang terbentuk dari penyisipan pelanggan 2 dan 10 ke dalam rute $0 – 1 – 9 – 0$ adalah rute $0 – 1 – 2 – 10 – 9 – 0$, dengan total jarak tempuh kendaraan $d_{01} + d_{12} + d_{210} + d_{109} + d_{90} = 20{,}3 + 17{,}5 + 4{,}3 + 12{,}8 + 6{,}5 = 61{,}4$ km. Selanjutnya akan dilakukan penyisipan pasangan pelanggan lainnya untuk masuk ke dalam rute $0 – 1 – 2 – 10 – 9 – 0$. Penyisipan pasangan pelanggan tersebut akan dijelaskan pada langkah berikut ini.

c. Pada langkah ini, dilakukan penyisipan pasangan pelanggan yang belum dilayani untuk masuk ke dalam rute $0 – 1 – 2 – 10 – 9 – 0$, yaitu pelanggan 3 dan 11, 4 dan 12, 5 dan 13, 6 dan 14, 7 dan 15, serta 8 dan 16.

Berdasarkan urutan pada daftar pelanggan (Tabel 3.6), maka dipilih pasangan pelanggan 3 dan 11 untuk disisipkan ke dalam rute. Dengan langkah yang sama seperti yang telah dijelaskan sebelumnya saat

menyisipkan pelanggan 2 dan 10 pada rute 0 – 1 – 9 – 0 (langkah b), didapat rute yang memenuhi kendala dan memiliki biaya penyisipan minimal, yaitu rute 0 – 1 – 2 – 10 – 3 – 9 – 11 – 0 dengan total jarak tempuh kendaraan $d_{01} + d_{12} + d_{210} + d_{103} + d_{39} + d_{911} + d_{110} = 20{,}3 + 17{,}5 + 4{,}3 + 13{,}6 + 14{,}5 + 1{,}9 + 7{,}3 = 79{,}4$ km.

d. Pada langkah ini, dilakukan penyisipan pasangan pelanggan yang belum dilayani untuk masuk ke dalam rute 0 – 1 – 2 – 10 – 3 – 9 – 11 – 0, yaitu pelanggan 4 dan 12, 5 dan 13, 6 dan 14, 7 dan 15, serta 8 dan 16.

Berdasarkan urutan pada daftar pelanggan (Tabel 3.6), maka dipilih pasangan pelanggan 4 dan 12 untuk disisipkan ke dalam rute. Namun, karena penyisipan pasangan pelanggan 4 dan 12 pada rute 0 – 1 – 2 – 10 – 3 – 9 – 11 – 0 tidak memenuhi kendala, maka dipilih pasangan pelanggan berikutnya yang memenuhi kendala yanitu pasangan pelanggan 5 dan 13. Dengan langkah yang sama seperti yang telah dijelaskan sebelumnya saat menyisipkan pelanggan 2 dan 10 pada rute 0 – 1 – 9 – 0 (langkah b), didapat rute yang memenuhi kendala dan memiliki biaya penyisipan minimal, yaitu rute 0 – 1 – 2 – 10 – 3 – 9 – 11 – 5 – 13 – 0 dengan total jarak tempuh kendaraan $d_{01} + d_{12} + d_{210} + d_{103} + d_{39} + d_{911} + d_{115} + d_{513} + d_{130} = 20{,}3 + 17{,}5 + 4{,}3 + 13{,}6 + 14{,}5 + 1{,}9 + 20{,}7 + 28{,}1 + 6{,}8 = 127{,}8$ km.

Selanjutnya pasangan pelanggan yang belum masuk ke dalam rute adalah pelanggan 4 dan 12, 6 dan 14, 7 dan 15, serta 8 dan 16. Penyisipan pasangan pelanggan 4 dan 12 pada rute 0 – 1 – 2 – 10 – 3 – 9 – 11 – 5 –

13 – 0 tidak dapat dilakukan karena tidak memenuhi kendala. Begitu juga dengan penyisipan pasangan pelanggan 6 dan 14, 7 dan 15, serta 8 dan 16. Dengan demikian, pada rute pertama terbentuk rute 0 – 1 – 2 – 10 – 3 – 9 – 11 – 5 – 13 – 0 dengan total jarak tempuh kendaraan 127,8 km. Selanjutnya untuk pasangan pelanggan yang belum dilayani, maka akan dibentuk rute kedua sebagai berikut.

2. Pembentukan Rute Kedua

Pada pembentukan rute kedua, kendaraan mengawali perjalanan dari Jogja Kurir Express sebagai depot (0). Kemudian dipilih pelanggan jemput yang memiliki jarak terjauh dengan 0. Berdasarkan urutan pada daftar pelanggan (Tabel 3.6) dan matriks jarak (lampiran 4), diantara pelanggan jemput 4, 6, 7, dan 8, pelanggan jemput yang paling jauh dengan 0 adalah pelanggan 4, yaitu dengan jarak 9,4 km. Maka, pelanggan 4 dimasukkan ke dalam rute beserta pasangannya yaitu pelanggan 13, sehingga rute yang terbentuk adalah 0 – 4 – 12 – 0 .

Analog dengan pembentukan rute pertama, dilakukan penyisipan pasangan pelanggan yang belum dilayani untuk masuk ke dalam rute 0 – 4 – 12 – 0, yaitu pasangan pelanggan 6 dan 14, 7 dan 15, serta 8 dan 16. Diantara 3 pasangan pelanggan tersebut, penyisipan pelanggan yang layak adalah penyisipan pelanggan 6 dan 14. Adapun rute yang terbentuk dari penyisipan pelanggan 6 dan 14 dengan biaya penyisipan minimal adalah rute 0 – 4 – 12 – 6 – 14 – 0 . Dengan demikian, pada rute kedua terbentuk rute 0 – 4 – 12 – 6 – 14 – 0 dengan total jarak tempuh kendaran $d_{04} + d_{412} + d_{126} + d_{614} + d_{140} = 9{,}4 +$

$9{,}4 + 7{,}9 + 3{,}4 + 10{,}2 = 40{,}3$ km. Selanjutnya untuk pasangan pelanggan yang belum dilayani, maka akan dibentuk rute ketiga sebagai berikut.

3. Pembentukan Rute Ketiga

Pada pembentukan rute ketiga, kendaraan mengawali perjalanan dari Jogja Kurir Express sebagai depot (0). Kemudian dipilih pelanggan jemput yang memiliki jarak terjauh dengan 0. Berdasarkan urutan pada daftar pelanggan (Tabel 3.6) dan matriks jarak (lampiran 4), diantara pelanggan jemput 7, dan 8, pelanggan jemput yang paling jauh dengan 0 adalah pelanggan 8, yaitu dengan jarak 8,9 km. Maka, pelanggan 8 dimasukkan ke dalam rute beserta pasangannya yaitu pelanggan 16, sehingga rute yang terbentuk adalah $0 - 8 - 16 - 0$.

Analog dengan pembentukan rute pertama dan kedua, dilakukan penyisipan pasangan pelanggan yang belum dilayani untuk masuk ke dalam rute $0 - 8 - 16 - 0$, yaitu pelanggan 7 dan 15. Oleh karena penyisipan pelanggan 7 dan 15 tidak memenuhi kendal, maka penyisipan tidak dapat dilakukan. Dengan demikian, pada rute ketiga terbentuk rute $0 - 8 - 16 - 0$ dengan total jarak tempuh kendaraan $d_{08} + d_{816} + d_{160} = 8{,}9 + 19{,}4 + 11{,}5 = 39{,}8$ km. Selanjutnya untuk pasangan pelanggan yang belum dilayani, maka akan dibentuk rute keempat sebagai berikut.

4. Pembentukan Rute Keempat

Pada tahap ini, hanya pasangan pelanggan 7 dan 15 yang belum masuk ke dalam rute untuk dilayani, maka dibentuk rute $0 - 7 - 15 - 0$. Dengan demikian,

pada rute keempat terbentuk rute $0 - 7 - 15 - 0$ dengan total jarak tempuh kendaraan $d_{07} + d_{715} + d_{150} = 7{,}8 + 26{,}3 + 31{,}5 = 65{,}6$ km.

Berdasarkan perhitungan yang telah dilakukan menggunakan algoritma *insertion heuristic*, permasalahan PDPTW di Jogja Kurir Express menghasilkan sebuah solusi yang terdiri dari 4 rute yang melayani 16 pelanggan. Adapun rekapitulasi hasil penyelesaian masalah menggunakan algoritma *insertion heuristic* adalah sebagai berikut.

Tabel 3.7 Rekapitulasi Hasil Penyelesaian Masalah dengan Algoritma *Insertion Heuristic*

No.	Rute	Jarak (km)	Waktu Tempuh (menit)
1.	$0 - 1 - 2 - 10 - 3 - 9 - 11 - 5 - 13 - 0$	127,8	300
2.	$0 - 4 - 12 - 6 - 14 - 0$	40,3	100
3.	$0 - 8 - 16 - 0$	39,8	96
4.	$0 - 7 - 15 - 0$	65,6	166
Total		**283,7**	**662**

Pada Tabel 3.7, pembentukan rute menggunakan algoritma *insertion heuristic* menghasilkan 4 rute. Pada rute pertama, kendaraan melayani 8 pelanggan yang terdiri dari 4 pelanggan jemput dan 4 pelanggan antar. Pada rute ini, kendaraan menempuh perjalanan sejauh 127,8 km dengan waktu penyelesaian selama 300 menit. Pada rute yang kedua, kendaraan melayani 4 pelanggan yang terdiri dari 2 pelanggan jemput dan 2 pelanggan antar. Pada rute ini, kendaraan menempuh perjalanan sejauh 40,3 km dengan waktu penyelesaian 100 menit. Pada rute ketiga, kendaraan melayani 2 pelanggan yaitu sepasang pelanggan jemput dan antar. Pada rute ini, kendaraan menempuh perjalanan sejauh 39,8 km dengan waktu penyelesaian 96 menit, sedangkan pada rute keempat, kendaraan juga melayani

sepasang pelanggan jemput dan antar. Pada rute ini, kendaraan menempuh perjalanan sejauh 65,6 km dengan waktu penyelesaian 166 menit.

Solusi yang dihasilkan dari penggunaan algoritma *insertion heuristic* tersebut kemudian digunakan sebagai solusi awal dalam penyelesaian masalah PDPTW menggunakan algoritma *simulated annealing* dan *large neighborhood search* yang akan dijelaskan pada bagian berikutnya.

b. Penyelesaian Masalah Menggunakan Algoritma *Simulated Annealing*

Penyelesaian masalah PDPTW di Jogja Kurir Express dengan algoritma *simulated annealing* menggunakan solusi awal yang didapat dari solusi akhir algoritma *insertion heuristic*, yaitu σ = {0 – 1 – 2 – 10 – 3 – 9 – 11 – 5 – 13 – 0 , 0 – 4 – 12 – 6 – 14 – 0, 0 – 8 – 16 – 0, 0 – 7 – 15 - 0}. Parameter yang akan digunakan dalam penelitian ini adalah $H_0 = 100$, $H_t = 0.1$, $L_t = 2$, α = 0.5 dan parameter $\beta = 10$. Berikut langkah-langkah penyelesaian masalah PDPTW di Jogja Kurir Express menggunakan algoritma *simulated annealing*.

Iterasi pertama

1. Akan dibentuk sub-lingkungan dari σ dengan melakukan *pair relocation* terhadap pasangan pelanggan yang dipilih secara acak. Misalkan pasangan pelanggan yang terpilih adalah pasangan pelanggan 5 dan 13, maka pelanggan 5 dan 13 akan dipindahkan ke rute lain yang memenuhi kendala. Solusi sublingkungan yang terbentuk terdiri dari tiga solusi, yaitu sebagai berikut.

$\sigma'_a = \{0-1-2-10-3-9-11-0, 0-4-12-6-14-5-13-0, 0-8-16-0, 0-7-15-0\}$.

$\sigma'_b = \{0-1-2-10-3-9-11-0, 0-4-12-6-14-0, 0-8-16-5-13-0, 0-7-15-0\}$.

$\sigma'_c = \{0-1-2-10-3-9-11-0, 0-4-12-6-14-0, 0-8-16-0, 0-7-15-5-13-0\}$.

2. Selanjutnya dilakukan perhitungan nilai fungsi evaluasi dari solusi sub-lingkungan $e(\sigma'_i)$ berdasarkan persamaan (3.1). Solusi sublingkungan tersebut akan disusun sedemikian sehingga $e(\sigma'_1) \leq e(\sigma'_2) \leq e(\sigma'_3)$, dimana $e(\sigma'_1)$ merupakan solusi sublingkungan yang memiliki nilai fungsi evalusi terkecil.

   a. Untuk σ'_a, nilai $e(\sigma'_a)$ adalah

$$|\sigma'_a| = 4$$

$$-\sum_{r\in\sigma'_a} |r|^2 = -(6^2+6^2+2^2+2^2) = -(36+36+4+4) = -80$$

$$\sum_{r\in\sigma'_a} t(r) = 79{,}4 + 73{,}8 + 39{,}8 + 65{,}6 = 258{,}6.$$

   Dengan demikian $e(\sigma'_a) = \langle 4, -80, 258{,}6\rangle$.

   b. Untuk σ'_b, nilai $e(\sigma'_b)$ adalah

$$|\sigma'_b| = 4$$

$$-\sum_{r\in\sigma'_b} |r|^2 = -(6^2+4^2+4^2+2^2) = -(36+16+16+4) = -72$$

$$\sum_{r\in\sigma'_b} t(r) = 79{,}4 + 40{,}3 + 73{,}2 + 65{,}6 = 258{,}5.$$

   Dengan demikian $e(\sigma'_b) = \langle 4, -72, 258{,}5\rangle$.

c. Untuk σ_c', nilai $e(\sigma_c')$ adalah

$$|\sigma_c'| = 4$$

$$-\sum_{r\in\sigma_c'}|r|^2 = -(6+4^2+2^2+4^2) = -(36+16+4+16) = -72$$

$$\sum_{r\in\sigma_c'} t(r) = 79{,}4 + 40{,}3 + 39{,}8 + 85 = 244{,}5.$$

Dengan demikian $e(\sigma_c') = \langle 4, -72, 244{,}5\rangle$.

Berdasarkan perhitungan yang telah dilakukan di atas, diperoleh $e(\sigma_a') = \langle 4, -80, 258{,}6\rangle$, $e(\sigma_b') = \langle 4, -72, 258{,}5\rangle$, dan $e(\sigma_c') = \langle 4, -72, 244{,}5\rangle$. Dengan demikian, susunan solusi lingkungan yang terbentuk adalah $\langle \sigma_1', \sigma_2', \sigma_3'\rangle$ dengan $\sigma_1' = \sigma_a'$, $\sigma_2' = \sigma_c'$, dan $\sigma_3' = \sigma_b'$.

3. Kemudian dilakukan perhitungan nilai fungsi evaluasi dari solusi saat ini $e(\sigma)$ berdasarkan persamaan (3.1). Hasil dari $e(\sigma)$ akan dibandingkan dengan $e(\sigma_1')$. Langkah ini menjadi sebuah langkah yang merupakan modifikasi dari algoritma *simulated annealing* yang dilakukan oleh Bent dan Hentenryck, 2003. Biasanya, pemilihan solusi lingkungan dari algoritma *simulated annea*ling dilakukan secara acak. Adapun perhitungan nilai fungsi evaluasi dari solusi saat ini, $e(\sigma)$, adalah sebagai berikut

$$|\sigma| = 4$$

$$-\sum_{r\in\sigma_c'}|r|^2 = -(8+4^2+2^2+2^2) = -(64+16+4+4) = -88$$

$$\sum_{r\in\sigma_c'} t(r) = 127{,}8 + 40{,}3 + 39{,}8 + 65{,}6 = 273{,}5.$$

Dengan demikian $e(\sigma) = \langle 4, -88, 273{,}5\rangle$.

Berdasarkan perhitungan di atas, diperoleh $e(\sigma) = \langle 4, -88, 273{,}5\rangle$. Pada perhitungan sebelumnya telah didapatkan nilai $e(\sigma_1') = e(\sigma_a') = \langle 4, -80, 258{,}6\rangle$, maka berdasarkan *lexicographic ordering*, $\langle 4, -80, 258{,}6\rangle > \langle 4, -88, 273{,}5\rangle$ atau nilai $e(\sigma_1') > e(\sigma)$.

Oleh karena $e(\sigma_1') > e(\sigma)$, maka akan dicari sebuah bilangan random *r* dengan persamaan (3.2) berikut.

$$r = [random\ (0{,}1)^{\beta} \times s]$$
$$= (0{,}9)^{10} \times 3$$
$$= 1\,{,}05 \approx 1$$

Selanjutnya akan ditentukan nilai fungsi evaluasi $\Delta\sigma$ yaitu selisih antara nilai fungsi evaluasi sublingkungan bilangan random *r*, $e(\sigma'_r)$, dengan nilai fungsi evaluasi solusi saat ini $e(\sigma)$ berdasarkan persamaan (2.27) sebagai berikut.

$$\Delta\sigma = e(\sigma'_r) - e(\sigma)$$
$$= e(\sigma'_1) - e(\sigma)$$
$$= -80 - (-88)$$
$$= 8$$

Dari perhitungan di atas, diperoleh bahwa $\Delta\sigma > 0$.

Oleh karena $\Delta\sigma > 0$, maka akan diselidiki apakah σ'_1 dapat diterima sebagai solusi baru dengan $exp(-\Delta\sigma/H)$. Ditentukan sebuah bilangan P yaitu bilangan acak diantara 0 dan 1, misal ditentukan $P = 0{,}00002$. Kemudian, dicari nilai dari $exp(-\Delta\sigma/H)$, yaitu $exp(-\Delta\sigma/H) = exp(-16/100) = 0{,}852144$. Karena $0{,}00002 < 0{,}852144$ atau $P \leq exp(-\Delta\sigma/H)$, maka

solusi sublingkungan σ'_1 dapat diterima sebagai solusi baru. Dengan demikian, solusi saat ini menjadi σ ={0 – 1 – 2 – 10 – 3 – 9 – 11 – 0, 0 – 4 – 12 – 6 – 14 – 5 – 13 – 0, 0 – 8 – 16 – 0, 0 – 7 – 15 – 0} dengan total jarak tempuh kendaraan 258,6 km.

Iterasi kedua

Diketahui solusi saat ini adalah σ ={0 – 1 – 2 – 10 – 3 – 9 – 11 – 0, 0 – 4 – 12 – 6 – 14 – 5 – 13 – 0, 0 – 8 – 16 – 0, 0 – 7 – 15 – 0}.

1. Akan dibentuk sublingkungan dari σ dengan melakukan *pair relocation* terhadap pasangan pelanggan yang dipilih secara acak. Misalkan pasangan pelanggan yang terpilih adalah pelanggan 7 dan 15, maka pelanggan 7 dan 15 akan dipindahkan ke rute lain yang memenuhi kendala. Adapun sub-lingkungan yang terbentuk terdiri dari dua solusi, yaitu sebagai berikut.

   σ'_a = {0 – 1 – 2 – 10 – 3 – 9 – 11 – 0, 0 – 4 – 12 – 7 – 15 – 6 – 14 – 5 – 13 - 0, 0 – 8 – 16 – 0}.

   σ'_b = {0 – 1 – 2 – 10 – 3 – 9 – 11 – 0, 0 – 4 – 12 – 6 – 14 – 5 – 13 – 0, 0 – 7 – 15 – 8 – 16 – 0}.

2. Selanjutnya dilakukan perhitungan nilai fungsi evaluasi dari solusi sub-lingkungan $e(\sigma'_i)$ b erdasarkan persamaan (3.1). Solusi sub-lingkungan tersebut akan disusun sedemikian sehingga $e(\sigma'_1) \leq e(\sigma'_2)$, dimana $e(\sigma'_1)$ adalah solusi sub-lingkungan yang memiliki nilai fungsi evalusi terkecil. Berikut perhitungan nilai fungsi evaluasi dari solusi sub-lingkungan $e(\sigma'_i)$.

a. Untuk σ'_a, nilai $e(\sigma'_a)$ adalah

$$|\sigma'_a| = 3$$

$$-\sum_{r\in\sigma'_a} |r|^2 = -(6^2 + 8^2 + 2^2) = -(36 + 64 + 4) = -104$$

$$\sum_{r\in\sigma'_a} t(r) = 79{,}4 + 125{,}9 + 39{,}8 = 245{,}1.$$

Dengan demikian $e(\sigma'_a) = \langle 3, -104, 245{,}1\rangle$.

b. Untuk σ'_b, nilai $e(\sigma'_b)$ adalah

$$|\sigma'_b| = 3$$

$$-\sum_{r\in\sigma'_b} |r|^2 = -(6^2 + 6^2 + 4^2) = (36 + 36 + 16) = -88$$

$$\sum_{r\in\sigma'_b} t(r) = 79{,}4 + 73{,}8 + 98{,}9 = 252{,}1.$$

Dengan demikian $e(\sigma'_b) = \langle 4, -88, 252{,}1\rangle$.

Berdasarkan perhitungan nilai fungsi evalusi yang telah dilakukan di atas, diperoleh $e(\sigma'_a) = \langle 3, -104, 245{,}1\rangle$ dan $e(\sigma'_b) = \langle 3, -88, 252{,}1\rangle$. Dengan demikian, susunan solusi lingkungan yang terbentuk adalah $\langle \sigma'_1, \sigma'_2\rangle$ dengan $\sigma'_1 = \sigma'_a$ dan $\sigma'_2 = \sigma'_b$.

3. Kemudian dilakukan perhitungan nilai fungsi evaluasi dari solusi saat ini $e(\sigma)$ berdasarkan persamaan (3.1). Hasil dari $e(\sigma)$ akan dibandingkan dengan $e(\sigma'_1)$. Adapun perhitungan nlai fungsi evaluasi dari solusi saat ini $e(\sigma)$ adalah nilai $e(\sigma'_1)$ pada iterasi pertama, yaitu $e(\sigma) = e(\sigma'_1) = \langle 4, -80, 258{,}62\rangle$. Pada perhitungan yang dilakukan sebelumnya juga telah

didapatkan nilai $e(\sigma_1') = e(\sigma_a') = \langle 3, -104, 245{,}1 \rangle$, maka berdasarkan *lexicographic ordering*, diperoleh $\langle 3, -104, 245{,}1 \rangle < \langle 4, -80, 258{,}62 \rangle$ atau nilai $e(\sigma_1') < e(\sigma)$.

Oleh karena $e(\sigma_1') < e(\sigma)$, maka σ_1' diterima sebagai solusi baru. Dengan demikian, solusi saat ini menjadi σ ={0 – 1 – 2 – 10 – 3 – 9 – 11 – 0, 0 – 4 – 12 – 7 – 15 – 6 – 14 – 5 – 13 - 0, 0 – 8 – 16 – 0} dengan total jarak tempuh kendaraan 245,1 km.

Dari perhitungan yang telah dilakukan menggunakan algoritma *simulated annealing*, permasalahan PDPTW di Jogja Kurir Express menghasilkan sebuah solusi yang terdiri dari 3 rute yang melayani 16 pelanggan. Adapun rekapitulasi hasil penyelesaian masalah menggunakan *algoritma simulated annealing* adalah sebagai berikut.

Tabel 3.8 Rekapitulasi Hasil Penyelesaian Masalah dengan Algoritma *Simulated Annealing*

Iterasi ke-	**Rute ke-**	**Perjalanan**	**Jarak (km)**	**Waktu Tempuh (menit)**
I	1	0 – 1 – 2 – 10 – 3 – 9 – 11– 0	79,4	208
	2	0 – 4 – 12 – 6 – 14 – 5 – 13 – 0	73,8	160
	3	0 – 8 – 16 – 0	39,8	96
	4	0 – 7 – 15 – 0	65,6	166
Total			**258, 6**	**630**
II	1	0 – 1 – 2 – 10 – 3 – 9 – 11– 0	79,4	208
	2	0 – 4 – 12 – 7– 15 – 6 – 14 – 5 – 13 – 0	125,9	275
	3	0 – 8 – 16 – 0	39,8	96
Total			**245, 1**	**579**

Pada Tabel 3.8, pembentukan rute menggunakan algoritma *simulated annealing* menghasilkan 3 rute yang melayani 16 pelanggan dengan 2 kali iterasi. Pada iterasi pertama, terbentuk 4 rute dan menempuh perjalanan sejauh 258,6 km

dengan waktu penyelesaian selama 630 menit, sedangkan pada iterasi yang kedua, terbentuk 4 rute dan kendaraan menempuh perjalanan sejauh 245,1 km dengan waktu penyelesaian 579 menit.

c. Penyelesaian Masalah Menggunakan Algoritma *Large Neighborhood Search*

Dalam penyelesaian masalah PDPTW di Jogja Kurir Express dengan algoritma *large neighborhood search*, digunakan solusi awal yang didapat dari solusi akhir algoritma *insertion heuristic*, yaitu σ ={0 – 1 – 2 – 10 – 3 – 9 – 11 – 5 – 13 – 0 , 0 – 4 – 12 – 6 – 14 – 0, 0 – 8 – 16 – 0, 0 – 7 – 15 - 0}. Parameter yang digunakan dalam penelitian ini adalah $x = 2$ dan $l = 2$. Langkah-langkah penyelesaian masalah PDPTW di Jogja Kurir Express menggunakan algoritma *large neighborhood search* sebagai berikut.

Iterasi Pertama

1. Akan dibentuk solusi sublingkungan dari σ dengan menggunakan metode penghapuasan dan perbaikan.
   a. Metode penghapusan

      Misalkan pelanggan yang terpilih adalah pasangan pelanggan 5 – 13 dan pelanggan 4 – 12, maka 2 pasangan pelanggan tersebut akan dihapus dari solusi σ , sehingga solusi saat ini menjadi σ = {0 – 1 – 2 – 10 – 3 – 9 – 11 – 0, 0 – 6 – 14 – 0 , 0 – 8 – 16 – 0, 0 – 7 – 15 – 0}.

   b. Metode perbaikan

      Selanjutnya akan dipilih secara acak antara pasangan pelanggan 5 – 13 dan pelanggan 4 – 12 untuk disisipkan kembali pada σ = {0 – 1 – 2 – 10

– 3 – 9 – 11 – 0, 0 – 6 – 14 – 0 , 0 – 8 – 16 – 0, 0 – 7 – 15 – 0}. Misalkan yang terpilih pertama adalah pasangan pelanggan 5 dan 13, maka kemungkinan yang terbentuk dengan penyisiapan pelanggan 5 dan 13 adalah sebagai berikut.

$\sigma_1 = \{0 – 1 – 2 – 10 – 3 – 9 – 11 – 5 – 13 – 0, 0 – 6 – 14 – 0, 0 – 8 – 16 – 0, 0 – 7 – 15 – 0\}$.

$\sigma_2 = \{0 – 1 – 2 – 10 – 3 – 9 – 11 – 0, 0 – 5 – 13 – 6 – 14 – 0, 0 – 8 – 16 – 0, 0 – 7 – 15 – 0\}$.

$\sigma_3 = \{0 – 1 – 2 – 10 – 3 – 9 – 11 – 0, 0 – 6 – 14 – 5 – 13 – 0, 0 – 8 – 16 – 0, 0 – 7 – 15 – 0\}$.

$\sigma_4 = \{0 – 1 – 2 – 10 – 3 – 9 – 11 – 0, 0 – 6 – 14 – 0, 0 – 5 – 13 – 8 – 16 – 0, 0 – 7 – 15 – 0\}$.

$\sigma_5 = \{0 – 1 – 2 – 10 – 3 – 9 – 11 – 0, 0 – 6 – 14 – 0, 0 – 8 – 16 – 5 – 13 – 0, 0 – 7 – 15 – 0\}$.

Selanjutnya dilakukan penyisipan pelanggan 4 dan 12 pada kelima kemungkinan di atas. Solusi sublingkungan yang terbentuk adalah sebagai berikut.

$\sigma_1' = \{0 – 1 – 2 – 10 – 3 – 9 – 11 – 5 – 13 – 0, 0 – 4 – 12 – 6 – 14 – 0, 0 – 8 – 16 – 0, 0 – 7 – 15 – 0\}$.

$\sigma_2' = \{0 – 1 – 2 – 10 – 3 – 9 – 11 – 5 – 13 – 0, 0 – 6 – 14 – 0, 0 – 4 – 12 – 8 – 16 – 0, 0 – 7 – 15 – 0\}$.

$\sigma_3' = \{0 – 1 – 2 – 10 – 3 – 9 – 11 – 5 – 13 – 0, 0 – 6 – 14 – 0, 0 – 8 – 16 – 0, 0 – 4 – 12 – 7 – 15 – 0\}$.

$\sigma'_4 = \{0 - 1 - 2 - 10 - 3 - 9 - 11 - 0, 0 - 5 - 13 - 6 - 14 - 0, 0 - 4 - 12 - 8 - 16 - 0, 0 - 7 - 15 - 0\}$.

$\sigma'_5 = \{0 - 1 - 2 - 10 - 3 - 9 - 11 - 0, 0 - 5 - 13 - 6 - 14 - 0, 0 - 8 - 16 - 0, 0 - 4 - 12 - 7 - 15 - 0\}$.

$\sigma'_6 = \{0 - 1 - 2 - 10 - 3 - 9 - 11 - 0, 0 - 6 - 14 - 0, 0 - 5 - 13 - 8 - 16 - 0, 0 - 4 - 12 - 7 - 15 - 0\}$.

$\sigma'_7 = \{0 - 1 - 2 - 10 - 3 - 9 - 11 - 0, 0 - 6 - 14 - 0, 0 - 8 - 16 - 5 - 13 - 0, 0 - 4 - 12 - 7 - 15 - 0\}$.

$\sigma'_8 = \{0 - 1 - 2 - 10 - 3 - 9 - 11 - 0, 0 - 6 - 14 - 0, 0 - 4 - 12 - 8 - 16 - 0, 0 - 7 - 15 - 5 - 13 - 0\}$.

$\sigma'_9 = \{0 - 1 - 2 - 10 - 3 - 9 - 11 - 0, 0 - 4 - 12 - 6 - 14 - 0, 0 - 8 - 16 - 0, 0 - 7 - 15 - 5 - 13 - 0\}$.

$\sigma'_{10} = \{0 - 1 - 2 - 10 - 3 - 9 - 11 - 0, 0 - 4 - 12 - 6 - 14 - 0, 0 - 5 - 13 - 8 - 16 - 0, 0 - 7 - 15 - 0\}$.

$\sigma'_{11} = \{0 - 1 - 2 - 10 - 3 - 9 - 11 - 0, 0 - 4 - 12 - 6 - 14 - 0, 0 - 8 - 16 - 5 - 13 - 0, 0 - 7 - 15 - 0\}$.

$\sigma'_{12} = \{0 - 1 - 2 - 10 - 3 - 9 - 11 - 0, 0 - 6 - 14 - 5 - 13 - 0, 0 - 4 - 12 - 8 - 16 - 0, 0 - 7 - 15 - 0\}$.

$\sigma'_{13} = \{0 - 1 - 2 - 10 - 3 - 9 - 11 - 0, 0 - 6 - 14 - 5 - 13 - 0, 0 - 8 - 16 - 0, 0 - 4 - 12 - 7 - 15 - 0\}$.

2. Kemudian dilakukan perhitungan nilai fungsi evaluasi dari solusi sub-lingkungan $f(\sigma'_i)$ dan nilai fungsi evaluasi dari solusi saat ini $f(\sigma)$

berdasarkan persamaan (3.3). Adapun perhitungan nilai fungsi evaluasi dari solusi sub-lingkungan $f(\sigma_i')$ adalah sebagai berikut.

a. Untuk σ_1', nilai $f(\sigma_1')$ adalah

$$|\sigma_1'| = 4$$

$$\sum_{r\in\sigma_a'} t(r) = 127{,}7 + 40{,}3 + 39{,}8 + 65{,}6 = 273{,}4.$$

Dengan demikian $f(\sigma_1') = \langle 4, 273{,}4\rangle$.

b. Untuk σ_2', nilai $f(\sigma_2')$ adalah

$$|\sigma_2'| = 4$$

$$\sum_{r\in\sigma_a'} t(r) = 127{,}7 + 25{,}5 + 72{,}3 + 65{,}6 = 291{,}1.$$

Dengan demikian $f(\sigma_2') = \langle 4, 291{,}1\rangle$.

c. Untuk σ_3', nilai $f(\sigma_3')$ adalah

$$|\sigma_3'| = 4$$

$$\sum_{r\in\sigma_a'} t(r) = 127{,}7 + 25{,}5 + 39{,}8 + 89{,}3 = 282{,}3.$$

Dengan demikian $f(\sigma_3') = \langle 4, 282{,}3\rangle$.

d. Untuk σ_4', nilai $f(\sigma_4')$ adalah

$$|\sigma_4'| = 4$$

$$\sum_{r\in\sigma_a'} t(r) = 79{,}4 + 75{,}2 + 72{,}3 + 65{,}6 = 292{,}5.$$

Dengan demikian $f(\sigma_4') = \langle 4, 269{,}6\rangle$.

e. Untuk σ_5', nilai $f(\sigma_5')$ adalah

$$|\sigma_5'| = 4$$

$$\sum_{r \in \sigma_a'} t(r) = 79{,}4 + 75{,}2 + 39{,}7 + 89{,}3 = 283{,}7.$$

Dengan demikian $f(\sigma_5') = \langle 4, 283{,}7 \rangle$.

f. Untuk σ_6', nilai $f(\sigma_6')$ adalah

$$|\sigma_6'| = 4$$

$$\sum_{r \in \sigma_a'} t(r) = 79{,}4 + 25{,}5 + 83{,}3 + 89{,}3 = 277{,}5.$$

Dengan demikian $f(\sigma_6') = \langle 4, 277{,}5 \rangle$.

g. Untuk σ_7', nilai $f(\sigma_7')$ adalah

$$|\sigma_7'| = 4$$

$$\sum_{r \in \sigma_a'} t(r) = 79{,}4 + 25{,}5 + 73{,}2 + 89{,}3 = 267{,}4.$$

Dengan demikian $f(\sigma_7') = \langle 4, 267{,}4 \rangle$.

h. Untuk σ_8', nilai $f(\sigma_8')$ adalah

$$|\sigma_8'| = 4$$

$$\sum_{r \in \sigma_a'} t(r) = 79{,}4 + 25{,}5 + 72{,}3 + 85 = 262{,}2.$$

Dengan demikian $f(\sigma_8') = \langle 4, 262{,}2 \rangle$.

i. Untuk σ_8', nilai $f(\sigma_9')$ adalah

$$|\sigma_9'| = 4$$

$$\sum_{r \in \sigma_a'} t(r) = 79{,}4 + 40{,}3 + 39{,}8 + 85 = 244{,}5.$$

Dengan demikian $f(\sigma_9') = \langle 4, 244{,}5 \rangle$.

j. Untuk σ'_{10}, nilai $f(\sigma'_{10})$ adalah

$$|\sigma'_{10}| = 4$$

$$\sum_{r \in \sigma'_a} t(r) = 79{,}4 + 40{,}3 + 83{,}3 + 65{,}6 = 268{,}6.$$

Dengan demikian $f(\sigma'_{10}) = \langle 4, 268{,}6 \rangle$.

k. Untuk σ'_{11}, nilai $f(\sigma'_{11})$ adalah

$$|\sigma'_{11}| = 4$$

$$\sum_{r \in \sigma'_a} t(r) = 79{,}4 + 40{,}3 + 73{,}2 + 65{,}6 = 258{,}5.$$

Dengan demikian $f(\sigma'_{11}) = \langle 4, 258{,}5 \rangle$.

l. Untuk σ'_{12}, nilai $f(\sigma'_{12})$ adalah

$$|\sigma'_{12}| = 4$$

$$\sum_{r \in \sigma'_a} t(r) = 79{,}4 + 59 + 72{,}3 + 65{,}6 = 276{,}3.$$

Dengan demikian $f(\sigma'_{12}) = \langle 4, 276{,}3 \rangle$.

m. Untuk σ'_{13}, nilai $f(\sigma'_{13})$ adalah

$$|\sigma'_{13}| = 4$$

$$\sum_{r \in \sigma'_a} t(r) = 79{,}4 + 59 + 39{,}8 + 89{,}3 = 267{,}5.$$

Dengan demikian $f(\sigma'_{13}) = \langle 4, 267{,}5 \rangle$.

Kemudian untuk perhitungan nilai fungsi evaluasi dari solusi saat ini $f(\sigma)$ adalah

$$|\sigma| = 4$$

$$\sum_{r \in \sigma} t(r) = 127{,}7 + 40{,}3 + 39{,}8 + 65{,}6 = 273{,}4.$$

Dengan demikian $f(\sigma) = \langle 4, 273{,}4 \rangle$.

3. Selanjutnya dipilih solusi sub-lingkungan σ'_i dengan nilai fungsi evaluasi minimum untuk kemudian dibandingkan hasilnya dengan $f(\sigma)$. Dari hasil perhitungan yang telah dilakukan, solusi sub-lingkungan yang memiliki nilai fungsi evaluasi minimum adalah σ'_4, yaitu $f(\sigma'_9) = \langle 4, 244{,}5 \rangle$, dan nilai dari $f(\sigma)$ adalah $\langle 4, 273{,}4 \rangle$, maka berdasarkan *lexicographic ordering* diperoleh $\langle 4, 244{,}5 \rangle < \langle 4, 273{,}4 \rangle$ atau $f(\sigma'_9) < f(\sigma)$.

Oleh karena $f(\sigma'_9) < f(\sigma)$, maka σ'_4 diterima sebagai solusi baru. Dengan demikain, solusi saat ini menjadi $\sigma = \{0 – 1 – 2 – 10 – 3 – 9 – 11 – 0, 0 – 4 – 12 – 6 – 14 – 0, 0 – 8 – 16 – 0, 0 – 7 – 15 – 5 – 13 – 0\}$ dengan total jarak tempuh kendaraan 269,6 km.

Iterasi kedua

Diketahui solusi saat ini adalah $\sigma = \{0 – 1 – 2 – 10 – 3 – 9 – 11 – 0, 0 – 4 – 12 – 6 – 14 – 0, 0 – 8 – 16 – 0, 0 – 7 – 15 – 5 – 13 – 0\}$.

1. Akan dibentuk solusi sublingkungan σ dengan mengunakan metode penghapuasan dan perbaikan.
    a. Metode penghapusan

        Misalkan pelanggan yang terpilih adalah pasangan pelanggan 1 – 9 dan pelanggan 6 – 14, maka 2 pasangan pealnggan tersebut akan dihapus dari

solusi σ , sehingga solusi saat ini menjadi σ = {0 – 2 – 10 – 3 – 11 – 0, 0 – 4 – 12 – 0, 0 – 8 – 16 – 0, 0 – 7 – 15 – 5 – 13 – 0}.

b. Metode perbaikan

Selanjutnya akan dipilih secara acak antara pasangan pelanggan 1 – 9 dan pelanggan 6 – 14, untuk disisipkan kembali pada σ = {0 – 2 – 10 – 3 – 11 – 0, 0 – 4 – 12 – 0, 0 – 8 – 16 – 0, 0 – 7 – 15 – 5 – 13 – 0}. Misalkan yang terpilih pertama adalah pasangan pelanggan 1 dan 9, maka kemungkinan yang terbentuk dengan penyisipan pelanggan 1 dan 9 adalah sebagai berikut.

σ_1 = {0 – 1 – 2 – 10 – 3 – 9 – 11 – 0, 0 – 4 – 12 – 0, 0 – 8 – 16 – 0, 0 – 7 – 15 – 5 – 13 – 0}.

σ_2 = {0 – 2 – 10 – 3 – 11 – 0, 0 – 4 – 12 – 1 – 9 – 0, 0 – 8 – 16 – 0, 0 – 7 – 15 – 5 – 13 – 0}.

σ_3 = {0 – 2 – 10 – 3 – 11 – 0, 0 – 4 – 12 – 0, 0 – 1 – 9 – 8 – 16 – 0, 0 – 7 – 15 – 5 – 13 – 0}.

σ_4 = {0 – 2 – 10 – 3 – 11 – 0, 0 – 4 – 12 – 0, 0 – 1 – 8 – 9 – 16 – 0, 0 – 7 – 15 – 5 – 13 – 0}.

σ_5 = {0 – 2 – 10 – 3 – 11 – 0, 0 – 4 – 12 – 0, 0 – 1 – 8 – 16 – 9 – 0, 0 – 7 – 15 – 5 – 13 – 0}.

σ_6 = {0 – 2 – 10 – 3 – 11 – 0, 0 – 4 – 12 – 0, 0 – 8 – 1 – 16 – 9 – 0, 0 – 7 – 15 – 5 – 13 – 0}.

σ_7 = {0 – 2 – 10 – 3 – 11 – 0, 0 – 4 – 12 – 0, 0 – 8 – 1 – 9 – 16 – 0, 0 – 7 – 15 – 5 – 13 – 0}.

$\sigma_8 = \{0 - 2 - 10 - 3 - 11 - 0, 0 - 4 - 12 - 0, 0 - 8 - 16 - 1 - 9 - 0, 0 - 7 - 15 - 5 - 13 - 0\}$.

$\sigma_9 = \{0 - 2 - 10 - 3 - 11 - 0, 0 - 4 - 12 - 0, 0 - 8 - 16 - 0, 0 - 7 - 1 - 15 - 5 - 9 - 13 - 0\}$.

Setelah itu, dilakukan penyisipan pelanggan 6 dan 14 pada sembilan kemungkinan di atas. Solusi sub-lingkungan yang terbentuk adalah sebagai berikut.

$\sigma_1' = \{0 - 1 - 2 - 10 - 3 - 9 - 11 - 0, 0 - 4 - 12 - 6 - 14 - 0, 0 - 8 - 16 - 0, 0 - 7 - 15 - 5 - 13 - 0\}$.

$\sigma_2' = \{0 - 1 - 2 - 10 - 3 - 9 - 11 - 0, 0 - 4 - 12 - 0, 0 - 8 - 16 - 6 - 14 - 0, 0 - 7 - 15 - 5 - 13 - 0\}$.

$\sigma_3' = \{0 - 2 - 10 - 3 - 11 - 0, 0 - 4 - 12 - 1 - 9 - 0, 0 - 8 - 16 - 6 - 14 - 0, 0 - 7 - 15 - 5 - 13 - 0\}$.

$\sigma_4' = \{0 - 2 - 10 - 3 - 11 - 0, 0 - 4 - 12 - 6 - 14 - 0, 0 - 1 - 9 - 8 - 16 - 0, 0 - 7 - 15 - 5 - 13 - 0\}$.

$\sigma_5' = \{0 - 2 - 10 - 3 - 11 - 0, 0 - 4 - 12 - 6 - 14 - 0, 0 - 1 - 8 - 9 - 16 - 0, 0 - 7 - 15 - 5 - 13 - 0\}$.

$\sigma_6' = \{0 - 2 - 10 - 3 - 11 - 0, 0 - 4 - 12 - 6 - 14 - 0, 0 - 8 - 1 - 16 - 9 - 0, 0 - 7 - 15 - 5 - 13 - 0\}$.

$\sigma_7' = \{0 - 2 - 10 - 3 - 11 - 0, 0 - 4 - 12 - 6 - 14 - 0, 0 - 8 - 1 - 9 - 16 - 0, 0 - 7 - 15 - 5 - 13 - 0\}$.

$\sigma_8' = \{0 - 2 - 10 - 3 - 11 - 0, 0 - 4 - 12 - 6 - 14 - 0, 0 - 8 - 16 - 1 - 9 - 0, 0 - 7 - 15 - 5 - 13 - 0\}$.

$\sigma'_9 =$ {0 – 2 – 10 – 3 – 11 – 0, 0 – 4 – 12 – 6 – 14 – 0, 0 – 8 – 16 – 0, 0 – 7 – 1 – 15 – 5 – 9 – 13 – 0}.

2. Kemudian, dilakukan perhitungan nilai fungsi evaluasi dari solusi sub-lingkungan $f(\sigma'_i)$ dan nilai fungsi evaluasi dari solusi saat ini $f(\sigma)$ berdasarkan persamaan (3.3). Adapun perhitungan nilai fungsi evaluasi dari solusi sub-lingkungan $f(\sigma'_i)$ adalah sebagai berikut.

   a. Untuk σ'_1, nilai $f(\sigma'_1)$ adalah

   $$|\sigma'_1| = 4$$

   $$\sum_{r \in \sigma'_1} t(r) = 79{,}4 + 40{,}3 + 39{,}8 + 85 = 244{,}5.$$

   Dengan demikian $f(\sigma'_1) = \langle 4, 244{,}5 \rangle$.

   b. Untuk σ'_2, nilai $f(\sigma'_2)$ adalah

   $$|\sigma'_2| = 4$$

   $$\sum_{r \in \sigma'_2} t(r) = 79{,}4 + 38{,}5 + 52{,}5 + 85 = 255{,}4.$$

   Dengan demikian $f(\sigma'_2) = \langle 4, 255{,}4 \rangle$.

   c. Untuk σ'_3, nilai $f(\sigma'_3)$ adalah

   $$|\sigma'_3| = 4$$

   $$\sum_{r \in \sigma'_3} t(r) = 43{,}5 + 62{,}9 + 52{,}5 + 85 = 243{,}9.$$

   Dengan demikian $f(\sigma'_3) = \langle 4, 243{,}9 \rangle$.

d. Untuk σ_4', nilai $f(\sigma_4')$ adalah

$$|\sigma_4'| = 4$$

$$\sum_{r \in \sigma_4'} t(r) = 43{,}5 + 40{,}3 + 86{,}4 + 85 = 255{,}2.$$

Dengan demikian $f(\sigma_4') = \langle 4, 255{,}2 \rangle$.

e. Untuk σ_5', nilai $f(\sigma_5')$ adalah

$$|\sigma_5'| = 4$$

$$\sum_{r \in \sigma_5'} t(r) = 43{,}5 + 40{,}3 + 90{,}8 + 85 = 259{,}6.$$

Dengan demikian $f(\sigma_5') = \langle 4, 259{,}6 \rangle$.

f. Untuk σ_6', nilai $f(\sigma_6')$ adalah

$$|\sigma_6'| = 4$$

$$\sum_{r \in \sigma_6'} t(r) = 43{,}5 + 40{,}3 + 69 + 85 = 237{,}2.$$

Dengan demikian $f(\sigma_6') = \langle 4, 237{,}2 \rangle$.

g. Untuk σ_7', nilai $f(\sigma_7')$ adalah

$$|\sigma_7'| = 4$$

$$\sum_{r \in \sigma_7'} t(r) = 43{,}5 + 40{,}3 + 85{,}9 + 85 = 245{,}3.$$

Dengan demikian $f(\sigma_7') = \langle 4, 245{,}3 \rangle$.

h. Untuk σ'_8, nilai $f(\sigma'_8)$ adalah

$$|\sigma'_6| = 4$$

$$\sum_{r \in \sigma'_8} t(r) = 43{,}5 + 40{,}3 + 76{,}5 + 85 = 245{,}3.$$

Dengan demikian $f(\sigma'_8) = \langle 4, 245{,}3 \rangle$.

i. Untuk σ'_9, nilai $f(\sigma'_9)$ adalah

$$|\sigma'_9| = 4$$

$$\sum_{r \in \sigma'_9} t(r) = 43{,}5 + 40{,}3 + 39{,}8 + 93{,}3 = 216{,}9.$$

Dengan demikian $f(\sigma'_6) = \langle 4, 216{,}9 \rangle$.

Kemudian untuk perhitungan nilai fungsi evaluasi dari solusi saat ini $f(\sigma)$ adalah

$$|\sigma| = 4$$

$$\sum_{r \in \sigma} t(r) = 79{,}4 + 40{,}3 + 39{,}8 + 85 = 244{,}5.$$

Dengan demikian $f(\sigma) = \langle 4, 244{,}5 \rangle$.

3. Selanjutnya dipilih solusi sublingkungan σ'_i dengan nilai fungsi evaluasi minimum untuk kemudian dibandingkan hasilnya dengan $f(\sigma)$. Dari hasil perhitungan yang telah dilakukan, solusi sublingkungan yang memiliki nilai evaluasi minimum adalah σ'_9 yaitu $f(\sigma'_9) = \langle 4\,, 216{,}9 \rangle$, dan nilai dari $f(\sigma)$ adalah $\langle 4, 244{,}5 \rangle$, maka berdasarkan *lexicographic ordering* diperoleh $\langle 4, 216{,}9 \rangle < \langle 4, 244{,}5 \rangle$ atau $f(\sigma'_9) < f(\sigma)$.

Oleh karena $f(\sigma'_9) < f(\sigma)$, maka σ'_9 diterima sebagai solusi baru. Dengan demikain, solusi saat ini menjadi $\sigma = \{0 - 2 - 10 - 3 - 11 - 0, 0 - 4 - 12 - 0, 0 - 8 - 16 - 0, 0 - 7 - 1 - 15 - 5 - 9 - 13 - 0\}$ dengan total jarak tempuh kendaraan 216,9 km.

Dari perhitungan yang telah dilakukan menggunakan algoritma *large neighborhood search*, permasalahan PDPTW di Jogja Kurir Express menghasilkan sebuah solusi yang terdiri dari 4 rute yang melayani 16 pelanggan. Adapun rekapitulasi hasil penyelesaian masalah menggunakan algoritma *large neighborhood search* adalah sebagai berikut.

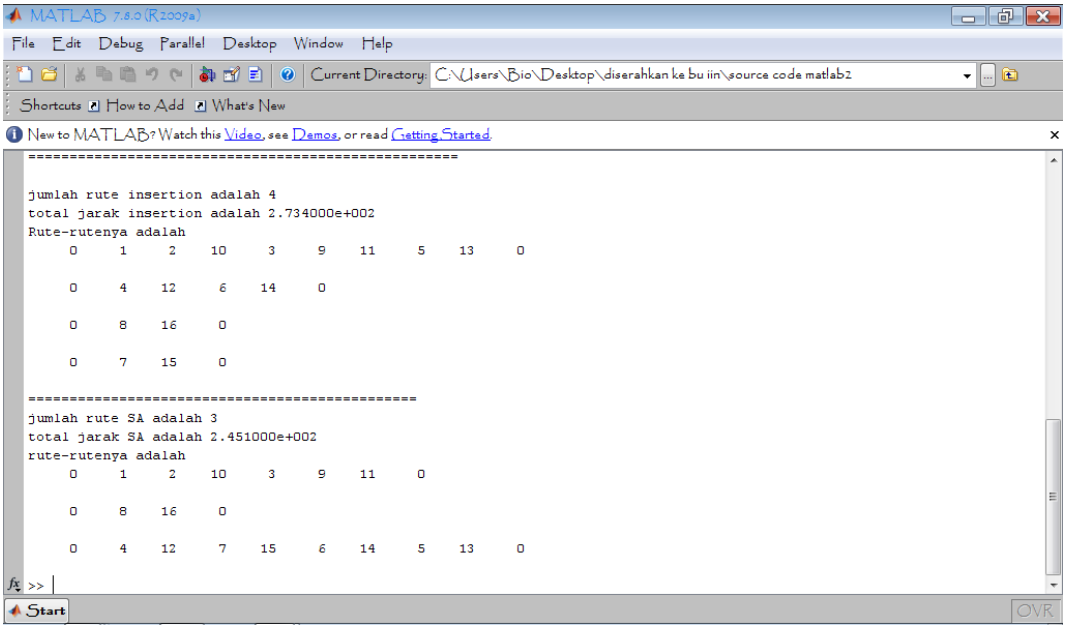
Tabel 3.9 Rekapitulasi Hasil Penyelesaian Masalah dengan Algoritma *Large Neighborhood Search*

Iterasi ke-	**Rute ke-**	**Perjalanan**	**Jarak (km)**	**Waktu tempuh (menit)**
I	1	0 – 1 – 2 – 10 – 3 – 9– 11– 0	79,4	208
	2	0 – 4 – 12 – 6 – 14 – 0	40,3	100
	3	0 – 8 – 16 – 0	39,8	96
	4	0 – 7 – 15 – 5 – 13 – 0	85	215
Total			**244,5**	**619**
II	1	0 – 2 – 10 – 3 – 11 – 0	43,5	144
	2	0 – 4 – 12 – 6 – 14 – 0	40,3	100
	3	0 – 8 – 16 – 0	39,8	96
	4	0 – 7 – 1 – 15 – 5 – 9– 13 – 0	93,3	229
Total			**216,9**	**569**

Pada Tabel 3.9, pembentukan rute menggunakan algoritma *large neighorhood search* menghasilkan 4 rute yang melayani 16 pelanggan dengan 2 kali iterasi. Pada iterasi pertama, kendaraan membentuk 4 rute dan menempuh perjalanan sejauh 244,5 km dengan waktu penyelesaian selama 619 menit. Pada

iterasi yang kedua, terbentuk 4 rute dan kendaraan menempuh perjalanan sejauh 216,9 km dengan waktu penyelesaian 569 menit.

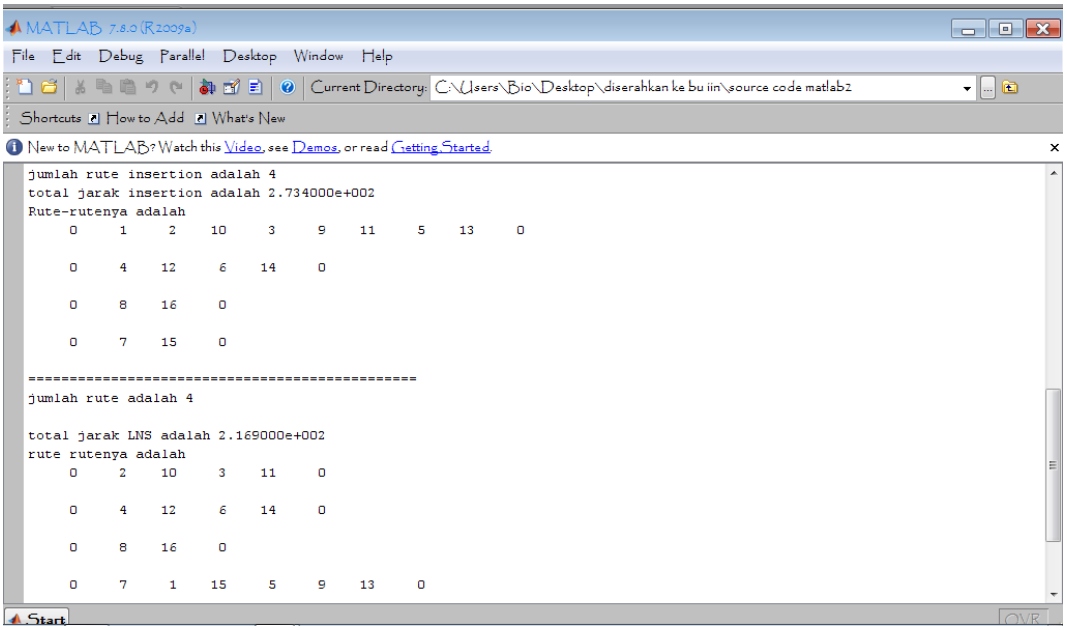
Hasil yang sama juga ditunjukkan dalam penggunaan Matlab untuk menyelesaian masalah PDPTW di Jogja Kurir Express. Berikut disajikan output Matlab untuk penggunaan algoritma *simulated annealing* dengan *insertion heuristic* sebagai solusi awal. *Source code* untuk kedua algoritma tersebut dapat dilihat pada lampiran 6.



Gambar 3.8 *Output* Matlab untuk Penggunaan Algoritma *Simulated Annealing* Pada Penyelesaian Masalah PDPTW

Pada Gambar 3.8, terlihat bahwa penggunaan algoritma *simulated annealing* dengan *insertion heuristic* sebagai solusi awal berhasil mengurangi jumlah rute dari 4 rute menjadi 3 rute dan total jarak dari 273,4 km menjadi 245,1 km.

Berikut disajikan pula *output* Matlab untuk penggunaan algoritma *large neigborhood search* dengan *insertion heuristic* sebagai solusi awal.



Gambar 3.9 *Output* Matlab untuk Penggunaan Algoritma *Large Neighborhood Search* Pada Penyelesaian Masalah PDPTW

Pada Gambar 3.9, terlihat bahwa penggunaan algoritma *large neighborhood search* dengan *insertion heuristic* sebagai solusi awal berhasil mengurangi total jarak tempuh kendaraan dari 273,4 km menjadi 216,9 km dengan jumlah rute yang sama yaitu 4 rute.

Berdasarkan perhitungan yang telah dilakukan menggunakan algoritma *simulated annealing* dan *large neighborhood search* dengan algoritma *insertion heuristic* sebagai solusi awal, baik melalui perhitungan manual maupun dengan bantuan program Matlab, didapatkan beberapa alternatif rute baru untuk melayani pelanggan Jogja Kurir Ekspress. Dari Tabel 3.7, Tabel 3.8, dan Tabel 3.9, data mengenai jumlah rute dan total jarak tempuh kendaraan yang terbentuk dapat direkapitulasi dalam Tabel 3.10 berikut.

Tabel 3.10 Rekapitulasi Rute dan Total Jarak Tempuh Kendaraan

Algoritma	Rute ke-	Perjalanan	Jarak (km)
Insertion Heuristic	1.	0 – 1– 2 – 10 – 3 – 9 – 11 – 5 – 13 – 0	127,8
	2.	0– 4 – 12 – 6 – 14 – 0	40,3
	3.	0 – 8 – 16 – 0	39,8
	4.	0 – 7 – 15 – 0	65,6
Total			**273,4**
Simulated Annealing	1	0 – 1 – 2 – 10 – 3 – 9 – 11 – 0	79,4
	2	0 – 4 – 12 – 7 – 15 – 6 – 14 – 5 – 13 – 0	125,9
	3	0 – 8 – 16 – 0	39,8
Total			**245,1**
Large Neighborhood Search	1	0 – 2 – 10 – 3 – 11 – 0	43,5
	2	0 – 4 – 12 – 6 – 14 – 0	40,3
	3	0 – 8 – 16 – 0	39,8
	4	0 – 7 – 1 – 15 – 5 – 9– 13 – 0	93,3
Total			**216,9**

Pada Tabel 3.10, secara keseluruhan, algoritma *simulated annealing* menghasilkan jumlah rute yang paling minimal dibandingkan dengan algoritma *large neighborhood search* dan algoritma *large neighborhood search* menghasilkan total jarak tempuh kendaraan yang paling minimal dibandingkan dengan algoritma *simulated annealing.* Algoritma *simulated annealing* dapat melayani 16 pelanggan yang terbagi menjadi 3 rute, lebih efektif satu rute dari algoritma *large neighborhood search.* Algoritma *large neighborhood search* menghasilkan total jarak tempuh 216,9 km, lebih efektif 28,2 km dari algoritma *simulated annealing*.

Dengan demikian, dapat disimpulkan bahwa berdasarkan jumlah rute yang terbentuk, algoritma *simulated annealing* lebih efektif dibandingkan agoritma *large neighborhood search* dan berdasarkan total jarak tempuh kendaraan, agoritma *large neighborhood search* lebih efektif dibandingkan algoritma *simulated annealing*.

BAB IV
KESIMPULAN DAN SARAN

A. KESIMPULAN

Berdasarkan pembahasan mengenai efektivitas algoritma *simulated annealing* dan *large neighborhood search* dalam penyelesaian masalah *pickup and delivery vehicle routing problem* (PDPTW), diperoleh hasil sebagai berikut.

1. Penggunaan algoritma *simulated annealing* dalam penyelesaian masalah PDPTW dapat dijelaskan pada langkah-langkah berikut ini.
   a. Tentukan solusi awal σ, yaitu solusi yang didapatkan dari solusi akhir algoritma *insertion heuristic.*
   b. Tentukan suhu awal H_0, suhu akhir H_t , *cooling rate* α, dan jumlah iterasi L_t.
   c. Tetapkan suhu awal H_0 sebagai pengontrol apakah solusi baru diterima atau tidak.
   d. Bentuk solusi sub-lingkungan σ' dengan melakukan *pair relocation* terhadap pelanggan i dan $@i$ yang dipilih secara acak.
   e. Lakukan pengecekkan apakah sublingkungan terbentuk.
      1) Jika sublingkungan terbentuk, maka hitung $e(\sigma'_i)$ menggunakan persamaan (3.1) dan urutkan solusinya dari nilai fungsi evaluasi yang terkecil, $\{\sigma'_1 \leq \cdots \leq \sigma'_s\}$, dengan s adalah banyaknya solusi dalam sublingkungan.

2) Jika sublingkungan tidak terbentuk, maka lanjut ke langkah h dan seterusnya secara terurut.

f. Hitung nilai $e(\sigma)$ berdasarkan persamaan (3.1) dan bandingkan hasilnya dengan $e(\sigma'_1)$. Lakukan pengecekkan apakah nilai $e(\sigma'_1) < e(\sigma)$.

1) Jika $e(\sigma'_1) < e(\sigma)$, maka solusi sublingkungan diterima sebagai solusi baru dan lanjut ke langkah h.

2) Jika $e(\sigma'_1) > e(\sigma)$, maka tentukan bilangan random *r* menggunakan rumus berikut:

$$r = [random\ (0,1)^{\beta} \times s] \tag{3.2}$$

dengan β adalah parameter yang telah ditentukan dan *s* adalah banyaknya solusi dalam sublingkungan. Kemudian, lanjut ke langkah g.

g. Hitung $\Delta\sigma$ berdasarkan persamaan (2.27), yaitu selisih antara $e(\sigma'_r)$ dan $e(\sigma)$. Kemudian lakukan pengecekkan apakah selisih nilai fungsi $\Delta\sigma \leq 0$.

1) Jika $\Delta\sigma \leq 0$, maka solusi sublingkungan bilangan random r , σ'_r diterima sebagai solusi baru.

2) Jika $\Delta\sigma > 0$, tentukan bilangan random *P* antara 0 dan 1. Kemudian, lakukan pengecekkan apakah $P \leq exp(-\Delta\sigma/H)$.

(a) Jika $P \leq exp(-\Delta\sigma/H)$, maka solusi sublingkungan bilangan random r , σ'_r diterima sebagai solusi baru.

(b) Jika $P > exp(-\Delta\sigma/H)$, maka lanjut ke langkah h.

h. Lakukan pengecekkan apakah iterasi sudah mencapai iterasi L_t.

1) Jika iterasi sudah mencapai iterasi L_t, maka cek apakah $H = H_t$.

(a) Jika $H = H_t$, maka lanjut ke langkah j.

(b) Jika $H \neq H_t$, maka lanjut ke langkah i.

2) Jika iterasi belum mencapai iterasi L_t, maka kembali ke langkah d. Ulangi hingga iterasi mencapai iterasi L_t.

i. Lakukan penurunan suhu H secara perlahan menggunakan suatu parameter *cooling rate* α , lalu kembali ke langkah d dan seterusnya secara terurut.

j. Proses dari algoritma *simulated annealing* telah selesai dan didapatkan solusi akhir σ.

Selanjutnya untuk penggunaan algoritma *large neighborhood search* dalam penyelesaian masalah PDPTW dapat dijelaskan pada langkah-langkah berikut ini.

a. Tentukan solusi awal *σ,* yaitu solusi yang didapatkan dari solusi akhir algoritma *insertion heuristic.*

b. Tentukan banyaknya pasangan pelanggan *x,* dan banyaknya iterasi *l.*

c. Bentuk solusi sublingkungan σ′ menggunakan metode penghapusan

d. Bentuk solusi sublingkungan σ′ menggunakan metode perbaikan

e. Hitung nilai $f(\sigma'_i)$ dan $f(\sigma)$ berdasarkan persamaan (3.3). Lalu pilih solusi sublingkungan σ'_i dengan nilai fungsi evaluasi minimum untuk dibandingkan hasilnya dengan $f(\sigma)$.

f. Lakukan pengecekan apakah nilai $f(\sigma'_i) < f(\sigma)$.

1) Jika $f(\sigma'_i) < f(\sigma)$, maka solusi sub-lingkungan σ'_i diterima sebagai solusi baru.
2) Jika $f(\sigma'_i) > f(\sigma)$, maka lanjut ke langkah g

g. Lakukan pengecekkan apakah iterasi sudah mencapai iterasi *l*.
1) Jika iterasi sudah mencapai iterasi *l*, maka lanjut ke langkah h.
2) Jika iterasi belum mencapai iterasi *l*, maka kembali ke langkah c dan seterusnya secara terurut. Ulangi sampai iterasi mencapai iterasi *l*

h. Proses dari algoritma *large neighborhood search* telah selesai dan didapatkan solusi akhir σ.

2. Dari hasil yang didapat pada Bab III, terlihat bahwa algoritma *simulated annealing* dan *large neighborhood search* berhasil menyelesaikan masalah *pickup and delivery vehicle routing problem with time windows* (PDPTW) pada data masalah penelitian optimasi (*benchmark instances*) dan contoh permasalahan penentuan rute di Jogja Kurir Express. Algoritma *simulated annealing* berhasil mengurangi jumlah rute dan algoritma *large neighborhood search* berhasil mengurangi total jarak tempuh kendaraan. Dengan demikian, dari hasil penelitian yang telah dilakukan, dapat disimpulkan bahwa berdasarkan jumlah rute yang terbentuk, algoritma *simulated annealing* lebih efektif dibandingkan agoritma *large neighborhood search* dan berdasarkan total jarak tempuh kendaraan, agoritma *large neighborhood search* lebih efektif dibandingkan algoritma *simulated annealing*.

B. SARAN

Saran yang dapat diberikan untuk pengembangan tugas akhir ini adalah mengimplementasikan algoritma *simulated annealing* dan *large neighborhood search* pada data *benchmark* lainnya dan pada kasus yang terjadi dalam kehidupan sehari-hari dengan jumlah pelanggan yang lebih besar. Penelitian ini juga dapat dikembangkan dengan menggunakan parameter yang berbeda pada iterasi setiap algoritma untuk melihat pengaruh parameter terhadap hasil yang diperoleh. Selain itu, penggunaan algoritma lain seperti algoritma hibrida, genetik, *tabu search*, dan sebagainya juga dapat dilakukan untuk pengembangan selanjutnya.

Saran untuk Jogja Kurir Express, rute yang dibentuk menggunakan algoritma *simulated annealing* ataupun *large neighborhood search* diharapkan dapat menjadi alternatif dalam memberikan pelayanan kepada pelanggan di kota Yogyakarta dan sekitarnya.

DAFTAR PUSTAKA

Bent, R., & Hentenryck, P. V. (2003). A Two-Hybrid Algorithm for Pickup and Delivery Vehicle Routing Problem with Times Windows. *Journal of the Operation Research Society*. Hlm.123-137.

Bertsimas, D., & Tsitsiklis, J. (1998). Simulated Annealing. *Statistical Science*, 8 (1). Hlm. 10-15.

Dumas, Y., Desrosiers, J., & Soumis, F. (1991). The Pickup and Delivery Problem with Time Windows. *Europeng Problem an Journal of Operation Research*, 54. Hlm. 7-22.

Dridi, Y. I., Kammarti, & R., Ksouri, M. (2011). Multi-Objective Optimization for The m-PDPTW. Aggregation Method With Use of Genetic Algorithm and Lower Bounds. *Int. J. Of Computers, Communication &Control VI*. Hlm. 246-257

Fajar Delli, W. (2006). Penyelesaian Masalah Pengambilan dan Pengiriman dengan Kendala Waktu Menggunakan Teknik Pembangkitan Kolom. *Skripsi*. Bogor : FMIPA IPB.

Fajarwati, I.A., & Wiwik A. (2012). Penerapan Algoritma Differential Evolution untuk Penyelesaian Permasalahan Vehicle Routing Problem with Delivery ang Pick-Up. *Jurnal Teknik POMITS* .Vol.1, No. 1. Hlm. 1-6.

Haibing Li, & A. Lim. A Metaheuristic for Pickup and Delivery Problem with Time Windows. *International Journal on Artificial Intelligence Tools*. World Scientific Publishing Company.

Hidayat. (1986). Teori Efektivitas dalam Kinerja Karyawan. Yogyakarta : Gajah Mada University Press.

Joubert J.W., & Claasen S. J. (2006). A Sequential Inserion Heuristic for The Initiaal Solution. *ORION : The Journal of ORSSA*,. 22. Hlm. 105-116.

Kallehauge B., Larsen J., & Marsen OBG. 2001. Lagrangean Duality Applied on Vehicle Routing Problem with Time Windows. *Technical Report*. IMM. Technical University of Denmark. DK-2800 Kgs. Lyngby – Denmark.

Fitri Karunia R. (2008). Pengembangan Algoritma Heuristik untuk Penyelesaian Dynamic Pickup and Delivery Problem With Time Windows (DPDPTW) untuk Penyedia Jasa City-Courier. *Skripsi*. Surabaya: FMIPA ITS.

Landete, Benavent, Mota, and Tirado. Benavent, dkk. *benchmark* PDPLT. Diakses dari http://www.mat.ucm.es/~gregoriotd/PDPLT.htm pada tanggal 1 September 2014. Jam 19.00 WIB.

Lau, H.C., & Liang, Z. (2001). Pickup and Delivery with Times Windows: Algorithms and Test Case Generation. *Proceeedings*, 13th IEEE International Conference on Tools with Artificial Intellegence (ICTAI'01). USA: Dalas.

Li & Lam. Li & Lam *benchmark* PDPTW. Diakses dari http://www.sintef.no/Projectweb/TOP/PDPTW/Li--Lim-benchmark/ pada tanggal 17 April 2014. Jam 09.00 WIB.

Mitrovic-Minic. (1998). Pickup and Delivery Problem with Time Windows: A Survey. *Technical Report 1998-2, SCS Simon Fraser University*.

Parment, M. Michael, Edgar G. Goodaire. *Discrete Mathematis with Graph Theory*. (2002). United States America: Prentice-Hall, Inc.

Pius A. Partanto, & M. Dahlan Bahri. (1994). Kamus Ilmiah Populer. Surabaya: Arkoba.

Pisinger, D., Ropke, S. (2010). Large Neighborhood Search. *Handbook of Metaheuristic Vol 146 of International Series in Operations Research and Managementh Science*. Hlm.399-419.

Risya Priwarnela (2012). Aplikasi Algoritma Hibrida Dua Tahap Pada Pickup and Delivery Vehicle Routing Problem with Times Windows. *Skripsi*. Jakarta: FMIPA UI.

Solomon, M. (1987). Algorithms for The Vehicle Routing and Scheduling Problems with Time Windows Constraints. *Operation Research, 35*. Hlm.254-265.

Tospornsampan, J., Ishii, M., Kita, I., & Kitamura, Y. (2007). Split-Pipe Design of Water Distribution. *World Academy of Science, Engineering and Technology 28.*

Toth P, & Vigo D. (2002). *An Overview of vehicle routing problems*. Di dalam Toth, P et al., editor. The Vehicle Routing Problem. Philadelphia: Siam. Hlm. 1-26.

Yeun, L. C., Ismail, W. R., Omar, K., & Zirour, M. (2008). Vehicle Routing Problem.: Model and Solution. Journal *of Measurement and Analysis.* 4 (1). Hlm. 205-218.

LAMPIRAN

Lampiran 1 *Source Code* Matlab untuk Data *Benchmark*

insertion_SA.m

```
clear;
clc;
fprintf('masukkan data yang diinginkan\n');
[file path] = uigetfile ('*.txt','pilih data banchmark');
fprintf('file %s dipilih\n',file);
fprintf('=============================================');
fprintf('\nsilahkan masukkan parameter yang anda inginkan\n');
fprintf('=============================================');
fprintf('\n');
fprintf('\nkapasitas kendaraan yang tersedia');
fprintf('\n');
kapasitas = input('\nmasukkan kapasitas kendaraan :');
fprintf('\n');
fprintf('====================');
fprintf('\nInsertion Heuristic\n');
fprintf('====================');
fprintf('\n');
miu = input ('\n Nilai Miu adalah');
alp1 = input ('\n Nilai Alpha 1 adalah');
alp2 = input ('\n Nilai Alpha 2 adalah');
alp3 = input ('\n Nilai Alpha 3 adalah');
fprintf('\n');
fprintf('====================');
fprintf('\nSimulated Annealing\n');
fprintf('====================');
fprintf('\n');
T = input ('\nmasukkan suhu awal :');
Takhir = input ('\nmasukkan suhu akhir :');
iterasi_tiap_suhu = input ('\nmasukkan iterasi tiap suhu :');
cooling_rate = input ('\nmasukkan nilai cooling rate :');
beta = input ('\nmasukkan nilai beta :');
fprintf('\n');
fprintf('==========================');
fprintf('==========================');
fprintf('\n');
tic;
m = load([path file]);
u = zeros(1,((length(m)+1)/2));
u(1)=1;
j=1;
for q=1:length(m)
    for l=q+1:length(m)
        d(q,l)=sqrt((m(q,2)-m(l,2))^2+(m(q,3)-m(l,3))^2);
   %matriks jarak
        d(l,q)=d(q,l);
    end
end
for i=2:length(m)
    if m(i,9) ~= 0
        [m(i,1) m(i,9)];
        c(j,:)=[m(i,1) m(i,9)]+1; %matriks pasangan pelanggan
        j=j+1;
```

```
    end
end
%insertion heuristic
for a=1:length(u)-1;
if u==1
    break;
end
%penentuan rute dari pelanggan pertama
l=0;
for p=find(u==0)
    if d(1,c(p-1,1))>l; %mencari jarak maksimum dari depot ke
pelanggan jemput
        l=d(1,c(p-1,1));
        ag=p;
    end
end
u(ag) = 1;
r = [1 c(ag-1,:) 1];
S(1)=0;
%mulai nyisip
for p=find(u==0)
    %ngitung jarak
    for i=1:length(r)-1
        S(i+1) = S(i) + m(r(i),7) + d(r(i),r(i+1));
        if S(i+1)<= m(r(i+1),5);
            S(i+1)= m(r(i+1),5);
        elseif S(i+1) > m(r(i+1),6);
            break;
        end
    end
    C = inf;
    for w=1:length(r)-1
        r1=ins(r,c(p-1,1),w);
        b=0; %kendala kapasitas
        i=1;
        b=cek_kendala(r1,m,kapasitas);
        %kendala time windows
        if b<= kapasitas
            output=cek_timewindows(r1,m,d);
            i = output{1};
            S1= output {2};
            %hitung biaya
            if i==length(r1)-1
                C11 = d(r1(w),r1(w+1)) + d(r1(w+1),r1(w+2))-
miu*d(r1(w),r1(w+2));
                C12 = S1(w+2)-S(w+1);
                C13 = m(r1(w+1),6)-S1(w+1);
                CS = alp1*(C11) + alp2*(C12) + alp3*(C13);
                if CS < C
                    C =CS;
                    Z1 = r1;
                    R1 = S1;
                    t = w;
                end
            end
        end
```

```
    end
    if C ~= inf
        D = inf;
        for v = t+1:length(Z1)-1
            r2=ins(Z1,c(p-1,2),v);
            b=0; %kapasitas kendala
            i=1;
            b=cek_kendala(r2,m,kapasitas);
            %kendala time windows
            if b<=kapasitas
                output=cek_timewindows(r2,m,d);
                w = output{1};
                S2 = output{2};
                %hitung biaya
                if w==length(r2)-1
                    C11 = d(r2(v),r2(v+1))+ d(r2(v+1),r2(v+2))-
miu*d(r2(v),r2(v+2));
                    C12 = S2(v+2) - R1(v+1);
                    C13 = m(r2(v+1),6) - S2(v+1);
                    CS = alp1*(C11)+ alp2*(C12) + alp3*(C13);
                    if CS < D
                        D = CS;
                        Z2 = r2;
                        R2 =S2;
                    end
                end
            end
        end
        if D~=inf
            u(p)=1;
            r=Z2;
        end
    end
end
r;
R{a}=r;
end
R;
distance_ins = 0;
for dis = 1 : length(R)
    for tance = 1 : length (R{dis})-1
        distance_ins = distance_ins + d(R{dis} (tance), R{dis}
(tance+1));
    end
end
Rins = R;
jum_rute_insertion = size(Rins,2);
tot_dis_insertion = distance_ins;
fprintf('\njumlah rute insertion adalah %d', jum_rute_insertion);
fprintf('\ntotal jarak insertion adalah %d\n', tot_dis_insertion);
disp ('Rute-rutenya adalah')
for i=1:length(Rins)
    disp(Rins{i}-1);
end
waktu_insertion = toc
fprintf('=============================================');
```

```
tic;
%% simulated annealing
R = Rins;
while T > Takhir
    for qp = 1 : iterasi_tiap_suhu
        [al ll] = size(R);
        a=ll;
        qp;
        [q n]=size(c);
        check_point=randi([1 q],1);
        cek1 = c(check_point,1); %cari pelanggan pickup
        cek2 = c(check_point,2); %cari pelanggan delivery
        R_pos=0;
        for i=1:a
            R_temp=R{i};
            length_r=length(R_temp);
            for j=1:length_r
                if cek1==R_temp(j)
                    R_pos=i;
                    col_del1=j;
                end
                if cek2==R_temp(j);
                    col_del2=j;
                    break;
                end
            end
            if R_pos>0
                break;
            end
        end
        ul=0;
        for p=1:a
            R_ling=R;
            R_ling{R_pos}=del(R_ling{R_pos},col_del1);
            R_ling{R_pos}=del(R_ling{R_pos},col_del2-1);
            if p~=R_pos
                R_temp=R_ling{p};
                length_r=length(R_temp);
                for j=1:length_r-1
                    R_tempt1=ins(R_temp,cek1,j);
                    %cek kendala
                    b = cek_kendala(R_tempt1,m,kapasitas);
                    if b<=kapasitas
                        output = cek_timewindows(R_tempt1,m,d);
                        i = output{1};
                        S1= output{2};
                        if i==length(R_tempt1)-1
                            for z=j+1:length(R_tempt1)-1
                                R_tempt2=ins(R_tempt1,cek2,z);
                                b=
cek_kendala(R_tempt2,m,kapasitas);
                                if b<=kapasitas

output=cek_timewindows(R_tempt2,m,d);
                                    w = output{1};
                                    S2 = output {2};
```

```
                                    if w==length(R_tempt2)-1
                                        f = R_tempt2;
                                        R_ling{p}=f;
                                        ul=ul+1;
                                        Q{ul}=R_ling;
                                    end
                                end
                            end
                        end
                    end
                end
            end
            for pp =1 : ul
                for ppp = length(Q{ul}): -1 : 1
                    if length (Q{pp}{ppp})==2
                        Q{pp}=del(Q{pp},ppp);
                    end
                end
            end
            ul;
            if ul==0
                R;
            end
            %mencari nilai sigma masing masing lingkungan
            if ul ~=0
                Q_place = zeros(ul,3);
                for ko = 1:ul
                    [so to]=size(Q{ko});
                    Q_place(ko,1)=to;
                    r_kuadrat=0;
                    for do = 1:to
                        [go yo] = size(Q{ko}{do});
                        r_kuadrat=r_kuadrat+(yo-2)^2;
                    end
                    Q_place(ko,2)=r_kuadrat*(-1);
                    distance=0;
                    for fo=1:to
                        for go=1:length(Q{ko}{fo})-1
                            distance=distance + d(Q{ko}{fo}(go),
Q{ko}{fo}(go+1));
                        end
                    end
                    Q_place(ko,3)=distance;
                end
                ul;
                Q_place;
                Q_place1=Q_place;
                [qq ww]= size (Q_place1);
                %mengurutkan sigma
                for er = 1 :qq
                    Q_place1;
                    [zz ww]=size(Q_place1);
                    wz=min(Q_place1(:,1));
                    az=find(Q_place1(:,1)==wz);
                    [sz tz]=size(az);
```

```
if sz >1
    WZ = zeros(sz,3);
    for iz=1:sz
        WZ(iz,1)=Q_place1(az(iz),1);
        WZ(iz,2)=Q_place1(az(iz),2);
        WZ(iz,3)=Q_place1(az(iz),3);
    end
    WZ;
    vz=min(WZ(:,2));
    bz=find(WZ(:,2)==vz);
    [hz lz]=size(bz);
    if hz >1
        VZ=zeros(hz,3);
        for jz=1:hz
            VZ(jz,1)=WZ(bz(jz),1);
            VZ(jz,2)=WZ(bz(jz),2);
            VZ(jz,3)=WZ(bz(jz),3);
        end
        VZ;
        uz=min(VZ(:,3));
        cz=find(VZ(:,3)==uz);
        [xz yz] = size(cz);
        if xz>1
            UZ=zeros(xz,3);
            for kz=1:xz
                UZ(kz,1)=VZ(cz(kz),1);
                UZ(kz,2)=VZ(cz(kz),2);
                UZ(kz,3)=VZ(cz(kz),3);
            end
            UZ=UZ(1,:);
        end
        if xz==1
            UZ=zeros(1,3);
            UZ(1,1)=VZ(cz(1),1);
            UZ(1,2)=VZ(cz(1),2);
            UZ(1,3)=VZ(cz(1),3);
            UZ;
        end
    end
    if hz ==1
        UZ=zeros(1,3);
        UZ(1,1)=WZ(bz(1),1);
        UZ(1,2)=WZ(bz(1),2);
        UZ(1,3)=WZ(bz(1),3);
        UZ;
    end
end
if sz==1
    UZ=zeros(1,3);
    UZ(1,1)=Q_place(az(1),1);
    UZ(1,2)=Q_place(az(1),2);
    UZ(1,3)=Q_place(az(1),3);
    UZ;
end
UZ;
for ez=1:qq
```

```
        if isequal(UZ,Q_place(ez,:))==1
            ssz=ez;
            Q_baru{er}=Q{ssz};
        end
    end
    zz;
    if zz ~=1
        for fg = 1:zz
            if isequal(UZ,Q_place1(fg,:))==1
                fgh=fg;
            end
        end
        Q_place1(fgh,:)=[];
    end
end
Q_baru;
Q_sementara=Q_baru{1};

%membandingakan Q_sementara dengan R
Rbaru=R;
[hh ii]=size(R);
bbb=0;
for xxx=1:ii
    [aa bb]=size (R{xxx});
    bbb=bbb+(bb-2)^2;
end
bbb=bbb*(-1);
d_R=0;
for rr=1:ii
    for rrr=1:length(R{rr})-1
        d_Rrr=d(R{rr}(rrr),R{rr}(rrr+1));
        d_R=d_R + d_Rrr;
    end
end
d_R;
eval_R=[ii bbb d_R];
for gw = 1:ul
    if isequal(Q_sementara,Q{gw})==1
        gf=gw;
        eval_Qsementara=Q_place(gf,:);
    end
end
if eval_Qsementara(1) < eval_R(1)
    Rbaru=Q_sementara;
end
if eval_Qsementara(1)==eval_R(1)
    if eval_Qsementara(2)<eval_R(2)
        Rbaru=Q_sementara;
    end
    if eval_Qsementara(2)==eval_R(2)
        if eval_Qsementara(3)<eval_R(3)
            Rbaru=Q_sementara;
        end
    end
end
Rbaru;
```

```
            %jika Q_sementara tidak lebih baik dari R
            if isequal (Rbaru,R)==1
                bil_r = ceil (rand^beta*ul);
                for wgw=1:ul
                    if isequal(Q_baru{bil_r},Q{wgw})==1
                        wgf=wgw;
                    end
                end
                eval_Qbilr=Q_place(wgf,:);
                eval_R;
                if eval_Qbilr(1)<eval_R(1)
                    Rbaru=Q_baru{bil_r};
                end
                if eval_Qbilr(1)==eval_R(1)
                    if eval_Qbilr(2)< eval_R(2)
                        Rbaru=Q_baru{bil_r};
                    end
                    if eval_Qbilr(2)>eval_R(2)
                        delta=eval_Qbilr(2)-eval_R(2);
                    end
                    if eval_Qbilr(2)==eval_R(2)
                        if eval_Qbilr(3)<=eval_R(2)
                            Rbaru=Q_baru{bil_r};
                        end
                        if eval_Qbilr(3)> eval_R(3)
                            delta=eval_Qbilr(3)-eval_R(3);
                        end
                    end
                end
                if delta > 0
                    if rand <= exp(-(delta)/T)
                        Rbaru=Q{bil_r};
                    end
                end
            end
            R=Rbaru;
        end
    end
    T=cooling_rate*T;
end
R_SA=R;
distance_sa=0;
for dis=1:length(R_SA)
    for tance = 1 : length(R_SA {dis})-1

distance_sa=distance_sa+d(R_SA{dis}(tance),R_SA{dis}(tance+1));
    end
end
fprintf('\njumlah rute SA adalah %d',size(R_SA,2));
fprintf('\ntotal jarak SA adalah %d\n',distance_sa);
disp('rute-rutenya adalah');
for i=1:length(R_SA)
    disp(R_SA{i}-1);
end
waktu_SA = toc
```

insertion_LNS.m

```
clear;
clc;
fprintf('masukkan data yang diinginkan\n');
[file path] = uigetfile ('*.txt','pilih data banchmark');
fprintf('file %s dipilih\n',file);
fprintf('================================================');
fprintf('\nsilahkan masukkan parameter yang anda inginkan\n');
fprintf('================================================');
fprintf('\n');
fprintf('\nkapasitas kendaraan yang tersedia');
fprintf('\n');
kapasitas = input('\nmasukkan kapasitas kendaraan :');
fprintf('\n');
fprintf('====================');
fprintf('\nInsertion Heuristic\n');
fprintf('====================');
fprintf('\n');
miu = input ('\n Nilai Miu adalah');
alp1 = input ('\n Nilai Alpha 1 adalah');
alp2 = input ('\n Nilai Alpha 2 adalah');
alp3 = input ('\n Nilai Alpha 3 adalah');
fprintf('\n');
fprintf('==========================');
fprintf('\nLarge Neighborhood Search\n');
fprintf('==========================');
fprintf('\n');
iterasi_LNS = input ('\nmasukkan iterasi untuk LNS :');
UK = input ('\nmasukkan banyaknya pasang pelanggan yang akan
direlokasi :');
tic;
m = load([path file]);
u = zeros(1,((length(m)+1)/2));
u(1)=1;
j=1;
for q=1:length(m)
    for l=q+1:length(m)
        d(q,l)=sqrt((m(q,2)-m(l,2))^2+(m(q,3)-m(l,3))^2);
   %matriks jarak
        d(l,q)=d(q,l);
    end
end
for i=2:length(m)
    if m(i,9) ~= 0
        [m(i,1) m(i,9)];
        c(j,:)=[m(i,1) m(i,9)]+1; %matriks pasangan pelanggan
        j=j+1;
    end
end
%insertion heuristic
for a=1:length(u)-1;
if u==1
    break;
end
%penentuan rute dari pelanggan pertama
l=0;
```

```
for p=find(u==0)
    if d(1,c(p-1,1))>l; %mencari jarak maksimum dari depot ke
pelanggan jemput
        l=d(1,c(p-1,1));
        ag=p;
    end
end
u(ag) = 1;
r = [1 c(ag-1,:) 1];
S(1)=0;
%mulai nyisip
for p=find(u==0)
    %ngitung jarak
    for i=1:length(r)-1
        S(i+1) = S(i) + m(r(i),7) + d(r(i),r(i+1));
        if S(i+1)<= m(r(i+1),5);
            S(i+1)= m(r(i+1),5);
        elseif S(i+1) > m(r(i+1),6);
            break;
        end
    end
    C = inf;
    for w=1:length(r)-1
        r1=ins(r,c(p-1,1),w);
        b=0; %kendala kapasitas
        i=1;
        b=cek_kendala(r1,m,kapasitas);
        %kendala time windows
        if b<= kapasitas
            output=cek_timewindows(r1,m,d);
            i = output{1};
            S1= output {2};
            %hitung biaya
            if i==length(r1)-1
                C11 = d(r1(w),r1(w+1)) + d(r1(w+1),r1(w+2))-
miu*d(r1(w),r1(w+2));
                C12 = S1(w+2)-S(w+1);
                C13 = m(r1(w+1),6)-S1(w+1);
                CS = alp1*(C11) + alp2*(C12) + alp3*(C13);
                if CS < C
                    C =CS;
                    Z1 = r1;
                    R1 = S1;
                    t = w;
                end
            end
        end
    end
    if C ~= inf
        D = inf;
        for v = t+1:length(Z1)-1
            r2=ins(Z1,c(p-1,2),v);
            b=0; %kapasitas kendala
            i=1;
            b=cek_kendala(r2,m,kapasitas);
            %kendala time windows
```

```
            if b<=kapasitas
                output=cek_timewindows(r2,m,d);
                w = output{1};
                S2 = output{2};
                %hitung biaya
                if w==length(r2)-1
                    C11 = d(r2(v),r2(v+1))+ d(r2(v+1),r2(v+2))-
miu*d(r2(v),r2(v+2));
                    C12 = S2(v+2) - R1(v+1);
                    C13 = m(r2(v+1),6) - S2(v+1);
                    CS = alp1*(C11)+ alp2*(C12) + alp3*(C13);
                    if CS < D
                        D = CS;
                        Z2 = r2;
                        R2 =S2;
                    end
                end
            end
        end
        if D~=inf
            u(p)=1;
            r=Z2;
        end
    end
end
r;
R{a}=r;
end
R;
distance_ins = 0;
for dis = 1 : length(R)
    for tance = 1 : length (R{dis})-1
        distance_ins = distance_ins + d(R{dis} (tance), R{dis}
(tance+1));
    end
end
Rins = R;
jum_rute_insertion = size(Rins,2);
tot_dis_insertion = distance_ins;
fprintf('\njumlah rute insertion adalah %d', jum_rute_insertion);
fprintf('\ntotal jarak insertion adalah %d\n', tot_dis_insertion);
disp ('Rute-rutenya adalah')
for i=1:length(Rins)
    disp(Rins{i}-1);
end
waktu_insertion = toc
fprintf('================================================');
tic;
hasil=zeros(1,2);
jumlah_rute=size(Rins,2);
total_jarak=distance_ins;
R=Rins;
%%LNS
for cak=1:iterasi_LNS
cak;
D=R;
```

```
sigma_R=[jumlah_rute total_jarak];
c_copy=c;
rel_custo=zeros(1,2);
D_lingkungan=[];
del_cust=zeros(1,UK);
cust_pick=del_cust;
cust_del=del_cust;
for i = 1 : UK
    [q1 q2] = size(c_copy);
    del_cust(i)=randi([1 q1],1);
    cust_pick(i)=c_copy(del_cust(i),1);
    cust_del(i)=c_copy(del_cust(i),2);
    rel_custo(i,1)=cust_pick(i);
    rel_custo(i,2)=cust_del(i);
    c_copy(del_cust(i),:)=[];
end
c_copy;
rel_custo;
[rr cc]=size(rel_custo);

%mengecek rute keberapa yang menjadi tempat penghapusan rel_cus
asa=zeros(1,UK);
for k=1:rr
    for kk=1:length(D)
        for kkk=1:length(D{kk})-1
            if D{kk}(kkk)==rel_custo(k,1)
                D{kk}=del(D{kk},kkk);
                asa (k)=kk;
            end
            if D{kk}(kkk)==rel_custo(k,2)
                D{kk}=del(D{kk},kkk);
                break;
            end
        end
    end
end
asa;
D;

%%mulai menyisip

rel_cust=rel_custo;
uk_rel_cust=size(rel_cust,1);
sisip=randi([1 uk_rel_cust],1);
pickup1=rel_cust(sisip,1);
deliver1=rel_cust(sisip,2);
oho=0;

for ho=1:length(D)
    D_ling1=D;
    D_temp=D{ho};
    length_Dtemp=length(D_temp);
    for hoho=1:length_Dtemp-1;
        D_temp1=ins(D_temp,pickup1,hoho);
        %cek kendala
        b=cek_kendala(D_temp1,m,kapasitas);
```

```
        if b<=kapasitas
            output=cek_timewindows(D_temp1,m,d);
            i=output{1};
            S1=output{2};
            if i==length(D_temp1)-1
                for hohoho=hoho+1:length(D_temp1)-1
                    D_temp2=ins(D_temp1,deliver1,hohoho);
                    b=cek_kendala(D_temp2,m,kapasitas);
                    if b<=200
                        output= cek_timewindows(D_temp2,m,d);
                        i=output{1};
                        S1=output{2};
                        if i==length(D_temp2)-1
                            fofo=D_temp2;
                            D_ling1{ho}=fofo;
                            oho=oho+1;
                            D_lingkungan{1}{oho}=D_ling1;
                        end
                    end
                end
            end
        end
    end
end
D_lingkungan{1};
rel_cust(sisip,:)=[];
rel_cust;

%%menyisipkan pasangan kedua hingga akhir
for iu=2:size(rel_custo,1)
    uk_rel_cust=size(rel_cust,1);
    sisip=randi([1 uk_rel_cust],1);
    pickup1=rel_cust(sisip,1);
    deliver1=rel_cust(sisip,2);
    oho=0;
    for sl=1:size(D_lingkungan{iu-1},2)
        D=D_lingkungan{iu-1}{sl};
        for ho=1:length(D)
            D_ling1=D;
            D_temp=D{ho};
            length_Dtemp=length(D_temp);
            for hoho=1:length_Dtemp-1;
                D_temp1=ins(D_temp,pickup1,hoho);
                %cek kendala
                b=cek_kendala(D_temp1,m,kapasitas);
                if b<=200
                    output=cek_timewindows(D_temp1,m,d);
                    i=output{1};
                    S1=output{2};
                    if i==length(D_temp1)-1
                        for hohoho=hoho+1:length(D_temp1)-1
                            D_temp2=ins(D_temp1,deliver1,hohoho);
                            b=cek_kendala(D_temp2,m,kapasitas);
                            if b<=kapasitas

output=cek_timewindows(D_temp2,m,d);
```

```
                                    i=output{1};
                                    S1=output{2};
                                    if i==length(D_temp2)-1
                                        fofo=D_temp2;
                                        D_ling1{ho}=fofo;
                                        oho=oho+1;
                                        D_lingkungan{iu}{oho}=D_ling1;
                                    end
                                end
                            end
                        end
                    end
                end
            end
        end
    end
    D_lingkungan{iu};
    D_lingkungan;
    rel_cust(sisip,:)=[];
end
D_lingkungan;
Q=D_lingkungan{size(rel_custo,1)};
[la pa]=size(Q);

%%menghapus rute yang hanya terdiri dari depot
for pp=1:pa
    for ppp=length(Q{pp}):-1:1
        if length(Q{pp}{ppp})==2
            Q{pp}=del(Q{pp},ppp);
        end
    end
end

%%menghitung nilai sigma
Q_place=zeros(1,2);
for ko=1:pa
    [so to]=size(Q{ko});
    Q_place(ko,1)=to;
    distance=0;
    for fo=1:to
        for go=1:length(Q{ko}{fo})-1
            distance=distance + d(Q{ko}{fo}(go),Q{ko}{fo}(go+1));
        end
    end
    Q_place(ko,2)=distance;
end
Q_place;

%%mencari sigma paling minimum
[zz ww]=size (Q_place);
wz=min(Q_place(:,1));
az=find(Q_place(:,1)==wz);
[sz tz]=size(az);
if sz>1
    WZ=zeros(sz,2);
    for iz=1:sz
        WZ(iz,1)=Q_place(az(iz),1);
```

```
        WZ(iz,2)=Q_place(az(iz),2);
    end
    WZ;
    vz=min(WZ(:,2));
    bz=find(WZ(:,2)==vz);
    [hz lz]=size(bz);
    if hz>1
        UZ=zeros(hz,2);
        for jz=1:hz
            UZ(jz,1)=WZ(bz(jz),1);
            UZ(jz,2)=WZ(bz(jz),2);
        end
        UZ=UZ(1,:);
    end
    if hz==1
        UZ=zeros(1,2);
        UZ(1,1)=WZ(bz(1),1);
        UZ(1,2)=WZ(bz(1),2);
        UZ;
    end
end
if sz==1
    UZ=zeros(1,2);
    UZ(1,1)=Q_place(az(1),1);
    UZ(1,2)=Q_place(az(1),2);
    UZ;
end
UZ;
for ez=1:pa
    if isequal(UZ,Q_place(ez,:))==1
        ssz=ez;
        Q_baru=Q{ssz};
    end
end
R_baru=R;
if UZ(1,1)<sigma_R(1,1);
    R=Q_baru;
    jumlah_rute=UZ(1,1);
    total_jarak=UZ(1,2);
else if UZ(1,2)<sigma_R(1,2)
        R=Q_baru;
        jumlah_rute=UZ(1,1);
        total_jarak=UZ(1,2);
    end
end
isequal(R_baru,R);
R;
Rnya{cak}=R;
hasil(cak,1)=jumlah_rute;
hasil(cak,2)=total_jarak;
end
hasil;
jumlah_rute=hasil(cak,1);
total_jarak=hasil(cak,2);
fprintf('\njumlah rute adalah %d\n',hasil(cak,1));
fprintf('\ntotal jarak LNS adalah %d\n',hasil(cak,2));
```

```
disp('rute rutenya adalah');
for i=1:length(R)
    disp(R{i}-1);
end
waktu_LNS = toc
```

cek kendala.m

```
function b = cek_kendala(R,m,kapasitas)
b=0;
i=1;
while i<=length(R) && b<=kapasitas
    b=b+m(R(i),4);
    i=i+1;
end
end
```

cek time windows.m

```
function output = cek_timewindows(r1,m,d)
S1(1)=0;
for i=1:length(r1)-1
    S1(i+1)=S1(i)+m(r1(i),7)+d(r1(i),r1(i+1));
    if S1(i+1)<=m(r1(i+1),5);
        S1(i+1)=m(r1(i+1),5);
    elseif S1(i+1)>m(r1(i+1),6)
        break;
    end
end
output{1}=i;
output{2}=S1;
end
```

ins.m

```
function A = ins(A,x,i)
if isempty(A)
    fprintf('\nmatrix kosong atau index melebihi batas\n');
else
    A=[A(1:i) x A(i+1:length(A))];
end
end
```

del.m

```
function A = del(A,i)
if isempty(A)||i>length(A)
    fprintf('\nmatrix kosong atau index melebihi batas\n');
elseif i== 1
    A =A(i+1:length(A));
elseif i==length(A)
    A = A(1:length(A)-1);
else
    A=[A(1:i-1) A(i+1:length(A))];
end
```

Lampiran 2

Hasil Penggunaan Algoritma *Simulated Annealing* dan *Large Neighborhood Search* pada 100 Tipe Data *Benchmark*

No	*Tipe Data*	***Insertion Heuristic***		***Simulated Annealing***		***Large Neighborhood Search***	
		|K|	**TD**	**|K|**	**TD**	**|K|**	**TD**
1	LR101	25	2151,583	19	1730,654	22	1766,114
2	LR102	22	2293,653	17	1688,428	17	1487,570
3	LR103	19	2131,993	13	1459,910	13	1292,676
4	LR104	16	2317,529	11	1441,559	11	1105,015
5	LR105	21	2270,653	14	1439,856	16	1459,704
6	LR106	19	2320,683	12	1314,139	14	1378,901
7	LR107	15	1804,047	11	1343,879	10	1111,313
8	LR108	17	2426,102	10	1175,255	10	1024,175
9	LR109	19	2487,986	12	1316,634	13	1300,152
10	LR110	16	2096,632	11	1259,882	14	1286,869
11	LRC104_50	8	1225,348	6	840,042	6	687,432
12	LRC106_50	11	1446,753	8	1160,759	9	1047,996
13	LRC102_60	12	1395,594	10	1336,602	10	1199,281
14	LRC102_50	12	1516,468	9	1168,741	10	1160,059
15	LRC102_20	5	536,038	4	503,924	4	499,645
16	LRC103_30	6	892,980	5	683,130	5	610,100
17	LRC104_40	7	1024,244	5	722,311	6	687,851
18	LRC105_50	12	1376,422	11	1112,755	11	1095,730
19	LRC106_60	13	1730,106	9	1218,004	9	1145,380
20	LRC103_60	11	1484,050	8	1050,074	9	1021,117
21	LRC107_60	10	1363,183	8	1032,135	9	903,716
22	LRC104_60	10	1524,857	7	999,285	8	897,726
23	LRC105_60	11	1470,051	9	1103,412	10	1062,734
24	LR106_60	15	1325,530	10	1081,970	10	944,0113
25	LR201_20	2	649,949	2	593,339	2	429,475
26	LR110_30	7	846,741	4	583,4521	5	541,706
27	LR105_50	10	1136695	8	910,285	9	863,262
28	LR105_60	14	1405,398	12	1132,729	12	1029,178
29	LR108_20	4	476,695	3	400,643	4	402,581
30	LR107_40	8	1076,181	6	741,2735	6	730,893
31	LC101_20	4	396,840	4	404,363	4	396,84
32	LC101_30	6	923,822	5	758,234	6	558,66
33	LC101_40	9	1409,927	7	820,628	7	750,405
34	LC101_50	10	1579,123	8	866,673	8	722,185
35	LC107_50	7	1599,623	6	1000,910	7	612,680
36	LC102_30	5	1167,228	5	668,042	5	576,527
37	LC102_40	6	912,121	5	924,518	6	635,021
38	LC103_20	4	750,169	4	616,663	4	388,688
39	LC103_30	4	866,651	4	687,839	4	553,757

40	LC103_40	5	1314,303	5	892,580	5	780,997
41	LC108_60	7	1769,747	6	969,097	7	885,797
42	LR101_20	6	456,966	6	456,966	6	449,459
43	LR101_30	8	718,697	8	685,812	8	672,787
44	LR101_40	10	990,652	10	914,928	10	890,325
45	LR101_50	14	1144,977	12	950,695	12	927,425
46	LR101_60	15	1325,530	13	1158,47	13	1077,88
47	LRC104_30	11	1446,753	8	1160,759	9	1047,996
48	LRC102_60	12	1395,594	10	1336,602	10	1199,281
49	LRC102_50	12	1516,468	9	1168,741	10	1160,059
50	LRC108_60	11	1563,995	7	1030,401	8	959,060
51	LRC108_40	7	1053,105	7	1052,418	6	664,439
52	LRC107_20	4	538,492	4	538,492	4	474,482
53	LRC160_30	6	876,690	6	873,642	6	701,018
54	LRC108_20	3	379,717	3	379,717	3	376,696
55	LRC108_50	10	1422,911	6	869,673	7	858,264
56	LRC103_20	5	626,747	3	490,414	4	435,850
57	LRC107_50	9	1108,777	7	890,758	8	824,253
58	LRC103_40	9	1182,848	6	794,658	7	742,029
59	LRC104_30	5	562,139	4	666,882	5	510,079
60	LRC106_20	5	722,286	4	490,859	4	490,859
61	LRC202_20	2	675,736	2	623,686	2	528,561
62	LC201_20	2	685,142	2	685,142	2	393,147
63	LR105_40	9	1040,332	8	890,183	9	819,042
64	LR102_20	6	526,278	5	473,962	5	417,832
65	LR102_30	7	638,742	7	563,899	7	528,263
66	LR104_40	7	892,892	5	563,331	6	541,673
67	LR104_50	8	901,132	5	617,679	6	88,223
68	LR102_60	13	1440,372	12	1119,589	12	999,112
69	LR103_20	5	459,497	4	322,115	4	318,820
70	LR103_30	7	913,117	6	685,421	6	679,376
71	LR103_40	7	850,516	7	858,332	7	780,580
72	LR103_50	11	1155,813	9	1031,827	9	929,791
73	LR108_60	10	1336,011	8	1132,453	8	809,031
74	LC107_60	9	1900,416	7	1393,139	8	694,336
75	LR107_60	10	1326,693	8	927,311	8	857,180
76	LR106_50	11	1148,115	7	869,118	8	792,174
77	LR104_20	4	479,285	3	320,185	3	320,185
78	LR108_30	6	663,423	4	509,319	4	472,086
79	LR107_50	9	989,016	7	862,431	7	708,931
80	LR106_40	8	806,928	7	699,430	7	664,993
81	LRC101	21	2480,552	14	1774,275	16	1741,994
82	LRC102	22	2931,222	12	1661,496	14	1666,815
83	LRC103	18	2739,268	11	1333,250	12	1392,972
84	LRC104	15	2327,823	10	1287,657	11	1285,483
85	LRC105	20	2559,004	13	1708,730	16	1803,292
86	LRC106	21	2929,406	12	1595,213	14	1571,317
87	LRC107	19	2674,068	11	1340,261	13	1308,230

88	LRC108	19	2916,401	10	1227,693	11	1273,767
89	LC121	35	8714,475	20	2777,797	21	2843,783
90	LR121	32	9234,821	20	5546,789	27	5636,077
91	LR111	17	2165,277	11	1183,208	13	1203,231
92	LR112	15	2019,707	10	1148,617	12	1143,092
93	LR201	6	2845,558	4	1903,500	6	1354,127
94	LC103	14	3480,805	10	1588,977	11	862,739
95	LC107	13	2809,078	11	1475,643	10	828,937
96	LC108	13	3004,662	11	1555,564	10	828,937
97	LC109	11	2195,430	10	1695,879	11	1032,373
98	LC201	4	1743,249	3	595,116	4	648,030
99	LC202	4	2204,918	3	871,770	3	591,557
100	LC205	4	2081,154	4	1537,009	3	588,876

Lampiran 3

Transportation Request

No	Nama Pengirim	No. Telepon Pengirim	Alamat pengirim	Nama Penerima	No. Telepon penerima	Alamat Penerima	Jenis Barang	Jumlah Barang	Layanna pengiriman
1	Mbak Lia	081326872721	Jl. Rejosari Rt 02/ RW 20 Sardonoharjo, Ngaglik, Sleman	Bpk.Fauzan	088216153657	Perum Mutiara Asri , Timur Kec. Banguntapan	Buku	2 kg	1 hari
2	Toko "Kusuka"	085643753533	Jl. Wijilan, No. 117, Yogyakarta	Larasati	089655245242	Krangan No.245,Yogyakarta	Bahan makanan	8 kg	1 hari
3	Fitriani Purnamasari	085643861726	Jl. Gatot Subroto No.10 Mandingan, Ringinharjo, Bantul	Apri	081392803223	Perum Janti Buana Asri B 11, Banguntapan, Bantul	Kain	5 kg	1 hari
4	Ummu Azka	085868849000	Mejing Kidul RT 01/RW 08, Ambarketawang, Gamping, Sleman	Retno Septina W.	081328721140	Warak Kidul, Sumberadi, Mlati, Sleman	Buku	4 kg	1 hari
5	Mama Moza	081227239300	Perum Griya Taman Asri Blok I No. 334, Donoharjo, Sleman	Ibu Henny	085722123280	Perum Pemda DIY, Banjardadap, No P48, Potorono, Bantul	Pakaian	8 kg	1 hari
6	Diah Retno	089691933315	Baturan, Jambon, Sleman	Liza Dwipantari A.	08122708237	Perum Griya Arga Permai H9, Nogotirto, Sleman	Sprei	5 kg	1 hari
7	Melanie Sanjaya	087863535353	Tegalrejo No. 252, Yogyakarta	Nurhayati	085643663736	Salam, Dukuharjo, Salam, Magelang	Bahan makanan	6 kg	1 hari
8	Iya Ratisa	085228555993	Jl. Raya Pleret Km 2,5, Perum Puri Sakinah No. C 5	Ika	081804349695	Jl.Nangka 1 No.167, Karangnangka, Maguwoharjo, Sleman	Pakaian	7 kg	1 hari

Lampiran 4

Matriks Jarak Antar Pelanggan dan Antar Pelanggan dengan Depot (km)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0	20,3	3,1	11,3	9,4	15,5	11,9	7,8	8,9	6,5	6,6	7,3	19,7	6,8	10,2	31,5	11,5
1	20,3	0	17,5	31,1	19,5	7,7	13,7	13,5	28,7	26,8	14,2	24,1	10,8	29,1	16,6	21,2	14,9
2	3,1	17,5	0	11,7	6	12,1	8,1	4,5	8,2	9,4	4,3	8,8	16,3	12,5	6,8	28,1	10,7
3	11,3	31,1	11,7	0	13,8	21,6	19,9	14	3,6	14,5	13,6	15,2	24,8	11,5	16,6	37,5	23,3
4	9,4	19,5	6	13,8	0	14,1	7,7	7,3	11,5	14,6	7,2	15,3	9,4	17,6	3,8	26,3	16,9
5	15,5	7,7	12,1	21,6	14,1	0	6,9	10,1	21,9	25	7,5	20,7	5,6	28,1	8,8	16	10
6	11,9	13,7	8,1	19,9	7,7	6,9	0	3,6	12,4	19,9	3	14,2	7,9	18	3,4	21	10,6
7	7,8	13,5	4,5	14	7,3	10,1	3,6	0	10,2	13	2,9	12,1	12,7	16,1	3,4	26,3	16,9
8	8,9	28,7	8,2	3,6	11,5	21,9	12,4	10,2	0	10,9	10,1	11,7	22,6	8,8	14,4	33,9	19,4
9	6,5	26,8	9,4	14,5	14,6	25	19,9	13	10,9	0	11,8	1,9	21,5	3	18,8	33,8	10
10	6,6	14,2	4,3	13,6	7,2	7,5	3	2,9	10,1	11,8	0	11,2	12,2	15	3,7	24,4	11,3
11	7,3	24,1	8,8	15,2	15,3	20,7	14,2	12,1	11,7	1,9	11,2	0	20,9	4,3	18,2	33,1	9,4
12	19,7	10,8	16,3	24,8	9,4	5,6	7,9	12,7	22,6	21,5	12,2	20,9	0	26,9	7,6	15,3	13,8
13	6,8	29,1	12,5	11,5	17,6	28,1	18	16,1	8,8	3	15	4,3	26,9	0	19	36,8	13
14	10,2	16,6	6,8	16,6	3,8	8,8	3,4	3,4	14,4	18,8	3,7	18,2	7,6	19	0	23	13,6
15	31,5	21,2	28,1	37,5	26,3	16	21	26,3	33,9	33,8	24,4	33,1	15,3	36,8	23	0	26,1
16	11,5	14,9	10,7	23,3	16,9	10	10,6	16,9	19,4	10	11,3	9,4	13,8	13	13,6	26,1	0

Keterangan :

0 Jogja Kurir Ekspress
1 Mbak Lia
2 Toko “Kusuka”
3 Fitriani P
4 Ummu Azka
5 Moza
6 Diah Retno
7 Melanie R.
8 Iya Ratisa
9 Bapak Fauzan
10 Larasati
11 Apri
12 Retno S. W.
13 Liza Dwipantari A.
14 Nurhayati
15 Ika
16 Ibu Henny

Lampiran 5

Matriks Waktu Tempuh Kendaraan (menit)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0	30	5	17	14	23	18	12	13	10	10	11	30	10	15	47	17
1	30	0	26	47	29	12	21	20	43	40	21	36	16	44	25	32	22
2	5	26	0	18	9	18	12	7	12	14	6	13	24	19	10	42	16
3	17	47	18	0	21	32	30	21	5	22	20	23	37	17	25	56	35
4	14	29	9	21	0	21	12	11	17	22	11	23	14	26	6	39	25
5	23	12	18	32	21	0	10	15	33	38	11	31	8	42	13	24	15
6	18	21	12	30	12	10	0	5	19	30	4	21	12	27	5	31	16
7	12	20	7	21	11	15	5	0	15	19	4	18	19	24	5	39	25
8	13	43	12	5	17	33	19	15	0	16	15	18	34	13	22	51	29
9	10	40	14	22	22	38	30	19	16	0	19	3	32	5	28	51	15
10	10	21	6	20	11	11	4	4	15	19	0	17	18	23	6	37	17
11	11	36	13	23	23	31	21	18	18	3	17	0	31	7	27	50	14
12	30	16	24	37	14	8	12	19	34	32	18	31	0	40	11	23	21
13	10	44	19	17	26	42	27	24	13	5	23	7	40	0	29	55	20
14	15	25	10	25	6	13	5	5	22	28	6	27	11	29	0	35	20
15	47	32	42	56	39	24	31	39	51	51	37	50	23	55	35	0	39
16	17	22	16	35	25	15	16	25	29	15	17	14	21	20	20	39	0

Keterangan :

0 Jogja Kurir Ekspress
1 Mbak Lia
2 Toko “Kusuka”
3 Fitriani P
4 Ummu Azka
5 Moza
6 Diah Retno
7 Melanie R.
8 Iya Ratisa
9 Bapak Fauzan
10 Larasati
11 Apri
12 Retno S. W.
13 Liza Dwipantari A.
14 Nurhayati
15 Ika
16 Ibu Henny

Lampiran 6 *Source Code* Matlab untuk Contoh Masalah PDPTW

insertion_SA.m

```
clear;
clc;
fprintf('masukkan data yang diinginkan\n');
[file path] = uigetfile ('*.txt','pilih data banchmark');
[jarak path] = uigetfile ('*.txt','masukkan matriks jarak');
fprintf('file %s dipilih\n',file);
fprintf('file %s dipilih\n',jarak);
fprintf('================================================');
fprintf('\nsilahkan masukkan parameter yang anda inginkan\n');
fprintf('================================================');
fprintf('\n');
fprintf('\nkapasitas kendaraan yang tersedia');
fprintf('\n');
kapasitas = input('\nmasukkan kapasitas kendaraan :');
fprintf('\n');
fprintf('====================');
fprintf('\nInsertion Heuristic\n');
fprintf('====================');
fprintf('\n');
miu = input('\n Nilai Miu adalah');
alp1 = input('\n Nilai Alpha 1 adalah');
alp2 = input('\n Nilai Alpha 2 adalah');
alp3 = input('\n Nilai Alpha 3 adalah');
fprintf('\n');
fprintf('====================');
fprintf('\nSimulated Annealing\n');
fprintf('====================');
fprintf('\n');
T = input('\nmasukkan suhu awal :');
Takhir = input('\nmasukkan suhu akhir :');
iterasi_tiap_suhu = input('\nmasukkan iterasi tiap suhu :');
cooling_rate = input('\nmasukkan nilai cooling rate :');
beta = input('\nmasukkan nilai beta :');
fprintf('\n');
fprintf('==========================');
fprintf('==========================');
fprintf('\n');
tic;
m = load([path file]);
d =load([path jarak]);
u = zeros(1,((length(m)+1)/2));
u(1)=1;
j=1;
for i=2:length(m)
    if m(i,7) ~= 0
        [m(i,1) m(i,7)];
        c(j,:)=[m(i,1) m(i,7)]+1; %matriks pasangan pelanggan
        j=j+1;
    end
end
```

```
%insertion heuristic
for a=1:length(u)-1;
if u==1
    break;
end
%penentuan rute dari pelanggan pertama
l=0;
for p=find(u==0)
    if d(1,c(p-1,1))>l; %mencari jarak maksimum dari depot ke
pelanggan jemput
        l=d(1,c(p-1,1));
        ag=p;
    end
end
u(ag) = 1;
r = [1 c(ag-1,:) 1];
S(1)=0;
%mulai nyisip
for p=find(u==0)
    %ngitung jarak
    for i=1:length(r)-1
        S(i+1) = S(i) + m(r(i),5) + d(r(i),r(i+1));
        if S(i+1)<= m(r(i+1),3);
            S(i+1)= m(r(i+1),3);
        elseif S(i+1) > m(r(i+1),4);
            break;
        end
    end
    C = inf;
    for w=1:length(r)-1
        r1=ins(r,c(p-1,1),w);
        b=0; %kendala kapasitas
        i=1;
        b=cek_kendala(r1,m,kapasitas);
        %kendala time windows
        if b<= kapasitas
            output=cek_timewindows(r1,m,d);
            i = output{1};
            S1= output {2};
            %hitung biaya
            if i==length(r1)-1
                C11 = d(r1(w),r1(w+1)) + d(r1(w+1),r1(w+2))-
miu*d(r1(w),r1(w+2));
                C12 = S1(w+2)-S(w+1);
                C13 = m(r1(w+1),4)-S1(w+1);
                CS = alp1*(C11) + alp2*(C12) + alp3*(C13);
                if CS < C
                    C =CS;
                    Z1 = r1;
                    R1 = S1;
                    t = w;
                end
            end
        end
    end
    if C ~= inf
```

```
        D = inf;
        for v = t+1:length(Z1)-1
            r2=ins(Z1,c(p-1,2),v);
            b=0; %kapasitas kendala
            i=1;
            b=cek_kendala(r2,m,kapasitas);
            %kendala time windows
            if b<=kapasitas
                output=cek_timewindows(r2,m,d);
                w = output{1};
                S2 = output{2};
                %hitung biaya
                if w==length(r2)-1
                    C11 = d(r2(v),r2(v+1))+ d(r2(v+1),r2(v+2))-
miu*d(r2(v),r2(v+2));
                    C12 = S2(v+2) - R1(v+1);
                    C13 = m(r2(v+1),4) - S2(v+1);
                    CS = alp1*(C11)+ alp2*(C12) + alp3*(C13);
                    if CS < D
                        D = CS;
                        Z2 = r2;
                        R2 =S2;
                    end
                end
            end
        end
        if D~=inf
            u(p)=1;
            r=Z2;
        end
    end
end
r;
R{a}=r;
end
R;
distance_ins = 0;
for dis = 1 : length(R)
    for tance = 1 : length (R{dis})-1
        distance_ins = distance_ins + d(R{dis} (tance), R{dis}
(tance+1));
    end
end
Rins = R;
jum_rute_insertion = size(Rins,2);
tot_dis_insertion = distance_ins;
fprintf('\njumlah rute insertion adalah %d', jum_rute_insertion);
fprintf('\ntotal jarak insertion adalah %d\n', tot_dis_insertion);
disp ('Rute-rutenya adalah')
for i=1:length(Rins)
    disp(Rins{i}-1);
end
waktu_insertion = toc;
fprintf('================================================');
tic;
%% simulated annealing
```

```
R = Rins;
while T > Takhir
    for qp = 1 : iterasi_tiap_suhu
        [al ll] = size(R);
        a=ll;
        qp;
        [q n]=size(c);
        check_point=randi([1 q],1);
        cek1 = c(check_point,1); %cari pelanggan pickup
        cek2 = c(check_point,2); %cari pelanggan delivery
        R_pos=0;
        for i=1:a
            R_temp=R{i};
            length_r=length(R_temp);
            for j=1:length_r
                if cek1==R_temp(j)
                    R_pos=i;
                    col_del1=j;
                end
                if cek2==R_temp(j);
                    col_del2=j;
                    break;
                end
            end
            if R_pos>0
                break;
            end
        end
        ul=0;
        for p=1:a
            R_ling=R;
            R_ling{R_pos}=del(R_ling{R_pos},col_del1);
            R_ling{R_pos}=del(R_ling{R_pos},col_del2-1);
            if p~=R_pos
                R_temp=R_ling{p};
                length_r=length(R_temp);
                for j=1:length_r-1
                    R_tempt1=ins(R_temp,cek1,j);
                    %cek kendala
                    b = cek_kendala(R_tempt1,m,kapasitas);
                    if b<=kapasitas
                        output = cek_timewindows(R_tempt1,m,d);
                        i = output{1};
                        S1= output{2};
                        if i==length(R_tempt1)-1
                            for z=j+1:length(R_tempt1)-1
                                R_tempt2=ins(R_tempt1,cek2,z);
                                b=
cek_kendala(R_tempt2,m,kapasitas);
                                if b<=kapasitas

output=cek_timewindows(R_tempt2,m,d);
                                    w = output{1};
                                    S2 = output {2};
                                    if w==length(R_tempt2)-1
                                        f = R_tempt2;
```

```
                                        R_ling{p}=f;
                                        ul=ul+1;
                                        Q{ul}=R_ling;
                                    end
                                end
                            end
                        end
                    end
                end
            end
        end
        for pp =1 : ul
            for ppp = length(Q{ul}): -1 : 1
                if length (Q{pp}{ppp})==2
                    Q{pp}=del(Q{pp},ppp);
                end
            end
        end
        ul;
        if ul==0
            R;
        end
        %mencari nilai sigma masing masing lingkungan
        if ul ~=0
            Q_place = zeros(ul,3);
            for ko = 1:ul
                [so to]=size(Q{ko});
                Q_place(ko,1)=to;
                r_kuadrat=0;
                for do = 1:to
                    [go yo] = size(Q{ko}{do});
                    r_kuadrat=r_kuadrat+(yo-2)^2;
                end
                Q_place(ko,2)=r_kuadrat*(-1);
                distance=0;
                for fo=1:to
                    for go=1:length(Q{ko}{fo})-1
                        distance=distance + d(Q{ko}{fo}(go),
Q{ko}{fo}(go+1));
                    end
                end
                Q_place(ko,3)=distance;
            end
            ul;
            Q_place;
            Q_place1=Q_place;
            [qq ww]= size (Q_place1);
            %mengurutkan sigma
            for er = 1 :qq
                Q_place1;
                [zz ww]=size(Q_place1);
                wz=min(Q_place1(:,1));
                az=find(Q_place1(:,1)==wz);
                [sz tz]=size(az);
                if sz >1
                    WZ = zeros(sz,3);
```

```
    for iz=1:sz
        WZ(iz,1)=Q_place1(az(iz),1);
        WZ(iz,2)=Q_place1(az(iz),2);
        WZ(iz,3)=Q_place1(az(iz),3);
    end
    WZ;
    vz=min(WZ(:,2));
    bz=find(WZ(:,2)==vz);
    [hz lz]=size(bz);
    if hz >1
        VZ=zeros(hz,3);
        for jz=1:hz
            VZ(jz,1)=WZ(bz(jz),1);
            VZ(jz,2)=WZ(bz(jz),2);
            VZ(jz,3)=WZ(bz(jz),3);
        end
        VZ;
        uz=min(VZ(:,3));
        cz=find(VZ(:,3)==uz);
        [xz yz] = size(cz);
        if xz>1
            UZ=zeros(xz,3);
            for kz=1:xz
                UZ(kz,1)=VZ(cz(kz),1);
                UZ(kz,2)=VZ(cz(kz),2);
                UZ(kz,3)=VZ(cz(kz),3);
            end
            UZ=UZ(1,:);
        end
        if xz==1
            UZ=zeros(1,3);
            UZ(1,1)=VZ(cz(1),1);
            UZ(1,2)=VZ(cz(1),2);
            UZ(1,3)=VZ(cz(1),3);
            UZ;
        end
    end
    if hz ==1
        UZ=zeros(1,3);
        UZ(1,1)=WZ(bz(1),1);
        UZ(1,2)=WZ(bz(1),2);
        UZ(1,3)=WZ(bz(1),3);
        UZ;
    end
end
if sz==1
    UZ=zeros(1,3);
    UZ(1,1)=Q_place(az(1),1);
    UZ(1,2)=Q_place(az(1),2);
    UZ(1,3)=Q_place(az(1),3);
    UZ;
end
UZ;
for ez=1:qq
    if isequal(UZ,Q_place(ez,:))==1
        ssz=ez;
```

```
            Q_baru{er}=Q{ssz};
        end
    end
    zz;
    if zz ~=1
        for fg = 1:zz
            if isequal(UZ,Q_place1(fg,:))==1
                fgh=fg;
            end
        end
        Q_place1(fgh,:)=[];
    end
end
Q_baru;
Q_sementara=Q_baru{1};

%membandingakan Q_sementara dengan R
Rbaru=R;
[hh ii]=size(R);
bbb=0;
for xxx=1:ii
    [aa bb]=size (R{xxx});
    bbb=bbb+(bb-2)^2;
end
bbb=bbb*(-1);
d_R=0;
for rr=1:ii
    for rrr=1:length(R{rr})-1
        d_Rrr=d(R{rr}(rrr),R{rr}(rrr+1));
        d_R=d_R + d_Rrr;
    end
end
d_R;
eval_R=[ii bbb d_R];
for gw = 1:ul
    if isequal(Q_sementara,Q{gw})==1
        gf=gw;
        eval_Qsementara=Q_place(gf,:);
    end
end
if eval_Qsementara(1) < eval_R(1)
    Rbaru=Q_sementara;
end
if eval_Qsementara(1)==eval_R(1)
    if eval_Qsementara(2)<eval_R(2)
        Rbaru=Q_sementara;
    end
    if eval_Qsementara(2)==eval_R(2)
        if eval_Qsementara(3)<eval_R(3)
            Rbaru=Q_sementara;
        end
    end
end
Rbaru;

%jika Q_sementara tidak lebih baik dari R
```

```
            if isequal (Rbaru,R)==1
                bil_r = ceil (rand^beta*ul);
                for wgw=1:ul
                    if isequal(Q_baru{bil_r},Q{wgw})==1
                        wgf=wgw;
                    end
                end
                eval_Qbilr=Q_place(wgf,:);
                eval_R;
                if eval_Qbilr(1)<eval_R(1)
                    Rbaru=Q_baru{bil_r};
                end
                if eval_Qbilr(1)==eval_R(1)
                    if eval_Qbilr(2)< eval_R(2)
                        Rbaru=Q_baru{bil_r};
                    end
                    if eval_Qbilr(2)>eval_R(2)
                        delta=eval_Qbilr(2)-eval_R(2);
                    end
                    if eval_Qbilr(2)==eval_R(2)
                        if eval_Qbilr(3)<=eval_R(2)
                            Rbaru=Q_baru{bil_r};
                        end
                        if eval_Qbilr(3)> eval_R(3)
                            delta=eval_Qbilr(3)-eval_R(3);
                        end
                    end
                end
                if delta > 0
                    if rand <= exp(-(delta)/T)
                        Rbaru=Q{bil_r};
                    end
                end
            end
            R=Rbaru;
        end
    end
    T=cooling_rate*T;
end
R_SA=R;
distance_sa=0;
for dis=1:length(R_SA)
    for tance = 1 : length(R_SA {dis})-1

distance_sa=distance_sa+d(R_SA{dis}(tance),R_SA{dis}(tance+1));
    end
end
fprintf('\njumlah rute SA adalah %d',size(R_SA,2));
fprintf('\ntotal jarak SA adalah %d\n',distance_sa);
disp('rute-rutenya adalah');
for i=1:length(R_SA)
    disp(R_SA{i}-1);
end
waktu_SA = toc;
```

insertion_LNS.m

```
clear;
clc;
fprintf('masukkan data yang diinginkan\n');
[file path] = uigetfile ('*.txt','pilih data banchmark');
[jarak path] = uigetfile ('*.txt','masukkan matriks jarak');
fprintf('file %s dipilih\n',file);
fprintf('file %s dipilih\n',jarak);
fprintf('================================================');
fprintf('\nsilahkan masukkan parameter yang anda inginkan\n');
fprintf('================================================');
fprintf('\n');
fprintf('\nkapasitas kendaraan yang tersedia');
fprintf('\n');
kapasitas = input('\nmasukkan kapasitas kendaraan :');
fprintf('\n');
fprintf('====================');
fprintf('\nInsertion Heuristic\n');
fprintf('====================');
fprintf('\n');
miu = input ('\n Nilai Miu adalah');
alp1 = input ('\n Nilai Alpha 1 adalah');
alp2 = input ('\n Nilai Alpha 2 adalah');
alp3 = input ('\n Nilai Alpha 3 adalah');
fprintf('\n');
fprintf('==========================');
fprintf('\nLarge Neighborhood Search\n');
fprintf('==========================');
fprintf('\n');
iterasi_LNS = input ('\nmasukkan iterasi untuk LNS :');
UK = input ('\nmasukkan banyaknya pasang pelanggan yang akan
direlokasi :');
tic;
m = load([path file]);
d=load([path jarak]);
u = zeros(1,((length(m)+1)/2));
u(1)=1;
j=1;
for i=2:length(m)
    if m(i,7) ~= 0
        [m(i,1) m(i,7)];
        c(j,:)=[m(i,1) m(i,7)]+1; %matriks pasangan pelanggan
        j=j+1;
    end
end
%insertion heuristic
for a=1:length(u)-1;
if u==1
    break;
end
%penentuan rute dari pelanggan pertama
l=0;
for p=find(u==0)
    if d(1,c(p-1,1))>l; %mencari jarak maksimum dari depot ke
pelanggan jemput
```

```
        l=d(1,c(p-1,1));
        ag=p;
    end
end
u(ag) = 1;
r = [1 c(ag-1,:) 1];
S(1)=0;
%mulai nyisip
for p=find(u==0)
    %ngitung jarak
    for i=1:length(r)-1
        S(i+1) = S(i) + m(r(i),5) + d(r(i),r(i+1));
        if S(i+1)<= m(r(i+1),3);
            S(i+1)= m(r(i+1),3);
        elseif S(i+1) > m(r(i+1),4);
            break;
        end
    end
    C = inf;
    for w=1:length(r)-1
        r1=ins(r,c(p-1,1),w);
        b=0; %kendala kapasitas
        i=1;
        b=cek_kendala(r1,m,kapasitas);
        %kendala time windows
        if b<= kapasitas
            output=cek_timewindows(r1,m,d);
            i = output{1};
            S1= output {2};
            %hitung biaya
            if i==length(r1)-1
                C11 = d(r1(w),r1(w+1)) + d(r1(w+1),r1(w+2))-
miu*d(r1(w),r1(w+2));
                C12 = S1(w+2)-S(w+1);
                C13 = m(r1(w+1),4)-S1(w+1);
                CS = alp1*(C11) + alp2*(C12) + alp3*(C13);
                if CS < C
                    C =CS;
                    Z1 = r1;
                    R1 = S1;
                    t = w;
                end
            end
        end
    end
    if C ~= inf
        D = inf;
        for v = t+1:length(Z1)-1
            r2=ins(Z1,c(p-1,2),v);
            b=0; %kapasitas kendala
            i=1;
            b=cek_kendala(r2,m,kapasitas);
            %kendala time windows
            if b<=kapasitas
                output=cek_timewindows(r2,m,d);
                w = output{1};
```

```
                S2 = output{2};
                %hitung biaya
                if w==length(r2)-1
                    C11 = d(r2(v),r2(v+1))+ d(r2(v+1),r2(v+2))-
miu*d(r2(v),r2(v+2));
                    C12 = S2(v+2) - R1(v+1);
                    C13 = m(r2(v+1),4) - S2(v+1);
                    CS = alp1*(C11)+ alp2*(C12) + alp3*(C13);
                    if CS < D
                        D = CS;
                        Z2 = r2;
                        R2 =S2;
                    end
                end
            end
        end
        if D~=inf
            u(p)=1;
            r=Z2;
        end
    end
end
r;
R{a}=r;
end
R;
distance_ins = 0;
for dis = 1 : length(R)
    for tance = 1 : length (R{dis})-1
        distance_ins = distance_ins + d(R{dis} (tance), R{dis}
(tance+1));
    end
end
Rins = R;
jum_rute_insertion = size(Rins,2);
tot_dis_insertion = distance_ins;
fprintf('\njumlah rute insertion adalah %d', jum_rute_insertion);
fprintf('\ntotal jarak insertion adalah %d\n', tot_dis_insertion);
disp ('Rute-rutenya adalah')
for i=1:length(Rins)
    disp(Rins{i}-1);
end
waktu_insertion = toc;
fprintf('================================================');
tic;
hasil=zeros(1,2);
jumlah_rute=size(Rins,2);
total_jarak=distance_ins;
R=Rins;
%%LNS
for cak=1:iterasi_LNS
cak;
D=R;
sigma_R=[jumlah_rute total_jarak];
c_copy=c;
rel_custo=zeros(1,2);
```

```
D_lingkungan=[];
del_cust=zeros(1,UK);
cust_pick=del_cust;
cust_del=del_cust;
for i = 1 : UK
    [q1 q2] = size(c_copy);
    del_cust(i)=randi([1 q1],1);
    cust_pick(i)=c_copy(del_cust(i),1);
    cust_del(i)=c_copy(del_cust(i),2);
    rel_custo(i,1)=cust_pick(i);
    rel_custo(i,2)=cust_del(i);
    c_copy(del_cust(i),:)=[];
end
c_copy;
rel_custo;
[rr cc]=size(rel_custo);

%mengecek rute keberapa yang menjadi tempat penghapusan rel_cus
asa=zeros(1,UK);
for k=1:rr
    for kk=1:length(D)
        for kkk=1:length(D{kk})-1
            if D{kk}(kkk)==rel_custo(k,1)
                D{kk}=del(D{kk},kkk);
                asa (k)=kk;
            end
            if D{kk}(kkk)==rel_custo(k,2)
                D{kk}=del(D{kk},kkk);
                break;
            end
        end
    end
end
asa;
D;

%%mulai menyisip

rel_cust=rel_custo;
uk_rel_cust=size(rel_cust,1);
sisip=randi([1 uk_rel_cust],1);
pickup1=rel_cust(sisip,1);
deliver1=rel_cust(sisip,2);
oho=0;

for ho=1:length(D)
    D_ling1=D;
    D_temp=D{ho};
    length_Dtemp=length(D_temp);
    for hoho=1:length_Dtemp-1;
        D_temp1=ins(D_temp,pickup1,hoho);
        %cek kendala
        b=cek_kendala(D_temp1,m,kapasitas);
        if b<=kapasitas
            output=cek_timewindows(D_temp1,m,d);
            i=output{1};
```

```
            S1=output{2};
            if i==length(D_temp1)-1
                for hohoho=hoho+1:length(D_temp1)-1
                    D_temp2=ins(D_temp1,deliver1,hohoho);
                    b=cek_kendala(D_temp2,m,kapasitas);
                    if b<=200
                        output= cek_timewindows(D_temp2,m,d);
                        i=output{1};
                        S1=output{2};
                        if i==length(D_temp2)-1
                            fofo=D_temp2;
                            D_ling1{ho}=fofo;
                            oho=oho+1;
                            D_lingkungan{1}{oho}=D_ling1;
                        end
                    end
                end
            end
        end
    end
end
rel_cust(sisip,:)=[];
rel_cust;

%%menyisipkan pasangan kedua hingga akhir
for iu=2:size(rel_custo,1)
    uk_rel_cust=size(rel_cust,1);
    sisip=randi([1 uk_rel_cust],1);
    pickup1=rel_cust(sisip,1);
    deliver1=rel_cust(sisip,2);
    oho=0;
    for sl=1:size(D_lingkungan{iu-1},2)
        D=D_lingkungan{iu-1}{sl};
        for ho=1:length(D)
            D_ling1=D;
            D_temp=D{ho};
            length_Dtemp=length(D_temp);
            for hoho=1:length_Dtemp-1;
                D_temp1=ins(D_temp,pickup1,hoho);
                %cek kendala
                b=cek_kendala(D_temp1,m,kapasitas);
                if b<=200
                    output=cek_timewindows(D_temp1,m,d);
                    i=output{1};
                    S1=output{2};
                    if i==length(D_temp1)-1
                        for hohoho=hoho+1:length(D_temp1)-1
                            D_temp2=ins(D_temp1,deliver1,hohoho);
                            b=cek_kendala(D_temp2,m,kapasitas);
                            if b<=kapasitas

output=cek_timewindows(D_temp2,m,d);
                                i=output{1};
                                S1=output{2};
                                if i==length(D_temp2)-1
                                    fofo=D_temp2;
```

```
                                        D_ling1{ho}=fofo;
                                        oho=oho+1;
                                        D_lingkungan{iu}{oho}=D_ling1;
                                    end
                                end
                            end
                        end
                    end
                end
            end
        end
    end
    D_lingkungan{iu};
    D_lingkungan;
    rel_cust(sisip,:)=[];
end
D_lingkungan;
Q=D_lingkungan{size(rel_custo,1)};
[la pa]=size(Q);

%%menghapus rute yang hanya terdiri dari depot
for pp=1:pa
    for ppp=length(Q{pp}):-1:1
        if length(Q{pp}{ppp})==2
            Q{pp}=del(Q{pp},ppp);
        end
    end
end
%%menghitung nilai sigma
Q_place=zeros(1,2);
for ko=1:pa
    [so to]=size(Q{ko});
    Q_place(ko,1)=to;
    distance=0;
    for fo=1:to
        for go=1:length(Q{ko}{fo})-1
            distance=distance + d(Q{ko}{fo}(go),Q{ko}{fo}(go+1));
        end
    end
    Q_place(ko,2)=distance;
end
Q_place;

%%mencari sigma paling minimum
[zz ww]=size (Q_place);
wz=min(Q_place(:,1));
az=find(Q_place(:,1)==wz);
[sz tz]=size(az);
if sz>1
    WZ=zeros(sz,2);
    for iz=1:sz
        WZ(iz,1)=Q_place(az(iz),1);
        WZ(iz,2)=Q_place(az(iz),2);
    end
    WZ;
    vz=min(WZ(:,2));
    bz=find(WZ(:,2)==vz);
```

```
    [hz lz]=size(bz);
    if hz>1
        UZ=zeros(hz,2);
        for jz=1:hz
            UZ(jz,1)=WZ(bz(jz),1);
            UZ(jz,2)=WZ(bz(jz),2);
        end
        UZ=UZ(1,:);
    end
    if hz==1
        UZ=zeros(1,2);
        UZ(1,1)=WZ(bz(1),1);
        UZ(1,2)=WZ(bz(1),2);
        UZ;
    end
end
if sz==1
    UZ=zeros(1,2);
    UZ(1,1)=Q_place(az(1),1);
    UZ(1,2)=Q_place(az(1),2);
    UZ;
end
UZ;
for ez=1:pa
    if isequal(UZ,Q_place(ez,:))==1
        ssz=ez;
        Q_baru=Q{ssz};
    end
end
R_baru=R;
if UZ(1,1)<sigma_R(1,1);
    R=Q_baru;
    jumlah_rute=UZ(1,1);
    total_jarak=UZ(1,2);
else if UZ(1,2)<sigma_R(1,2)
        R=Q_baru;
        jumlah_rute=UZ(1,1);
        total_jarak=UZ(1,2);
    end
end
isequal(R_baru,R);
R;
Rnya{cak}=R;
hasil(cak,1)=jumlah_rute;
hasil(cak,2)=total_jarak;
end
hasil;
jumlah_rute=hasil(cak,1);
total_jarak=hasil(cak,2);
fprintf('\njumlah rute adalah %d\n',hasil(cak,1));
fprintf('\ntotal jarak LNS adalah %d\n',hasil(cak,2));
disp('rute rutenya adalah');
for i=1:length(R)
    disp(R{i}-1);
end
waktu_LNS = toc
```

cek_kendala.m

```
function b = cek_kendala(R,m,kapasitas)
b=0;
i=1;
while i<=length(R) && b<=kapasitas
    b=b+m(R(i),2);
    i=i+1;
end
end
```

cek_timewindows.m

```
function output = cek_timewindows(r1,m,d)
S1(1)=0;
for i=1:length(r1)-1
    S1(i+1)=S1(i)+m(r1(i),5)+d(r1(i),r1(i+1));
    if S1(i+1)<=m(r1(i+1),3);
        S1(i+1)=m(r1(i+1),3);
    elseif S1(i+1)>m(r1(i+1),4)
        break;
    end
end
output{1}=i;
output{2}=S1;
end
```

del.m

```
function A = del(A,i)
if isempty(A)||i>length(A)
    fprintf('\nmatrix kosong atau index melebihi batas\n');
elseif i== 1
    A =A(i+1:length(A));
elseif i==length(A)
    A = A(1:length(A)-1);
else
    A=[A(1:i-1) A(i+1:length(A))];
End
```

ins.m

```
function A = ins(A,x,i)
if isempty(A)
    fprintf('\nmatrix kosong atau index melebihi batas\n');
else
    A=[A(1:i) x A(i+1:length(A))];
end
end
```

JOGJA KURIR EXPRESS
"Direct & Urgent Delivery Service"
Alamat : Jalan Sisingamangaraja MG III/1249 Yogyakarta 55153
Telp. 085643761512

SURAT KETERANGAN

No.011/JKE/01/2015

Yang bertanda tangan di bawah ini:

Nama : Dhika Ardian
Jabatan : Owner Jogja Kurir Express
Alamat : Jalan Sisingamangaraja MG III/1249, Yogyakarta 55153

Menerangkan dengan sebenarnya bahwa:

Nama : Anie Viktaria
NIM : 10305141039
Prodi : Matamatika
Fakultas : Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Yogyakarta

Telah melakukan penelitian di Jogja Kurir Express guna memperoleh data yang diperlukan sehubungan dengan penyusunan Tugas Akhir Skripsi dengan judul " Efektivitas Algoritma *Simulated Annealing* dan *Large Neighborhood Search* dalam Penyelesaian *Pickup and Delivery Vehicle Routing Problem with Time Windows* ".

Demikian surat keterangan ini dibuat untuk digunakan sebagaimana mestinya.

Yogyakarta, 22 Januari 2015

Owner Jogja Kurir Express

Dhika Ardian