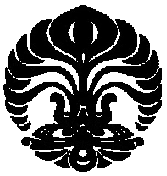


UNIVERSITAS INDONESIA

APLIKASI ALGORITMA HIBRIDA DUA TAHAP PADA *PICKUP AND DELIVERY VEHICLE ROUTING PROBLEM WITH TIME WINDOWS*

SKRIPSI

RISYA PRIWARNELA
0806452293

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
PROGRAM STUDI SARJANA MATEMATIKA
DEPOK
JULI 2012

UNIVERSITAS INDONESIA

APLIKASI ALGORITMA HIBRIDA DUA TAHAP PADA *PICKUP AND DELIVERY VEHICLE ROUTING PROBLEM WITH TIME WINDOWS*

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana sains

RISYA PRIWARNELA
0806452293

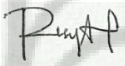
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
PROGRAM STUDI SARJANA MATEMATIKA
DEPOK
JULI 2012

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya sendiri,

dan semua sumber yang dikutip maupun dirujuk

telah saya nyatakan dengan benar

Nama	: Risya Priwarnela
NPM	: 0806452293
Tanda Tangan	: 
Tanggal	: Juni 2012

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

HALAMAN PENGESAHAN

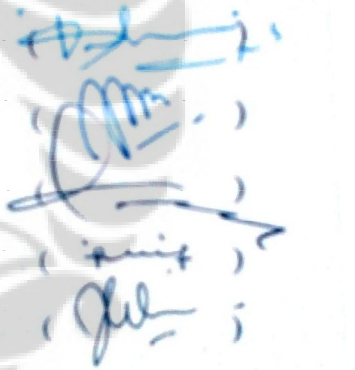
Skripsi ini diajukan oleh :

Nama	:	Risya Priwarnela
NPM	:	0806452293
Program Studi	:	Matematika
Judul Skripsi	:	Aplikasi Algoritma Hibrida Dua Tahap pada *Pickup and Delivery Vehicle Routing Problem with Time Windows*

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Sains pada Program Studi S1 Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Indonesia.

DEWAN PENGUJI

Pembimbing I	: Rahmi Rusin, S.Si, M.ScTech
Pembimbing II	: Dr. Yudi Satria, M.T.
Penguji I	: Prof. Dr. Djati Kerami
Penguji II	: Dra. Siti Aminah, M.Kom
Penguji III	: Helen Burhan, M.Si



Ditetapkan di : Depok
Tanggal : 20 Juni 2012

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

KATA PENGANTAR

Alhamdulillah, puji syukur kepada Allah SWT atas segala rahmat dan hidayah yang telah diberikan sehingga penulis dapat menyelesaikan tugas akhir ini. Shalawat dan salam selalu terucap untuk Nabi Muhammad SAW, semoga kita semua mendapat syafaat darinya kelak di hari akhir. Penulis sadar bahwa tugas akhir ini dapat terselesaikan berkat bantuan dan dukungan dari berbagai pihak. Oleh karena itu penulis ingin mengucapkan terima kasih kepada pihak-pihak yang memberikan bantuan dan dukungan selama penulis menjalani perkuliahan dan juga dalam penyelesaian tugas akhir ini. Ucapan terima kasih ditujukan kepada :

1. Rahmi Rusin, S.Si, M.ScTech dan Dr. Yudi Satria, M.T. selaku Pembimbing I dan Pembimbing II yang telah meluangkan waktu dan pikiran di tengah kesibukannya, dan selalu memberikan semangat kepada penulis untuk menyelesaikan tugas akhir ini. Semoga Allah memberi balasan terbaik untuk kebaikan serta bimbingan Bapak dan Ibu.
2. Dr. Yudi Satria, M.T. selaku Ketua Departemen, Rahmi Rusin, S.Si, M.ScTech selaku Sekretaris Departemen, dan Dr. Dian Lestari, DEA selaku Koordinator Pendidikan yang telah membantu proses penyelesaian tugas akhir ini.
3. Dra. Saskya Mary Soemartojo M.Si selaku Pembimbing Akademik yang telah meluangkan waktunya dan dengan sabar memberi saran kepada penulis tentang mata kuliah ataupun peminatan yang penulis ambil selama proses perkuliahan hingga penyelesaian tugas akhir ini.
4. Prof. Dr. Djati Kerami, Dra. Siti Aminah, M.Kom dan Helen Burhan, M.Si selaku penguji kolokium, terima kasih atas masukan dan pertanyaan yang Bapak/Ibu berikan.
5. Seluruh dosen di Departemen Matematika UI atas ilmu yang bermanfaat dan bantuan yang diberikan kepada penulis. Semoga Allah membalas semua kebaikan yang Bapak/Ibu berikan.
6. Seluruh karyawan di Departemen Matematika UI yang turut membantu selama perkuliahan dan penyelesaian tugas akhir ini. Untuk Mbak Santi,

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

karyawan TU yang baik banget udah mau diribetin dan jadi informan saat penulis kebingungan mencari pembimbing, terima kasih banyak ya Mbak.

7. Kedua orang tua penulis (Ibu dan Ayah), kakak-kakak penulis (Uda Insan, Uda Dafiq, Uda Fadli) dan kakak ipar penulis (Kak Indri dan Kak Tyas) yang senantiasa memberikan dukungan dan kasih sayang kepada penulis serta menjadi teman diskusi serta keponakan penulis (Raka, *the active little boy)*.
8. Teman - teman yang berjuang bersama dalam penulisan tugas akhir (Tuti, Ines, Numa, Mei, Qiqi, Nita, Ade, Awe, Dhila, Anisah, Maul, Uni Achi, Andy, Adhi, Bowo, Ucil, Vika, Cindy, Resti, Emy, Hindun, Janu, Dhea, Kak Anis, Oline, Ega, Eka, Icha, Sita, Luthfa, Arief, Fani, Wulan, Umbu, Kak Ayat, Kak Fauzan, Kak Putri), mari kita raih kesuksesan di masa depan.
9. Andy, Umbu, Kak Yanu, Hendry dan Hindun,teman-teman penulis yang menjadi teman diskusi serta turut membantu dalam penyelesaian program tugas akhir ini.
10. Ines, Tuti dan Numa, sobat – sobat kutek yang menjadi teman diskusi semua hal serta setia 'membunuh' malam di *coffee shop* alias warkop.
11. Keluarga Matematika angkatan 2008 atas kebersamaan yang tak mungkin terlupakan, dan semangat berjuang bersama. Semoga silaturahim kita semua akan terus berjalan hingga akhir nanti.
12. Tasya, Desy, Citra dan Dela, teman – teman kosan penulis selama 3 tahun lamanya. Semoga kita semua bisa sukses dengan bidang masing-masing.
13. Sobat-sobat SMA penulis, Pipit, Noy dan Asti yang setia berbagi cerita dan pelajaran berharga dalam hidup.
14. Seluruh keluarga besar Matematika angkatan 2009, 2010, dan 2011.
15. Semua pihak yang membantu penulis yang tidak dapat disebutkan satu persatu.

Akhir kata, penulis memohon maaf atas kesalahan dan kekurangan pada tugas akhir ini. Semoga tugas akhir ini dapat memberikan manfaat bagi penulis dan pembaca.

Penulis

2012

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI
TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini :

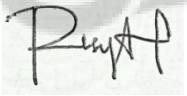
Nama	: Risya Priwarnela
NPM	: 0806452293
Program Studi	: Sarjana Matematika
Departemen	: Matematika
Fakultas	: Matematika dan Ilmu Pengetahuan Alam
Jenis karya	: Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalti Free Right*) atas karya ilmiah saya yang berjudul :
Aplikasi Algoritma Hibrida Dua Tahap pada *Pickup and Delivery Vehicle Routing Problem with Time Windows*

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/format-kan, mengelola dalam bentuk pangkalan data (database), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan pemilik Hak Cipta.

Demikian surat ini saya buat dengan sebenarnya.

Dibuat di : Depok
Tanggal : Juni 2012
Yang Menyatakan



(Risya Priwarnela)

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

ABSTRAK

Nama : Risya Priwarnela
Program Studi : Matematika
Judul : Aplikasi Algoritma Hibrida Dua Tahap pada *Pickup and Delivery Vehicle Routing Problem with Time Windows*

Pickup and Delivery Vehicle Routing Problem with Time Windows (PDPTW) adalah suatu permasalahan dalam pencarian rute optimal untuk memenuhi permintaan sejumlah pelanggan dengan setiap permintaan terdiri dari permintaan jemput dan antar. Solusi yang ingin dicapai adalah solusi dengan banyaknya rute yang minimum dan total jarak yang minimum. Tugas akhir ini membahas aplikasi algoritma hibrida dua tahap pada PDPTW dan implementasinya pada data *benchmark* Li dan Lim dengan menggunakan perangkat lunak. Tahap pertama menggunakan algoritma *simulated annealing* untuk meminimumkan banyaknya rute dengan pembentukan solusi awal menggunakan metode *insertion heuristic* dan tahap kedua menggunakan algoritma *large neighborhood search* untuk meminimumkan total jarak.

Kata Kunci : *pickup and delivery vehicle routing problem with time windows, simulated annealing, large neighborhood search, insertion heuristic*
xii + 68 halaman ; 7 gambar, 5 tabel
Daftar Referensi : 15 (1987-2011)

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

ABSTRACT

Name : Risya Priwarnela
Study Program : Mathematics
Tittle : An Application of Two-Stage Hybrid Algorithm for Pickup and Delivery Vehicle Routing Problem with Time Windows

Pickup and Delivery Vehicle Routing Problem with Time Windows (PDPTW) is a problem of finding optimal route to serve customer's demands where each demand consists of pickup and delivery service. The optimal solution is the solution with minimum number of routes and minimum total distance. This final project presents an application of two-stage hybrid algorithm for PDPTW and its implementation on Li and Lim benchmark data using software. The first stage uses simulated annealing algorithm to minimize the number of routes with insertion heuristic used in the construction of initial solution. Then, the second stage uses large neighborhood search algorithm to minimize the total distance. That algorithm is implemented for benchmark problem.

Keywords : pickup and delivery vehicle routing problem with time windows, simulated annealing, large neighborhood search, insertion heuristic
xii + 68 pages ; 7 pictures, 5 tables
Bibliography : 15 (1987-2011)

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

DAFTAR ISI

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Masalah manajemen logistik dan transportasi adalah masalah yang biasa ditemukan sehari-hari. Salah satu permasalahan yang berkaitan dengan hal ini adalah masalah penentuan rute kendaraan dalam pendistribusian barang atau jasa. *Vehicle Routing Problem* (VRP) merupakan masalah dalam mencari rute optimal untuk pengiriman atau pengumpulan barang atau jasa dari satu atau lebih depot ke sejumlah kota atau pelanggan dengan memenuhi sejumlah kendala (Yeun dkk, 2008). Masing-masing kendaraan yang digunakan dalam proses pengumpulan atau pengiriman tersebut memiliki kapasitas tertentu sehingga VRP sering juga disebut sebagai CVRP (*Capacitated Vehicle Routing Problem*) (Kallehauge dkk, 2001). Tujuan yang ingin dicapai dalam VRP adalah total biaya yang minimum. Total biaya minimum dicapai dengan meminimalkan total jarak dan meminimalkan banyaknya kendaraan yang digunakan. VRP yang memiliki kendala waktu tertentu (*time windows*) disebut *Vehicle Routing Problem with Time Windows* (VRPTW).

Pickup and Delivery Vehicle Routing Problem with Time Windows (PDPTW) adalah perumuman dari VRPTW yang bertujuan untuk membentuk rute optimal untuk memenuhi permintaan pelanggan, dimana masing-masing permintaan terdiri dari pelayanan jemput dan pelayanan antar dengan kendala kapasitas, *time windows*, dan *precedence*. *Precedence* berarti pada sebuah permintaan, pelayanan jemput dilakukan sebelum pelayanan antar. Selain itu, setiap rute harus memenuhi kendala *pairing* yaitu pelayanan jemput dan pelayanan antar dalam sebuah permintaan harus dilayani oleh kendaraan yang sama (Dumas dkk, 2001).

Algoritma yang dapat digunakan dalam menyelesaikan PDPTW terdiri dari algoritma yang bersifat eksak dan heuristik. Pada tugas akhir ini akan dibahas aplikasi algoritma yang bersifat heuristik dalam menyelesaikan PDPTW yang terbagi menjadi dua tahap. Tahap pertama dilakukan untuk meminimalkan

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

banyaknya rute atau ekivalen dengan meminimalkan banyaknya kendaraan dan tahap kedua dilakukan untuk meminimalkan biaya perjalanan yang ekivalen dengan total jarak. Kedua tahap ini dilakukan dengan alasan jika algoritma yang digunakan hanya fokus pada meminimalkan biaya perjalanan, maka solusi yang diperoleh mungkin memiliki rute yang tidak minimal. Jadi setelah dilakukan tahap pertama dan didapatkan banyaknya rute yang minimal, akan dilakukan tahap kedua untuk meminimalkan biaya perjalanan.

Algoritma yang digunakan pada tahap pertama adalah algoritma *simulated annealing*. Algoritma *simulated annealing* merupakan salah satu contoh metode heuristik yang digunakan untuk mencari pendekatan global optimum suatu masalah. Algoritma ini berbasiskan metode probabilitas yang dikembangkan oleh Kirkpatrick et al, 1983 untuk menemukan nilai minimum global dari sebuah fungsi yang memiliki beberapa nilai lokal minimum. *Simulated annealing* berjalan berdasarkan proses *annealing* pada pembuatan kristal suatu material, yaitu proses pendinginan suatu benda padat sehingga strukturnya 'membeku' pada suatu energi yang minimum (Berstimas dan Tsitsiklis, 1993). Pada pembuatan kristal suatu material, dilakukan pemanasan hingga mencapai satu titik tertentu. Saat material berada dalam keadaan yang panas, atom-atom akan bergerak bebas dengan tingkat energi yang tinggi. Kemudian dilakukan penurunan suhu secara perlahan dengan harapan atom-atom itu akan berada pada posisi yang optimum dengan energi yang minimum. Energi yang minimum inilah yang menjadi tujuan utama algoritma *simulated annealing*. Banyaknya rute pada masalah PDPTW dianalogikan sebagai energi yang ingin dicapai pada keadaan optimum.

Sedangkan algoritma tahap kedua adalah algoritma *large neighborhood search*. Algoritma *large neighborhood search* adalah algoritma pencarian solusi terbaik yang memanfaatkan metode *local search* dengan tujuan memperbaiki solusi yang telah ada dengan cara mengevaluasi nilai fungsi tujuan. Pada algoritma *large neighborhood search,* evaluasi nilai fungsi tujuan dilakukan dengan cara membuat lingkungan dari suatu solusi sekarang kemudian menghitung nilai fungsi solusi lingkungan tersebut untuk dibandingkan dengan solusi yang telah ada hingga didapat solusi terbaik.

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

Penentuan solusi awal dalam algoritma *simulated annealing* pada tugas akhir ini dilakukan dengan metode *insertion heuristic*. Pada tahap pertama dilakukan algoritma *simulated annealing* dengan menggunakan solusi awal tersebut kemudian pada tahap kedua dilakukan algoritma *large neighborhood search* menggunakan solusi awal yang didapat dari solusi akhir algoritma *simulated annealing*. Dengan penggabungan dua algoritma tesebut maka algoritma ini disebut juga sebagai algoritma hibrida dua tahap.

1.2 Perumusan Masalah dan Ruang Lingkup

Perumusan masalah tugas akhir ini adalah bagaimana menjelaskan aplikasi algoritma hibrida dua tahap pada masalah *Pickup and Delivery Vehicle Routing Problem with Time Windows*.

Ruang lingkup pada tugas akhir ini adalah implementasi algoritma hibrida dua tahap pada data *benchmark* (Li & Lim *benchmark*) dengan 100 dan 200 pelanggan.

1.3 Metode Penulisan

Metode yang digunakan dalam penulisan tugas akhir ini adalah dengan studi literatur dan simulasi.

1.4 Tujuan Penulisan

Penulisan tugas akhir ini bertujuan untuk menjelaskan aplikasi algoritma hibrida dua tahap pada permasalahan *Pickup and Delivery Vehicle Routing Problem with Time Windows* dan mengimplementasikan algoritma tersebut pada data *benchmark* (Li & Lim *benchmark*) dengan menggunakan perangkat lunak.

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

BAB 2
LANDASAN TEORI

Pada Bab 2 ini akan dijelaskan mengenai beberapa teori yang akan digunakan pada bab berikutnya. Pada Subbab 2.1 akan dijelaskan mengenai *Vehicle Routing Problem with Time Windows* (VRPTW), kemudian dilanjutkan dengan pembahasan mengenai *Pickup and Delivery Vehicle Routing Problem with Time Windows* (PDPTW) pada Subbab 2.2. Setelah itu pada Subbab 2.3 akan dijelaskan algoritma yang digunakan pada tahap pertama yaitu *simulated annealing* dan terakhir algoritma *large neighborhood search* yang digunakan pada tahap kedua akan dijelaskan pada Subbab 2.4.

2.1 *Vehicle Routing Problem with Time Windows*

Dalam dunia logistik atau pendistribusian barang atau jasa dibutuhkan suatu penentuan rute kendaraan untuk menunjang kelancaran pelayanan barang atau jasa tersebut. Masalah ini dikenal dengan sebutan *Vehicle Routing Problem* (VRP). Menurut Yeun dkk (2008), VRP merupakan masalah pencarian rute optimal untuk pengiriman atau pengumpulan barang atau jasa dari satu atau lebih depot ke sejumlah kota atau pelanggan dengan memenuhi kendala tertentu. Depot merupakan tempat kendaraan memulai dan mengakhiri perjalanan pendistribusian barang atau jasa. Selain itu setiap kota atau pelanggan dikunjungi tepat satu kali. Pada VRP, masing-masing kendaraan yang digunakan untuk melakukan pelayanan barang atau jasa memiliki kapasitas tertentu sehingga sering disebut juga sebagai *Capacitated Vehicle routing problem* (CVRP) (Kallehauge dkk, 2001). Jika VRP memiliki suatu kendala waktu yang disebut *time windows* saat pendistribusian barang atau jasa maka VRP disebut *Vehicle Routing Problem with Time Windows* (VRPTW). *Time windows* pada pelanggan adalah sebuah interval waktu diperbolehkannya kendaraan untuk memulai pelayanan. Apabila kendaraan telah sampai di pelanggan sebelum waktu awal diperbolehkannya kendaraan untuk memulai pelayanan maka kendaraan harus menunggu untuk memulai pelayanan.

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

2.2 *Pickup and Delivery Vehicle Routing Problem with Time Windows*

Pick up and Delivery Vehicle Routing Problem with Time windows (PDPTW) adalah masalah penentuan rute kendaraan yang merupakan perumuman dari VRPTW (Dumas dkk, 2001). PDPTW yang merupakan generalisasi dari VRPTW termasuk ke dalam masalah *NP-hard* karena VRP dikenal sebagai masalah *NP-hard* (Lau dan Liang, 2001). Masalah *NP-hard* adalah masalah yang memerlukan waktu komputasi yang cukup lama dalam mendapatkan solusinya karena masalah *NP-hard* tidak dapat diselesaikan dalam waktu polinomial. Pada PDPTW sebuah permintaan dari pelanggan terdiri dari pelayanan jemput dan pelayanan antar barang atau jasa. Selain itu rute optimal yang ingin dicapai dalam memenuhi permintaan harus memenuhi kendala kapasitas kendaraan, *time windows, precedence* dan *pairing* (Dumas dkk,2001). Kendala kapasitas yang dimaksud adalah setiap kendaraan memiliki kapasitas tertentu dan jika kapasitas kendaraan sudah penuh, maka kendaraan tidak dapat melayani pelayanan jemput selanjutnya. Namun setelah melakukan pelayanan antar kendaraan bisa melakukan pelayanan jemput yang lain selama muatan yang berada pada kendaraan belum mencapai kapasitas. Sedangkan kendala *time windows* yang harus dipenuhi adalah interval waktu yang ditentukan bagi setiap kendaraan untuk dapat memulai pelayanan. Selain pelanggan, depot juga memiliki kendala *time window* yaitu sebuah interval waktu yang menunjukan waktu awal keberangkatan kendaraan dari depot dan waktu kembalinya kendaraan ke depot.

Dalam sebuah permintaan, pelayanan jemput harus dilakukan sebelum pelayanan antar. Contohnya pelanggan i ingin barangnya diantar ke pelanggan j, maka kendaraan harus ke pelanggan i terlebih dahulu untuk mengambil barang yang akan diantar baru kemudian ke pelanggan j sebagai tempat tujuan pengiriman barang. Kendala seperti ini yang dimaksud dengan kendala *precedence*. Selain itu pada sebuah permintaan, pelayanan jemput-antar harus dilakukan oleh kendaraan yang sama, artinya tidak ada pemindahan barang atau jasa dari satu kendaraan ke kendaraan lain. Misalnya pelanggan i ingin diantarkan barangnya ke pelanggan j, maka kendaraan yang menjemput barang di pelanggan

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

i dan mengantar barang ke pelanggan j harus sama. Hal ini yang dimaksud dengan kendala *pairing*. Sehingga PDPTW adalah sebuah masalah pencarian rute optimal dengan permintaannya terdiri atas jemput antar dan memenuhi kendala kapasitas, *time windows*, *precedence* dan *pairing*.

Berikutnya pada Subbab 2.2.1 akan diberikan definisi variabel yang digunakan dalam formulasi PDPTW dan pada Subbab 2.2.2 akan diberikan formulasi PDPTW. Pendefinisian variabel serta formulasi PDPTW pada tugas akhir ini mengacu kepada Dumas, Y., dkk (1991), Bent, R., Hentenryck, P. V. (2003) dan Desaulniers, G., dkk (2002).

2.2.1 Definisi Variabel PDPTW

Misalkan terdapat n permintaan dan jika i adalah simpul untuk pelanggan jemput, maka $@i$ adalah pelanggan antar, sedangkan 0 dan $@0$ adalah simpul untuk depot. Pada tugas akhir ini yang dibahas adalah PDPTW dengan depot tunggal. Himpunan $N = \{0, 1, 2, \dots, n, @1, @2, \dots, @n, @0\}$ adalah kumpulan simpul untuk semua pelanggan dan depot. Himpunan $P^+ = \{1, 2, \dots, n\}$ adalah kumpulan simpul untuk pelanggan jemput dan himpunan $P^- = \{@1, @2, \dots, @n\}$ adalah kumpulan simpul untuk pelanggan antar. Sedangkan $P = P^+ \cup P^-$ adalah kumpulan simpul pelanggan jemput dan antar.

Setiap pelanggan i memiliki permintaan sejumlah q_i yang harus diantar dari pelanggan i ke pelanggan $@i$ dan pelanggan $@i$ menerima permintaan dari pelanggan i sebesar $q_{@i}$ yang nilainya adalah $-q_i$. Selanjutnya misalkan $[a_i, b_i]$ adalah *time window* pelayanan jemput ke pelanggan i, $[a_{@i}, b_{@i}]$ menotasikan *time window* pelayanan antar untuk pelanggan i dan $[a_0, b_0]$ menotasikan *time window* kendaraan berangkat dari depot sedangkan $[a_{@0}, b_{@0}]$ adalah *time window* untuk kedatangan kembali kendaraan ke depot. V adalah himpunan kendaraan yang digunakan pada rute untuk melayani pelanggan dengan kendaraan sebanyak $|V|$. Diasumsikan kapasitas tiap kendaraan $v \in V$ homogen yaitu Q. Kemudian untuk setiap $i, j \in N$, t_{ij}^v merepresentasikan waktu perjalanan dari pelanggan i ke j oleh kendaraan v, c_{ij}^v adalah biaya perjalanan dari i ke j oleh kendaraan v dan s_i^v adalah waktu pelayanan pada pelanggan i oleh kendaraan v.

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

Tiga variabel yang digunakan dalam formulasi, yaitu

1. Variabel biner $X_{ij}^{v}, v \in V,\ i,j \in N, i \neq j$

$$X_{ij}^{v} = \begin{cases} 1, & \text{jika kendaraan } v \text{ berjalan dari } i \text{ ke } j \\ 0, & \text{lainnya} \end{cases}$$

2. Variabel waktu $T_i^v, i \in P$ dan $T_0^v, v \in V$

   T_i^v adalah waktu dimulainya pelayanan untuk pelanggan i oleh kendaraan v sedangkan T_0^v adalah waktu saat kendaraan v meninggalkan depot dan $T_{@0}^v$ adalah waktu saat kendaraan v kembali ke depot.

3. Variabel muatan $Y_i^v, i \in P, v \in V$.

   Y_i^v adalah total muatan dalam kendaraan v setelah meninggalkan pelanggan i, diasumsikan kendaraan memiliki muatan kosong saat berangkat dari depot atau $Y_0^v = 0$.

Ketiga variabel di atas merupakan variabel keputusan dengan kendala non negatif, artinya nilai setiap variabel pada kendalanya tidak ada yang negatif.

2.2.2 Formulasi PDPTW

$$\text{Min} \sum_{v \in V} \sum_{i,j \in N} c_{ij}\ X_{ij}^{v} \qquad (2.1)$$

d.k.

$$\sum_{v \in V} \sum_{j \in P \cup \{@0\}} X_{ij}^{v} = 1,\ \ i \in P \qquad (2.2)$$

$$\sum_{j \in N} X_{ij}^{v} - \sum_{j \in N} X_{j@i}^{v} = 0,\ \ i \in P^{+},\ \ v \in V \qquad (2.3)$$

$$\sum_{j \in P} X_{0j}^{v} = 1,\ \ v \in V \qquad (2.4)$$

$$\sum_{i \in P} X_{i,@0}^{v} = 1,\ \ v \in V \qquad (2.5)$$

$$\sum_{i \in P \cup \{0\}} X_{ij}^{v} - \sum_{i \in P \cup \{@0\}} X_{ji}^{v} = 0,\ \ j \in P,\ \ v \in V \qquad (2.6)$$

$$T_i^v + s_i^v + t_{i,@i}^v \leq T_{@i}^v,\ \ i \in P^{+} \qquad (2.7)$$

$$X_{ij}^{v} = 1 \longrightarrow T_i^v + s_i^v + t_{ij}^v \leq T_j^v,\ \ i,j \in N,\ \ v \in V \qquad (2.8)$$

$$a_i \leq T_i^v \leq b_i,\ \ i \in N\ \ \ ,v \in V \qquad (2.9)$$

$$X_{ij}^{v} = 1 \longrightarrow Y_i^v + q_j = Y_j^v, i \in P \cup \{0\}, j \in P \cup \{@0\}, v \in V \qquad (2.10)$$

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

$$0 \leq Y_{@i}^{v} \leq Q - q_i, \ @i \in P^-, \quad v \in V \tag{2.11}$$

$$Y_0^v = 0, \quad q_i \leq Y_i^v \leq Q, \ i \in P^+ \tag{2.12}$$

$$X_{ij}^v \text{ biner}, \ i,j \in N, \ v \in V \tag{2.13}$$

Model di atas merupakan masalah pemrograman bilangan bulat biner. Kendala (2.2) dan (2.3) menunjukan bahwa setiap permintaan akan dilayani tepat satu kali dan dengan kendaraan yang sama. Kendala (2.4) dan (2.5) menunjukan bahwa setiap kendaraan akan memulai rute dari depot dan akan kembali ke depot. Kendala (2.6) menunjukan bahwa kendaraan yang mengunjungi pelanggan j akan meninggalkan pelanggan j. Kendala (2.7) menunjukan kendala *precedence*, yaitu pada suatu permintaan, pelanggan jemput dikunjungi sebelum pelanggan antar. Kendala (2.8) berarti apabila ada perjalanan dari i ke j maka waktu dimulainya pelayanan pada pelanggan j lebih besar atau sama dengan dari waktu dimulainya pelayanan pelanggan i ditambah waktu pelayanan pelanggan i ditambah waktu tempuh dari i ke j. Sedangkan kendala (2.9) adalah kendala *time window* untuk setiap masing-masing pelanggan dan depot artinya waktu memulai pelayanan harus berada pada selang waktu atau *time window* yang ditentukan. Kendala (2.10) menunjukan bahwa muatan kendaraan setelah meninggalkan pelanggan i ditambah dengan muatan yang diambil pada pelanggan j bila j adalah pelanggan jemput atau dikurangi dengan muatan yang diantar pada pelanggan j bila j adalah pelanggan antar adalah muatan kendaraan setelah meninggalkan j. Sedangkan kendala (2.11) menunjukkan bahwa muatan kendaraan setelah meninggalkan pelanggan antar $@i$ lebih besar sama dengan 0 dan lebih kecil sama dengan kapasitas kendaraan dikurangi muatan yang harus diantar dari pelanggan i ke $@i$. Dan kendala (2.12) menunjukan kendala kapasitas kendaraan, yaitu muatan di dalam kendaraan setelah kendaraan meninggalkan pelanggan tidak boleh melebihi kapasitas kendaraan.

Berdasarkan Mitrovic-Minic (1998) dan Dridi dkk (2011), beberapa algoritma yang digunakan untuk penyelesaian PDPTW terdiri dari algoritma yang bersifat eksak dan heuristik. Contoh algoritma yang bersifat eksak adalah *dynamic programming* dan *column generation*. Sedangkan untuk algoritma yang bersifat heuristik dibagi ke dalam tiga macam yaitu *construction heuristic*, *improvement heuristic* dan *metaheuristic*. Untuk *construction heuristic* contohnya adalah

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

clustering (1980, 1985), *mini-clustering* (1989, 1991) dan *insertion* (1986, 1990). Algoritma yang termasuk *improvement heuristic* adalah *local search* (1983), *variable dept arc exchange* (1993), dan *cyclic transfer* (1993). Sedangkan untuk algoritma yang termasuk dalam *metaheuristic* antara lain *constraint-directed search* (1992), algoritma genetika (2008), *tabu search* (2005), dan *simulated annealing* (1993).

Pada tugas akhir ini akan digunakan algoritma hibrida dua tahap dalam menyelesaikan PDPTW yaitu *simulated annealing* dan *large neighborhood search*. Algoritma *simulated annealing* dimulai dengan pembuatan solusi awal. Pembuatan solusi awal pada tugas akhir ini dilakukan dengan menggunakan algoritma *insertion heuristic*. Lalu solusi yang didapatkan dari algoritma *simulated annealing* akan dipakai sebagai solusi awal untuk algoritma *large neighborhood search*.

2.3 *Simulated Annealing*

Algoritma *simulated annealing* (SA) adalah algoritma yang dikenalkan oleh Metropolis et al, 1953. Algoritma ini berbasiskan metode probabilitas yaitu penerimaan solusi baru berdasarkan probablitas tertentu yang dikembangkan oleh Kirkpatrick et al, 1983 untuk menemukan nilai minimum global dari sebuah fungsi yang memiliki beberapa nilai lokal minimum. *Simulated annealing* diadaptasi dari proses *annealing* pada pembuatan kristal suatu material, yaitu proses pendinginan suatu benda padat sehingga strukturnya 'membeku' pada suatu energi yang minimum (Bertsimas dan Tsitsiklis, 1993). Pada pembuatan kristal suatu material, dilakukan pemanasan hingga mencapai satu titik tertentu. Saat material berada dalam keadaan yang panas, atom-atom akan bergerak bebas dengan tingkat energi yang tinggi. Kemudian dilakukan penurunan suhu secara perlahan dengan harapan atom-atom itu akan berada pada posisi yang optimum dengan energi yang minimum. Algoritma SA dapat dipandang sebagai algoritma *local search* yang terkadang bergerak menuju solusi yang biayanya lebih besar atau solusi yang tidak lebih baik dengan harapan pergerakan ini dapat mengeluarkan keadaan dari lokal minimum (Bertsimas dan Tsitsiklis, 1993). Hal ini menjadi salah satu keunggulan algoritma SA.

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

Berdasarkan Tospornsampan dkk (2007), algoritma SA bertujuan untuk meminimumkan suatu fungsi yang dimulai dengan penentuan solusi awal yang dianggap sebagai solusi sekarang serta penentuan suhu awal. Dari solusi awal tersebut dibangun sejumlah lingkungan solusi layak yang didapatkan dengan cara penyusunan ulang secara acak dari solusi yang ada. Misalkan σ adalah solusi sekarang dan σ' adalah solusi lingkungan kemudian $e(\sigma)$ dan $e(\sigma')$ masing-masing adalah nilai fungsi dari solusi σ dan solusi lingkungan σ'. Solusi lingkungan σ' dikatakan lebih baik jika $\Delta\sigma < 0$ dengan $\Delta\sigma = e(\sigma') - e(\sigma)$ adalah selisih nilai fungsi objektif dari solusi lingkungan yang baru σ' dengan solusi sekarang σ. Sedangkan jika solusi lingkungan tidak lebih baik yaitu $\Delta\sigma > 0$, maka terdapat dua kemungkinan :

1. Solusi lingkungan diterima
   Solusi lingkungan tersebut dapat diterima dengan probabilitas $exp\ (-\Delta\sigma/H)$, dengan H adalah suhu saat itu yang menjadi pengontrol apakah solusi baru diterima atau tidak. Penerimaan solusi lingkungan yang tidak lebih baik dapat terjadi bila suatu bilangan random P antara 0 dan 1, nilainya kurang dari $exp\ (-\Delta\sigma/H)$.
2. Solusi tidak diterima
   Jika P tidak lebih kecil dari $exp(-\Delta\sigma/H)$ maka solusi sekarang tidak berubah.

Pada suhu yang sama dilakukan penyusunan ulang solusi lingkungan yang layak dan akan diulangi langkah yang sama hingga mencapai iterasi yang diinginkan. Setelah itu dilakukan penurunan suhu dengan suatu parameter tertentu, proses tadi akan terus berulang hingga tercapai kriteria penghentian algoritma.

Menurut Tospornsampan dkk (2007), terdapat tiga komponen dalam algoritma SA :

1. Proses *Annealing*
   Proses ini merupakan proses utama dalam algoritma SA. Proses *annealing* pada SA bertujuan agar sistem tidak terjebak dalam lokal minimum dan bergantung pada parameter berikut :

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

a. Suhu awal

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

b. Jumlah iterasi pada tiap suhu

Pada setiap suhu, ditentukan sejumlah iterasi tertentu, L_t. Jika sudah mencapai iterasi L_t pada suatu suhu, baru kemudian suhu akan diturunkan. Jumlah iterasi ini dapat konstan ataupun berubah-ubah dalam tiap suhunya.

c. Pemilihan parameter untuk menurunkan suhu

Setelah dilakukan sejumlah iterasi maka dilakukan penurunan suhu dengan menggunakan parameter α yang disebut *cooling rate* atau parameter penurunan suhu yang berkisar antara 0 dan 1. Nilai α yang disarankan Kirkpatrick et al, 1983 adalah antara 0.8 dan 0.99.

Proses penurunan suhu secara perlahan dituliskan sebagai berikut :

$$H_t = \alpha H_{t-1}$$

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

2. Penyusunan Ulang

Penyusunan ulang atau pembuatan solusi lingkungan dilakukan secara acak dengan cara mengubah solusi yang ada dengan solusi yang baru. Prosedur ini dilakukan dengan berbagai cara tergantung pada masalahnya.

3. Penghentian Algoritma

Agar proses *annealing* berhenti, dibutuhkan suatu kriteria yang ditentukan sejak awal proses. Kriteria tersebut dapat berupa banyaknya iterasi, dimana proses akan berhenti jika tidak ada solusi baru yang dapat diterima hingga mencapai iterasi tersebut atau kriteria lainnya adalah suhu minimum dimana proses akan berhenti saat mencapai suhu minimum tersebut.

Berikut adalah langkah algoritma SA secara umum :

1. Tentukan solusi awal, σ
2. Tentukan suhu awal H_0, suhu akhir H_t, *cooling rate* α dan jumlah iterasi L_t
3. Bentuk solusi lingkungan σ'
4. Hitung $\Delta\sigma = e(\sigma') - e(\sigma)$

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

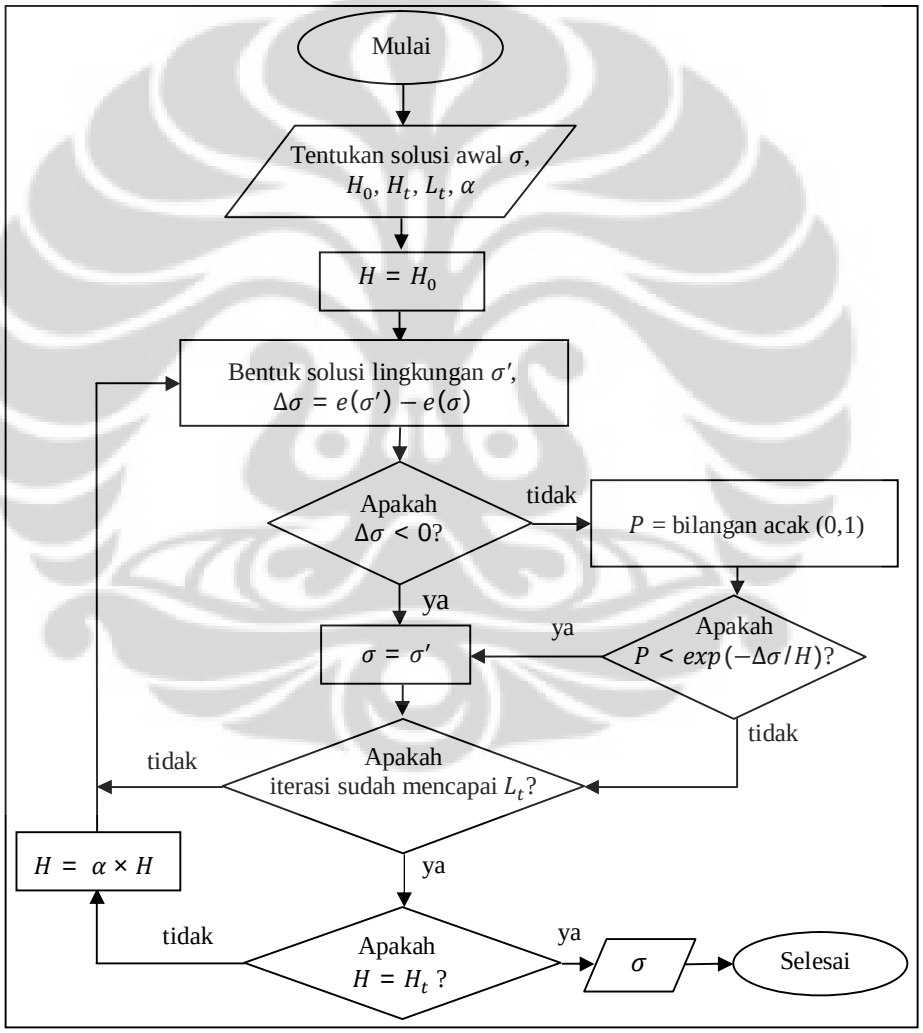
- Jika $\Delta\sigma < 0$ maka $\sigma = \sigma'$
- Jika $\Delta\sigma \geq 0$

  P = random (0,1)

  Jika $P < exp(-\Delta\sigma/H)$ maka $\sigma = \sigma'$

5. Ulangi langkah 3 dan 4 hingga iterasi yang diinginkan
6. Lakukan penurunan suhu H dan ulangi langkah 3, 4, 5 hingga mencapai suhu minimum.

Berikut akan disajikan *flowchart* algoritma SA secara umum



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

Gambar 2.1 *Flowchart* Algoritma SA

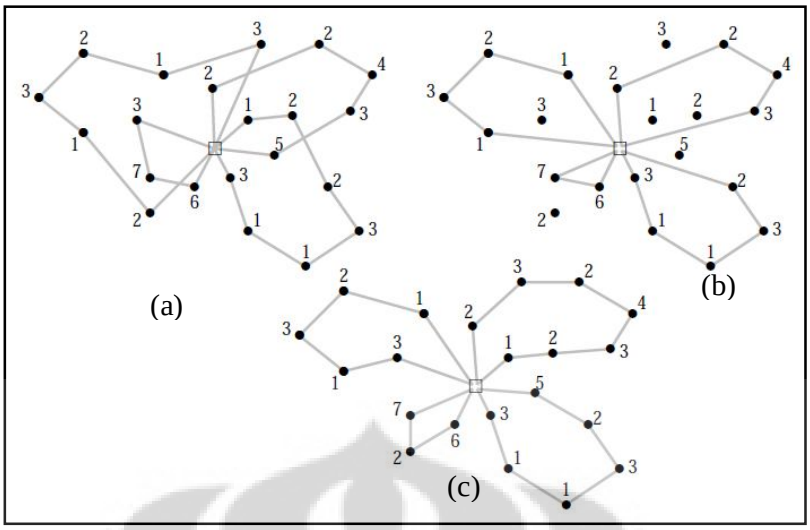
Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

2.4 *Large Neighborhood Search*

Large neighborhood search (LNS) pertama kali dikemukakan oleh Paul Shaw, 1998 (Pisinger dan Ropke, 2010). Algoritma LNS adalah algoritma pencarian solusi terbaik yang memanfaatkan metode *local search*, yaitu metode pencarian solusi berdasarkan lingkungan dari solusi awal. Pada algoritma LNS, evaluasi fungsi dilakukan dengan cara membuat lingkungan dari suatu solusi awal terlebih dahulu kemudian memilih solusi dari lingkungan tersebut untuk dievaluasi pada fungsi tujuan lalu dibandingkan dengan solusi awal hingga mencapai solusi terbaik. Berdasarkan Pisinger dan Ropke (2010), pembuatan lingkungan pada LNS dilakukan dengan metode penghapusan dan perbaikan. Metode penghapusan akan merusak atau merubah solusi yang telah ada sedangkan metode perbaikan akan membangun kembali solusi yang telah diubah oleh metode penghapusan. Solusi lingkungan $N(\sigma')$ dari σ adalah sebuah himpunan solusi yang didapat dengan menerapkan metode penghapusan dan metode perbaikan.

Pisinger dan Ropke (2010) memberi contoh penggambaran metode penghapusan dan perbaikan dalam *Capacitated Vehicle Routing Problem* (CVRP). Metode penghapusan pelanggan pada solusi dapat menyebabkan pemotongan rute dimana pelanggan tersebut berada. Cara paling sederhana yang digunakan adalah dengan memilih pelanggan yang akan dihapus secara acak. Dan metode perbaikan dapat dilakukan dengan menyisipkan kembali pelanggan ke dalam rute dengan *greedy heuristic*. Sebuah cara sederhana dalam menyisipkan pelanggan adalah dengan menyisipkan pelanggan yang telah dihapus ke dalam rute dengan biaya penyisipan terkecil hingga semua pelanggan berada dalam rute. Berikut adalah contoh yang memperlihatkan langkah dalam menerapkan metode penghapusan dan perbaikan dalam masalah CVRP.

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012



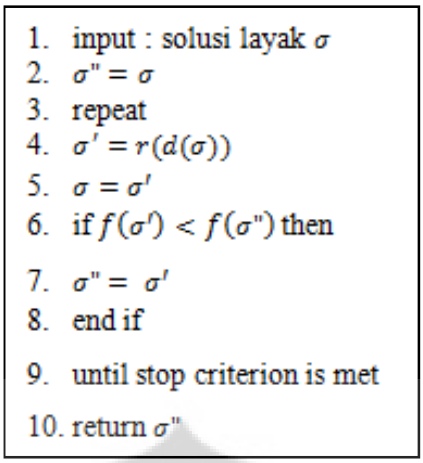
[Sumber : Pisinger dan Ropke (2010)]

Gambar 2.2 Contoh Penghapusan dan Perbaikan pada CVRP

Gambar 2.1 (a) memperlihatkan solusi CVRP sebelum dilakukan penghapusan. Sedangkan Gambar 2.1 (b) memperlihatkan solusi setelah dilakukan penghapusan 6 pelanggan dari rutenya masing - masing yaitu pelanggan 1, 2, 2, 3, 3, dan 5 sehingga terbentuk solusi yang belum mencakup semua pelanggan. Dan terakhir adalah Gambar 2.1 (c) memperlihatkan solusi setelah dilakukan metode perbaikan dengan menyisipkan kembali 6 pelanggan yang sebelumnya telah dihapus ke dalam solusi yang belum mencakup semua pelanggan.

Berikut akan diberikan penjelasan tentang LNS secara lebih detail. Tiga variabel yang digunakan dalam algoritma LNS adalah σ'', σ, dan σ'. Variabel σ'' adalah solusi terbaik yang didapatkan selama dilakukan pencarian, σ adalah solusi sekarang, dan σ' adalah solusi sementara yang dapat dibuang ataupun dipakai dan menggantikan solusi sekarang. Fungsi $d(.)$ adalah fungsi yang digunakan untuk menghapus sebagian komponen dalam solusi sedangkan $r(.)$ adalah fungsi yang digunakan untuk memperbaiki solusi. Secara spesifik, $d(\sigma)$ adalah solusi σ yang telah dikenakan penghapusan. Sedangkan $r(d(\sigma))$ digunakan untuk memperbaiki solusi yang telah berubah sehingga kembali didapatkan solusi yang layak. *Pseudocode* algoritma LNS dapat dilihat dalam Gambar 2.3 berikut.

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

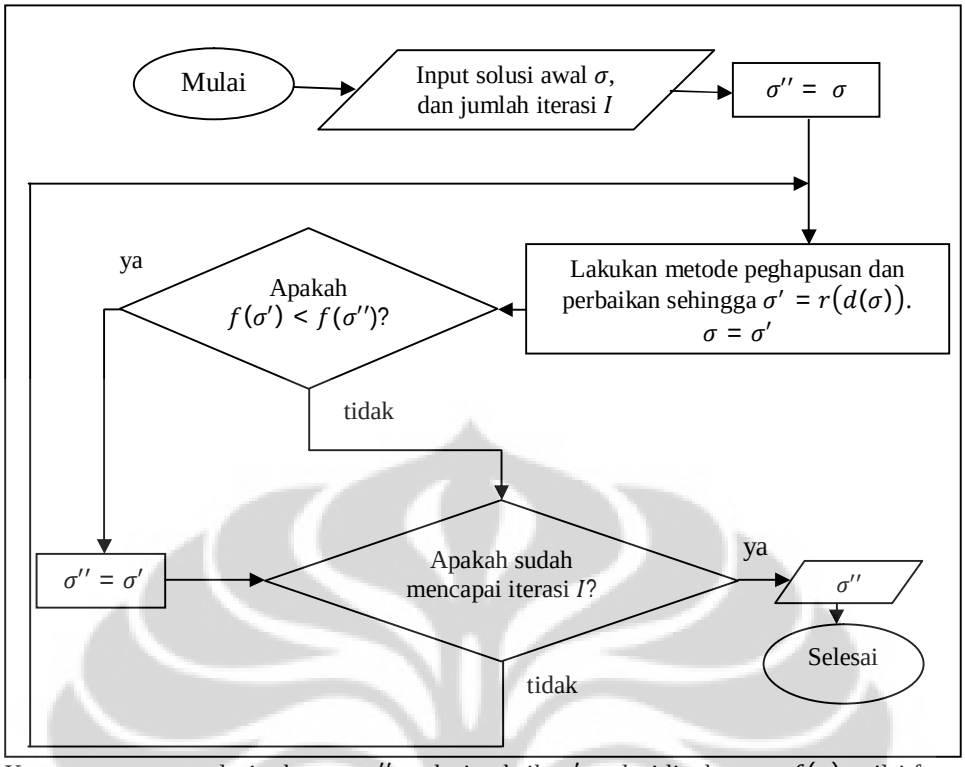

1. input : solusi layak σ
2. $\sigma" = \sigma$
3. repeat
4. $\sigma' = r(d(\sigma))$
5. $\sigma = \sigma'$
6. if $f(\sigma') < f(\sigma")$ then
7. $\sigma" = \sigma'$
8. end if
9. until stop criterion is met
10. return $\sigma"$

[Sumber : Pisinger dan Ropke (2010)]

Gambar 2.3 *Pseudocode* LNS secara umum "telah diolah kembali"

Algoritma dimulai dengan memasukkan solusi layak σ, lalu pada baris 2 dilakukan inisialisasi solusi terbaik global. Baris 4 memperlihatkan proses dilakukannya metode penghapusan dan perbaikan sehingga didapat solusi σ' yang layak. Pada baris 5 dilakukan inisialisasi terhadap σ kemudian pada baris 6 akan ditentukan apakah solusi tersebut akan diterima menggantikan solusi sekarang atau solusi tersebut akan ditolak. Fungsi untuk menerima σ' sebagai solusi baru dapat diimplementasikan dengan berbagai cara. Cara yang sederhana adalah dengan menerima solusi σ' apabila σ' lebih baik dari solusi sekarang. Setelah itu solusi tersebut akan dibandingkan dengan solusi terbaik yang ada. Nilai fungsi objektif solusi σ dinotasikan dengan $f(\sigma)$. Pada baris 7, solusi akan diperbaharui jika $f(\sigma')$ lebih baik dari $f(\sigma")$. Pada baris 9 akan dilakukan pengecekan kriteria dalam penghentian proses algoritma. Kriteria penghentian algoritma dilakukan dengan beberapa cara. Biasanya kriteria yang digunakan adalah dengan membatasi jumlah iterasi ataupun waktu iterasi. Pada artikel LNS Paul Shaw (1998), solusi baru yang diterima adalah solusi yang memperbaiki solusi sebelumnya. Namun pada pada artikel lain seperti Pisinger dan Ropke, 2006 solusi baru dapat diterima dengan kriteria yang mengadaptasi kriteria pada algoritma *simulated annealing* yaitu penerimaan solusi lingkungan yang lebih baik atau menerima solusi lingkungan yang tidak lebih baik dengan probabilitas tertentu (Pisinger dan Ropke, 2010). Berikut adalah gambar *flowchart* algoritma LNS secara umum

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012



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

Gambar 2.4 *Flowchart* Algoritma LNS

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

BAB 3
PENGGUNAAN ALGORITMA HIBRIDA DUA TAHAP DALAM MENYELESAIKAN *PICKUP AND DELIVERY VEHICLE ROUTING PROBLEM WITH TIME WINDOWS* (PDPTW)

Pada bab ini akan dibahas mengenai algoritma hibrida dua tahap dalam menyelesaikan *Pickup and Delivery Vehicle Routing Problem with Time Windows* (PDPTW). Pada Subbab 3.1 akan dijelaskan mengenai representasi solusi pada masalah PDPTW, selanjutnya pada Subbab 3.2 akan dijelaskan metode yang digunakan dalam membentuk solusi awal yaitu metode *insertion heuristic* serta penggunaannya dalam PDPTW. Berikutnya pada Subbab 3.3 akan dijelaskan contoh penyelesaian masalah PDPTW pada tahap pertama yaitu menggunakan algoritma *simulated annealing* (SA) dan pada Subbab 3.4 akan dijelaskan tahap selanjutnya yaitu menggunakan algoritma *large neighborhood search* (LNS) dalam menyelesaikan PDPTW.

3.1 Representasi Solusi

Solusi pada masalah PDPTW dinotasikan dengan σ yaitu himpunan rute dengan setiap rutenya terdiri dari depot dan sejumlah pelanggan. Himpunan σ dapat ditulis sebagai berikut : σ = { Rute 1, Rute 2,..,Rute *n*} dengan *n* adalah banyaknya rute dengan d_{ij} adalah jarak i ke j. Sebagai contoh untuk sebuah masalah sederhana yang terdiri dari 3 permintaan dengan 3 pelanggan jemput yaitu i, j, dan k serta 3 pelanggan antar yaitu $@i$, $@j$, dan $@k$ didapatkan sebuah solusi yang terdiri dari 2 rute dengan rute 1 terdiri atas pelanggan i, j, $@i$, dan $@j$ serta rute 2 terdiri atas pelanggan k dan $@k$. Maka bentuk representasi solusinya adalah $\sigma = \{0 - i - j - @i - @j - 0, 0 - k - @k - 0\}$ dengan

Rute 1 = $0 - i - j - @i - @j - 0$

Rute 2 = $0 - k - @k - 0$

Banyak rute = 2

Total jarak = $d_{0i} + d_{ij} + d_{j@i} + d_{@i,@j} + d_{j0} + d_{0k} + d_{k,@k} + d_{@k,0}$

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

3.2 *Insertion Heuristic* dalam Pembentukan Solusi Awal

Algoritma hibrida dua tahap dalam menyelesaikan PDPTW dimulai dengan membentuk solusi awal menggunakan *insertion heuristic* yang akan dijelaskan di Subbab 3.2.1 dan implementasinya pada Subbab 3.2.2.

3.2.1 *Insertion Heuristic*

Menurut Rosenkrantz dkk (1977), metode *insertion heuristic* merupakan metode yang terbukti populer dalam menyelesaikan berbagai masalah penentuan rute kendaraan (*Vehicle Routing Problem*) dan masalah penjadwalan. Algoritma ini pertama kali dikenalkan dan dianalisa sebagai metode yang populer untuk masalah optimisasi pada masalah *Travelling Salesman Problem* (Campbell dan Martin, 2004). Metode *insertion heuristic* membentuk solusi layak yaitu sebuah himpunan rute dimulai dengan memilih pelanggan pertama untuk masuk ke dalam rute lalu memasukkan pelanggan-pelanggan lain yang belum masuk ke rute secara berulang hingga seluruh pelanggan masuk ke dalam rute. Ada dua hal yang harus diputuskan setiap iterasi *insertion heuristic*, yaitu pelanggan mana yang akan disisipkan serta di bagian mana pada rute pelanggan tersebut akan disisipkan.

Pemilihan pelanggan pertama untuk masuk ke dalam rute dilihat berdasarkan jarak terjauh dari depot, atau pelanggan yang harus segera dilayani (Joubert dan Claseen, 2006). Setelah didapatkan sebuah rute awal, metode *insertion heuristic* pada Solomon (1987) menggunakan dua fungsi pada setiap iterasi untuk menyisipkan satu pelanggan diantara dua pelanggan lain. Fungsi tersebut menggambarkan biaya yang dihitung dalam menyisipkan pelanggan.

Terdapat tiga cara menghitung biaya penyisipan pelanggan dalam *insertion heuristic* (Solomon, 1987), yaitu :

1. $c_1(i,u,j) = \alpha_1\, c_{11}(i,u,j) + \alpha_2\, c_{12}(i,u,j), \alpha_1 + \alpha_2 = 1, \alpha_1 \geq 0, \alpha_1 \geq 0$
   $c_2(i,u,j) = \lambda d_{0u} - c_1(i,u,j), \lambda \geq 0$
   $c_{11}(i,u,j)$ menunjukan penambahan jarak yang dihasilkan jika pelanggan u disisipkan diantara pelanggan i dan j. Sedangkan $c_{12}(i,u,j)$ menunjukan

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

pergeseran waktu dimulainya pelayanan pada pelanggan j saat pelanggan u disisipkan antara pelanggan i dan j. Nilai $c_{11}(i,u,j)$ dan $c_{12}(i,u,j)$ dapat dihitung dengan cara

$c_{11}(i,u,j) = d_{iu} + d_{uj} - \mu d_{ij}, \mu \geq 0$

$c_{12}(i,u,j) = T_{j_u} - T_j$

dimana d_{iu} adalah jarak dari pelanggan i ke pelanggan u, T_j adalah waktu dimulainya pelayanan untuk pelanggan j, T_{j_u} adalah waktu dimulainya pelayanan untuk pelanggan j jika pelanggan u dalam rute dan d_{ij} adalah jarak dari pelanggan i ke j.

Cara ini bertujuan untuk mendapatkan keuntungan maksimum dengan menempatkan pelanggan yang akan disisipkan ke rute yang telah ada daripada menempatkannya ke rute baru. Contohnya saat $\mu = \alpha_1 = \lambda = 1$ dan $\alpha_2 = 0$, $c_2(i,u,j)$ adalah penghematan jarak tempuh ketika menempatkan pelanggan u di dalam rute yang ada, daripada menempatkan pelanggan u ke dalam rute baru. Tempat penyisipan terbaik untuk pelanggan yang belum masuk ke dalam rute adalah tempat penyisipan yang dapat meminimalkan jarak dan waktu, yaitu tempat penyisipan yang menghasilkan penambahan jarak dan waktu yang minimal.

2. $c_1(i,u,j)$ seperti yang didefinisikan sebelumnya

   $c_2(i,u,j) = \beta_1 R_d(u) + \beta_2 R_t(u), \beta_1 + \beta_2 = 1, \beta_1 \geq 0, \beta_2 > 0$

   dengan $R_d(u)$dan $R_t(u)$ adalah total jarak rute dan waktu pada rute yang setelah u masuk dalam rute.

   Cara ini bertujuan untuk memilih pelanggan yang biaya penyisipannya akan meminimalkan total jarak dan waktu.

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

   $c_2(i,u,j) = c_1(i,u,j)$

   Nilai $c_{11}(i,u,j)$ dan $c_{12}(i,u,j)$ seperti yang telah didefinisikan sebelumnya sedangkan $c_{13}(i,u,j)$ menunjukan interval waktu antara dimulainya pelayanan pada pelanggan u dan waktu terakhir kendaraan boleh memulai pelayanan. Nilai $c_{13}(i,u,j)$ dapat dihitung dengan cara :

   $c_{13}(i,u,j) = b_u - T_u$

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

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

Cara ini, selain menimbang aspek sebelumnya, juga menimbang aspek lain yaitu keadaan pelanggan yang mendesak untuk dilayani.

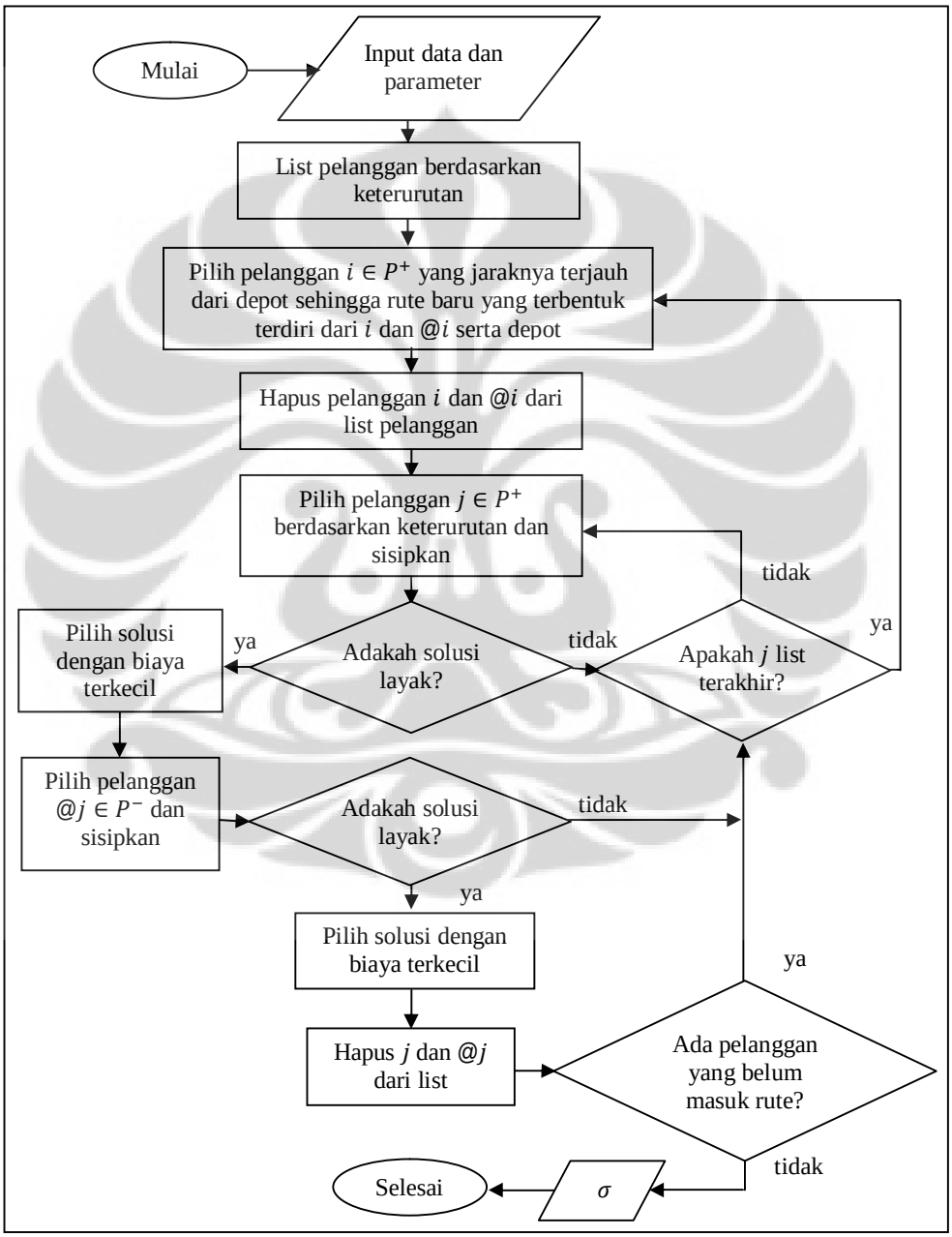
Dari ketiga cara di atas, dapat terlihat bahwa metode *insertion heuristic* memungkinkan pelanggan yang belum masuk rute untuk disisipkan diantara dua pelanggan lain yang memenuhi kendala daripada menempatkannya pada bagian belakang rute (Solomon, 1987).

Proses *insertion heuristic* pada *Pickup and Delivery Vehicle Routing Problem with Time Windows* (PDPTW) dimulai dengan memilih pelanggan pertama untuk masuk ke dalam rute. Pelanggan pertama yang dipilih berdasarkan jarak terjauh dari depot ke pelanggan jemput. Misalkan pelanggan i adalah pelanggan jemput yang memiliki jarak terjauh dari depot, maka rute pertama yang terbentuk adalah $0 - i - @i - 0$. Setelah itu akan disisipkan pelanggan yang belum masuk ke rute tersebut. Pelanggan dipilih berdasarkan keterurutan pelanggan jemput. Penyisipan dimulai dengan menyisipkan pelanggan jemput ke rute tersebut. Misalkan pelanggan j akan disisipkan maka kemungkinan rute yang akan terbentuk adalah $0 - j - i - @i - 0, 0 - i - j - @i - 0$ dan $0 - i - @i - j - 0$. Tiap kemungkinan rute tersebut akan diperiksa apakah memenuhi solusi layak atau tidak. Jika memenuhi solusi layak maka akan dicari biaya penyisipan. Jika terdapat lebih dari satu rute yang merupakan solusi layak maka akan dipilih rute dengan biaya penyisipan terkecil. Dalam tugas akhir ini, yang dipakai untuk menghitung biaya penyisipan adalah cara yang ketiga.

Misalkan rute layak yang terbentuk dengan biaya penyisipan terkecil adalah $0 - j - i - @i - 0$, selanjutnya akan disisipkan pelanggan $@j$ yang merupakan pasangan pelanggan j. Agar memenuhi kendala *precedence* dan *pairing* maka kemungkinan rute yang terbentuk saat menyisipkan pelanggan adalah $0 - j - @j - i - @i - 0, 0 - j - i - @j - @i - 0$ dan $0 - j - i - @i - @j - 0$. Dan dengan cara yang sama saat menyisipkan pelanggan j akan dihasilkan rute yang memenuhi kendala dengan penyisipan biaya terkecil. Jika dari semua kemungkinan rute yang terbentuk saat penyisipan pelanggan $@j$ tidak ada yang memenuhi semua kendala maka pelanggan j dan $@j$ tidak dapat masuk ke rute tersebut dan dicari pelanggan lain yang belum masuk ke dalam rute untuk

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

disisipkan. Kemudian jika tidak ada pelanggan yang dapat disisipkan ke rute tersebut maka akan dibentuk rute baru untuk pelanggan-pelanggan yang belum masuk ke dalam rute dengan cara yang sama seperti sebelumnya. Metode ini akan berhenti saat semua pelanggan telah masuk kedalam rute dan dihasilkan sekumpulan rute yang memenuhi kendala. *Flowchart* metode *insertion heuristic* dapat dilihat pada Gambar 3.1 berikut



Gambar 3.1 *Flowchart* Algoritma *Insertion Heuristic*

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

3.2.2 Implementasi dalam Masalah

Contoh Masalah 3.1

Misalkan akan dicari rute optimal dari sebuah masalah PDPTW yang terdiri dari sebuah depot serta 8 pelanggan yang ingin dilayani. Pelanggan tersebut terdiri dari 4 pelanggan jemput dan 4 pelanggan antar. Kendaraan yang tersedia memiliki kapasitas yang sama yaitu 50. Tabel 3.1 memberikan keterangan lengkap mengenai banyak muatan, *time windows*, waktu pelayanan dari masing-masing pelanggan serta keterangan apakah pelanggan i merupakan pelanggan jemput atau antar. Sedangkan Tabel 3.2 memberikan keterangan jarak dari depot ke pelanggan dan jarak antar pelanggan. Diasumsikan bahwa jarak antar pelanggan juga jarak dari depot ke pelanggan simetris. Dan Tabel 3.3 memberi keterangan pasangan pelanggan jemput dan antar berdasarkan keterurutan.

Tabel 3.1 Data Pelanggan

i	q_i	a_i	b_i	s_i	k	l
0	0	0	600	0	0	0
1	30	16	120	70	0	5
2	20	50	120	70	0	4
3	10	10	65	70	0	7
4	−20	210	250	70	2	0
5	−30	320	400	70	1	0
6	10	130	200	70	0	8
7	−10	170	270	70	3	0
8	−10	300	410	70	6	0

Keterangan tabel :

- i menotasikan pelanggan ke i dengan 0 adalah depot.
- q_i adalah banyaknya muatan yang harus di jemput dari pelanggan i atau di antar ke pelanggan i. Jika q_i positif artinya kendaraan menjemput muatan dari pelanggan i dan jika q_i negatif artinya kendaraan mengantar muatan ke pelanggan i.

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

- $[a_i, b_i]$ adalah *time window* pelanggan i.
- s_i adalah lama waktu pelayanan pada pelanggan i.
- k dan l memberikan keterangan apakah pelanggan i merupakan pelanggan jemput atau antar. Jika $k = 0$ maka pelanggan i adalah pelanggan jemput dan muatan yang dijemput harus diantar ke pelanggan l. Begitu juga sebaliknya.
- Untuk memudahkan penyelesaian masalah, digunakan variabel tambahan z_i yaitu waktu sampainya kendaraan di pelanggan i karena sampainya kendaraan di pelanggan i belum tentu sama dengan waktu pelayanan dimulai.

Tabel 3.2 Data Jarak Antar Pelanggan serta Jarak Pelanggan dan Depot (satuan jarak)

	0	1	2	3	4	5	6	7	8
0	0	16.55	10	15.13	15	27.59	26.92	33.54	40.36
1	16.55	0	15.30	31.32	17	13	18.68	23	25
2	10	15.30	0	19.21	5	21.93	18.03	25	34.48
3	15.13	31.32	19.21	0	22.67	40.82	36.80	43.86	53.53
4	15	17	5	22.67	0	20.40	14.14	21.21	32.31
5	27.59	13	21.93	40.82	20.40	0	11.66	12.08	12.81
6	26.92	18.68	18.03	36.80	14.14	11.66	0	7.07	20.10
7	33.54	23	25	43.86	21.21	12.08	7.07	0	15.30
8	40.36	25	34.48	53.53	32.31	12.81	20.10	15.30	0

Tabel 3.3 Urutan Pasangan Pelanggan Jemput dan Antar

Urutan	Pelanggan Jemput	Pelanggan Antar
1	1	5
2	2	4
3	3	7
4	6	8

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

Metode *insertion heuristic* dimulai dengan mencari pelanggan jemput yang memiliki jarak terjauh dari depot. Pelanggan tersebut kemudian dimasukkan ke dalam rute beserta pasangannya. Dari Tabel 3.3 diperoleh d_{01} = 16.55, d_{02} = 10, d_{03} = 15.13, d_{06} = 26.92. Pelanggan 6 memiliki jarak terjauh dari depot maka pelanggan pertama yang masuk dalam rute adalah pelanggan 6. Lalu pasangan pelanggan 6 adalah pelanggan 8, jadi rute yang terbentuk adalah 0 – 6 – 8 – 0. Diasumsikan kecepatan kendaraan adalah 1 dan konstan maka jarak antara dua simpul ekivalen dengan waktu tempuh sehingga t_{06} = d_{06} = 26.92. Pelayanan pada pelanggan 6 dimulai pada waktu T_6 = 130 karena a_6 = 130. Jadi meskipun z_6 = 26.92 , kendaraan harus menunggu hingga waktu T_6 untuk memulai pelayanan. Kendaraan akan meninggalkan pelanggan 6 menuju pelanggan 8 pada waktu T_6 + s_6 = 200. Dengan d_{68} = 20.10 maka z_8 = 220.10 dan memulai pelayanan pada waktu T_8 = 300 karena a_8 = 300. Setelah itu kendaraan akan meninggalkan pelanggan 8 dan menuju depot pada waktu T_8 + s_8 = 370. Dengan d_{80} = d_{08} = 40.36 maka T_0 = 410.36.

Untuk kendala kapasitas, kendaraan meninggalkan depot dengan keadaan tanpa muatan. Setelah mengunjungi pelanggan 6 maka muatan dalam kendaraan adalah 10 (Y_6 = 10), kemudian kendaraan menuju pelanggan 8 untuk mengantar muatan dari pelanggan 6 sehingga Y_8 = 0. Terlihat bahwa rute 0 – 6 – 8 – 0 memenuhi kendala kapasitas, *time windows, precedence* serta *pairing*.
Dari rute 0 – 6 – 8 – 0 didapatkan total jarak tempuh adalah d_{06} + d_{68} + d_{80} = 26.92 + 20.10 + 40.36 = 87.38.

Kemudian akan disisipkan pelanggan 1 dan 5. Hal ini berdasarkan urutan di Tabel 3.3. Pertama disisipkan pelanggan 1 pada rute 0 – 6 – 8 – 0 dan akan dicari biaya penyisipan minimum. Rute yang mungkin terbentuk dengan penyisipan pelanggan 1 adalah 0 – 1 – 6 – 8 – 0, 0 – 6 – 1 – 8 – 0, atau 0 – 6 – 8 – 1 – 0. Masing-masing rute tersebut harus memenuhi kendala kapasitas, dan *time windows.* Kendala *precedence* dan *pairing* untuk pasangan pelanggan 1 dan 5 akan dipenuhi saat pelanggan 5 masuk ke dalam rute.

Pertama akan diperiksa apakah rute 0 – 1 – 6 – 8 – 0, 0 – 6 – 1 – 8 – 0 atau 0 – 6 – 8 – 1 – 0 memenuhi kendala. Dengan cara yang sama seperti yang

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

dijelaskan sebelumnya saat memeriksa kendala pada rute $0-6-8-0$, didapat bahwa rute $0-1-6-8-0$ adalah rute yang memenuhi kendala sedangkan dua rute lainnya tidak. Kemudian akan dihitung biaya penyisipan pelanggan 1 diantara depot dan pelanggan 6.

Lalu digunakan cara ketiga untuk menghitung biaya penyisipan terkecil dengan parameter $\mu = 1, \alpha_1 = 0.5, \alpha_2 = 0.5, \alpha_3 = 0$ (Solomon, 1987). Penetapan parameter tersebut menggambarkan bahwa biaya dihitung berdasarkan pertambahan jarak dan waktu setelah menyisipkan seorang pelanggan diantara dua pelanggan lain.

Biaya penyisipan pelanggan 1 di antara depot dan pelanggan 6 adalah sebagai berikut :

$c_{11}(0,1,6) = d_{01} + d_{16} - \mu d_{06}$

$= 16.55 + 18.68 - 26.92$

$= 8.31$

$c_{12}(0,1,6) = T_{6_1} - T_6 = 0$

$c_{13}(0,1,6) = b_1 - T_1 = 120 - 16.66 = 103.45$

$c_1(0,1,6) = \alpha_1 c_{11}(0,1,6) + \alpha_2 c_{12}(0,1,6) + \alpha_3 c_{13}(0,1,6)$

$= 0,5(8.31) + 0.5(0) + 0$

$= 4.15$

$c_2(0,1,6) = 4.15$

Rute yang terbentuk sekarang adalah $0-1-6-8-0$. Dari rute ini akan disisipkan pelanggan 5 yang akan melengkapi kendala *precedence* dan *pairing*. Agar memenuhi kendala *precedence* dan *pairing* maka rute yang mungkin terbentuk dari disisipkannya pelanggan 5 adalah $0-1-5-6-8-0, 0-1-6-5-8-0$ atau $0-1-6-8-5-0$. Dari ketiga rute tersebut yang memenuhi kendala kapasitas dan *time windows* adalah rute $0-1-6-5-8-0$ dan $0-1-6-8-5-0$. Maka akan dihitung biaya penyisipan pelanggan 5 pada kedua rute tersebut.

1. Penyisipan pelanggan 5 di antara pelanggan 6 dan pelanggan 8

   $c_{11}(6,5,8) = d_{65} + d_{58} - \mu d_{68}$

   $= 11.66 + 12.81 - 20.10$

   $= 4.37$

   $c_{12}(6,5,8) = T_{8_5} - T_8$

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

$= 402.81 - 300$

$= 102.81$

$c_{13}(6,5,8) = b_5 - T_5$

$= 80$

$c_1(6,5,8) = \alpha_1\, c_{11}(6,5,8) + \alpha_2\, c_{12}(6,5,8) + \alpha_3\, c_{13}(6,5,8)$

$= 0.5(4.37) + 0.5(102.81) + 0$

$= 53.59$

$c_2(6,5,8) = 53.59$

2. Penyisipan 5 diantara pelanggan 8 dan depot

$c_{11}(8,5,0) = d_{85} + d_{50} - \mu d_{80}$

$= 12.81 + 27.59 - 40.36$

$= 0.04$

$c_{12}(8,5,0) = T_{0_5} - T_0$

$= 480.4 - 410.36$

$= 70.04$

$c_{13}(8,5,0) = b_5 - T_5$

$= 400 - 382.81$

$= 17,19$

$c_1(8,5,0) = \alpha_1\, c_{11}(8,5,0) + \alpha_2\, c_{12}(8,5,0) + \alpha_3\, c_{13}(8,5,0)$

$= 0.5(0.04) + 0.5(70.04) + 0$

$= 35.04$

$c_2(8,5,0) = 35.04$

Dari dua rute yang mungkin terlihat bahwa biaya penyisipan terkecil diperoleh dari rute $0-1-6-8-5-0$. Jadi rute baru yang terbentuk adalah $0-1-6-8-5-0$. Pada rute $0-1-6-8-5-0$ dengan total jarak tempuh adalah $d_{01} + d_{16} + d_{68} + d_{85} + d_{50} = 16.55 + 18.68 + 20.10 + 12.81 + 27.59 = 95.73$.

Selanjutnya pasangan pelanggan yang belum disisipkan adalah $2-4$ dan $3-7$. Penyisipan pelanggan $2-4$ pada rute $0-1-6-8-5-0$ tidak dapat dilakukan karena tidak memenuhi kendala. Begitu juga dengan kendala $3-7$ sehingga pelanggan $2-4$ dan $3-7$ harus dibuat rute baru.

Pemilihan pelanggan pertama dalam pembentukan rute baru dilakukan dengan cara yang sama yaitu memilih pelanggan yang jaraknya paling jauh dari

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

depot. Di antara pelanggan 2 dan 3 yang letaknya paling jauh adalah pelanggan 3 sehingga rute baru yang terbentuk adalah $0-3-7-0$. Pada rute $0-3-7-0$ dengan total jarak tempuh adalah $d_{03} + d_{37} + d_{07} = 15.13 + 43.86 + 33.54 = 92.53$.

Pelanggan yang belum masuk rute adalah $2-4$. Penyisipan pelanggan $2-4$ tidak dapat dilakukan karena tidak memenuhi kendala sehingga pelanggan $2-4$ dibuat rute baru yaitu $0-2-4-0$. Pada rute $0-2-4-0$ dengan total jarak tempuh adalah $d_{02} + d_{24} + d_{40} = 10 + 5 + 15 = 30$. Maka solusi awal dari Contoh Masalah 3.1 adalah

$\sigma = \{0-1-6-8-5-0, 0-3-7-0, 0-2-4-0\}$ dengan

Rute 1 = $0-1-6-8-5-0$,

Rute 2 = $0-3-7-0$

Rute 3 = $0-2-4-0$

Banyak rute = 3

Total jarak tempuh = $95.73 + 92.53 + 30 = 218.26$

3.3 *Simulated Annealing* dalam Mengurangi Banyaknya Rute

Pada subbab ini akan dijelaskan mengenai algoritma *simulated annealing*. Algoritma *simulated annealing* (SA) dimulai dengan pembentukan solusi awal yang telah dijelaskan pada Subbab 3.2 menggunakan algoritma *insertion heuristic*. Kemudian akan dibangun sebuah solusi lingkungan yang akan dijelaskan di Subbab 3.3.1, evaluasi fungsi yang akan dijelaskan pada Subbab 3.3.2, perhitungan selisih nilai fungsi evaluasi yaitu $\Delta\sigma$ pada Subbab 3.3.3 serta implementasinya dalam masalah pada Subbab 3.3.4.

3.3.1 Pembentukan Solusi Sub-lingkungan

Berdasarkan Bent dan Hentenryck (2003), solusi lingkungan dibangun dengan menggunakan *simple pair relocation*. Misalkan diberikan solusi awal σ, $\mathcal{N}(\sigma)$ menotasikan lingkungan dari σ, yaitu himpunan solusi layak yang didapatkan dengan menggunakan *pair relocation*. *Pair relocation* adalah pemindahan sepasang pelanggan dari suatu tempat ke tempat lain. *Pair relocation*

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

yang digunakan dalam tugas akhir ini adalah dengan memindahkan sepasang pelanggan dari satu rute ke rute lain. Hal ini dilakukan agar dapat tercapai pengurangan banyaknya rute.

Salah satu hal menarik dari algoritma SA adalah bagaimana algoritma ini mengeksplorasi lingkungan. Dalam tugas akhir ini eksplorasi terhadap lingkungan dilakukan dengan membangun sub-lingkungan secara acak. Sub-lingkungan dibangun dengan memilih secara acak pelanggan i dan $@i$ dari N yang merupakan himpunan pelanggan serta melakukan *pair relocation* terhadap pelanggan i dan $@i$. Sub-lingkungan ini dibangun untuk mencari solusi yang lebih baik dari solusi sebelumnya. Himpunan sub-lingkungan dinotasikan dengan $\mathcal{N}(i,\sigma)$ yaitu himpunan yang terdiri dari solusi lingkungan layak yang terbentuk setelah dilakukan *pair relocation* pelanggan i dan $@i$. Misalkan terdapat suatu solusi awal $\sigma = \{0-i-@i-j-@j-0, 0-k-@k-0\}$, lalu secara acak terpilih pelanggan i maka akan dilakukan *pair relocation* untuk pelanggan i dan $@i$. *Pair relocation* dilakukan dengan memindahkan i dan $@i$ ke rute lain. Sub-lingkungan yang mungkin terbentuk adalah

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

Dari enam kemungkinan di atas yang menjadi solusi sub-lingkungan adalah sub-lingkungan yang memenuhi kendala.

3.3.2 Fungsi Evaluasi

Salah satu hal yang penting dari algoritma SA adalah fungsi evaluasi. Fungsi evaluasi yang biasanya dipakai adalah fungsi yang bergantung pada banyaknya rute dan biaya perjalanan, namun fungsi evaluasi seperti ini hanya mengarah pada solusi dengan biaya yang kecil dan terkadang banyaknya rute yang tidak dapat dikurangi. Untuk mengatasi hal ini, fungsi evaluasi yang

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

digunakan dalam algoritma SA pada tugas akhir ini adalah fungsi evaluasi berdasarkan *lexicographic ordering*

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

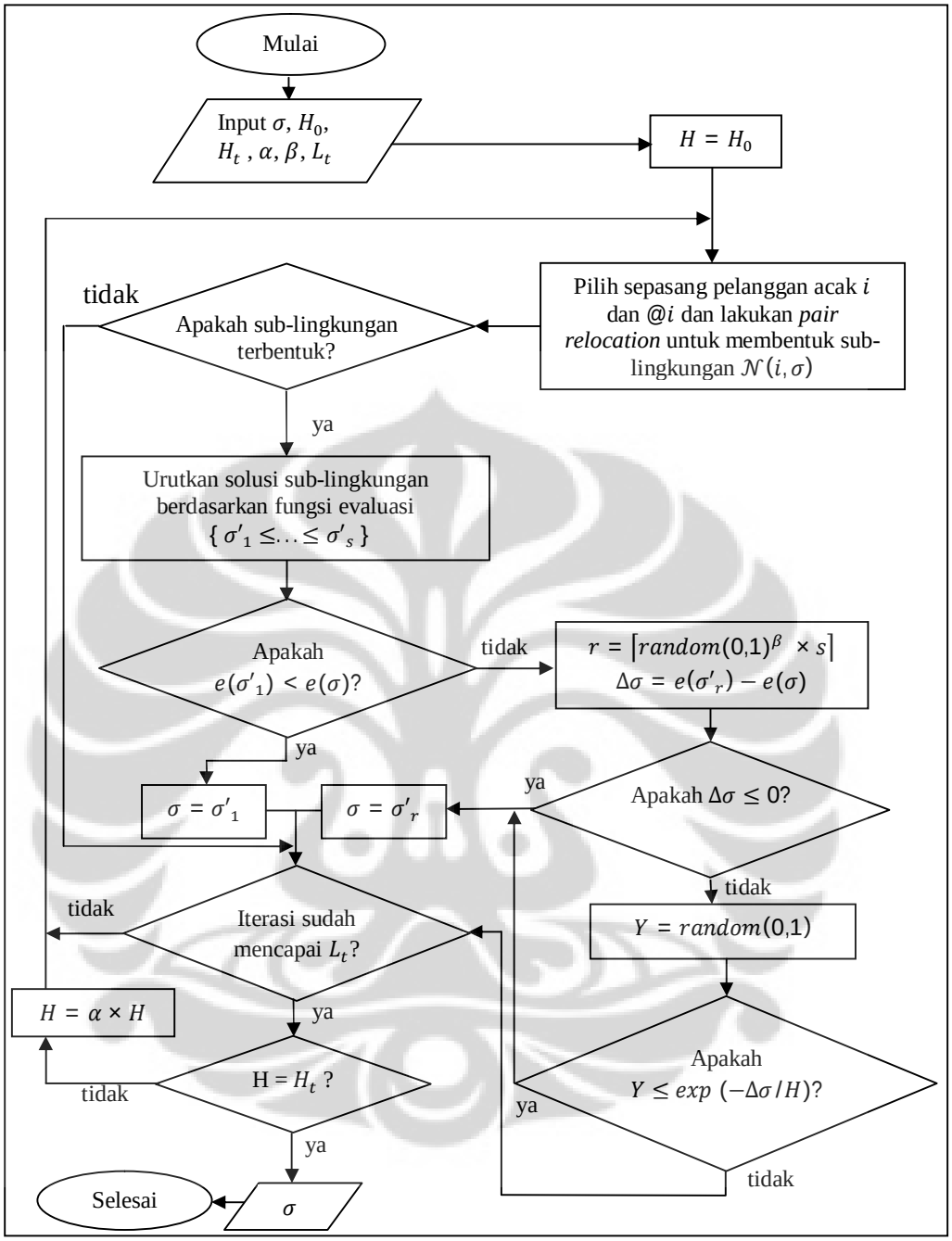
untuk meminimalkan banyaknya rute (Bent dan Hentenryck, 2003). Komponen pertama $|\sigma|$ merupakan banyaknya rute pada solusi σ. Komponen kedua, $|r|$ adalah banyaknya pelanggan di setiap rute dalam himpunan rute σ, komponen ini akan mengarahkan algoritma untuk memindahkan pelanggan dari rute yang kecil ke rute yang lebih besar. Sehingga dengan memaksimumkan nilai $\sum_{r\in\sigma} |r|^2$ diharapkan akan terpilih solusi yang terdiri dari rute dengan pelanggan yang lebih padat daripada solusi yang terdiri dari rute-rute dengan pelanggan yang jumlahnya berimbang. Misalkan terdapat dua solusi dengan banyaknya rute sama yaitu 2 buah rute. Namun pada solusi pertama σ_1, rute 1 dan 2 terdiri dari 4 pelanggan sehingga nilai $-\sum_{r\in\sigma_1} |r|^2$ adalah -32, sedangkan pada solusi yang kedua σ_2, rute 1 terdiri dari 6 pelanggan dan rute 2 terdiri dari 2 pelanggan sehingga nilai $-\sum_{r\in\sigma_2} |r|^2$ adalah -40, maka yang terpilih adalah σ_2. Sedangkan komponen terakhir adalah biaya perjalanan dari sekumpulan rute σ. Biaya perjalanan ini dilihat dari total jarak tempuh semua kendaraan yang melalui sekumpulan rute tersebut.

3.3.3 Perhitungan Selisih Nilai Fungsi Evaluasi

Selisih nilai fungsi evaluasi atau $\Delta\sigma$ akan dihitung jika solusi lingkungan tidak lebih baik dari solusi sebelumnya. Dengan fungsi evaluasi berdasarkan *lexicographic ordering* maka nilai $\Delta\sigma$ adalah selisih nilai pertama yang membedakan fungsi evaluasi antara dua solusi (Bent dan Hentenryck, 2003). Contohnya $e(\sigma_i) = \langle 2, -40, 190 \rangle$ dan $e(\sigma_j) = \langle 2, -36, 190 \rangle$ maka $\Delta\sigma = -36 - (-40) = 4$.

Berikut adalah *flowchart* algoritma SA yang digunakan dalam tugas akhir ini

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012



Keterangan : σ = solusi, H_0 = suhu awal, H_t = suhu minimum, α = *cooling rate*, β= parameter, L_t= iterasi tiap suhu, H = suhu sekarang, σ' = solusi sub-lingkungan, $e(\sigma)$ = fungsi evaluasi σ

Gambar 3.2 *Flowchart* Algoritma SA

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

3.3.4 Implementasi dalam Masalah

Dari Contoh Masalah 3.1, didapatkan solusi awal $\sigma = \{0-1-6-8-5-0, 0-3-7-0, 0-2-4-0\}$. Misalkan parameter yang ditentukan adalah $H_0 = 1$, $\alpha = 0.5$, iterasi = 2 dan $H_t = 0.5$.

Iterasi ke - 1

Akan dibentuk sub-lingkungan dari σ dengan memilih pelanggan secara acak. Misalkan pelanggan yang terpilih adalah pelanggan 1, maka akan dilakukan *pair relocation* untuk pelanggan 1 dan 5. Sub-lingkungan terbentuk dengan memindahkan pelanggan 1 dan 5 ke rute lain yang memenuhi kendala. Sub-lingkungan yang terbentuk adalah

$\sigma'_a = \{0-6-8-0, 0-3-1-7-5-0, 0-2-4-0\}$

$\sigma'_b = \{0-6-8-0, 0-3-7-0, 0-1-2-4-5-0\}$

Pada Contoh Masalah 3.1, sub-lingkungan yang terbentuk terdiri dari dua solusi, maka solusi lingkungan tersebut akan disusun sehingga $e(\sigma'_1) \leq e(\sigma'_2)$. Fungsi evaluasi yang digunakan adalah

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

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

   $|\sigma'_a| = 3$

   $-\sum_{r\in\sigma'_a} |r|^2 = -24$

   $\sum_{r\in\sigma'_a} t(r) = 87{,}38 + 109.12 + 30 = 226.5$

   $e(\sigma'_a) = \langle 3, -24, 226.5 \rangle$

2. Untuk σ_b, nilai $e(\sigma'_b)$ adalah

   $|\sigma'_b| = 3$

   $-\sum_{r\in\sigma'_b} |r|^2 = -(4 + 4 + 16) = -24$

   $\sum_{r\in\sigma'_b} t(r) = 87{,}38 + 92{,}53 + 84{,}84 = 264{,}75$

   $e(\sigma'_b) = \langle 3, -24, 264{,}75 \rangle$

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

Maka susunan solusi lingkungan yang terbentuk adalah $\langle \sigma'_1, \sigma'_2 \rangle$ dengan $\sigma'_1 = \sigma'_a$ dan $\sigma'_2 = \sigma'_b$. Kemudian akan dibandingkan $e(\sigma)$ dan $e(\sigma'_1)$. Langkah ini menjadi sebuah langkah yang merupakan modifikasi dari algoritma SA yang dilakukan oleh Bent dan Hentenryck, 2003. Biasanya, pemilihan solusi lingkungan dari algoritma SA dilakukan dengan cara acak.

- Untuk $e(\sigma)$

  $|\sigma| = 3$

  $-\sum_{r\in\sigma} |r|^2 = -(4 + 4 + 16) = -24$

  $\sum_{r\in\sigma} t(r) = 95.73 + 92.53 + 30 = 218.26$

  $e(\sigma) = \langle 3, -24, 218.26 \rangle$
- Untuk $e(\sigma'_1)$

  $e(\sigma'_1) = \langle 3, -24, 226.5 \rangle$

Berdasarkan *lexicographic ordering*, $\langle 3, -24, 218.26 \rangle < \langle 3, -24, 226.5 \rangle$ atau $e(\sigma'_1) > e(\sigma)$, maka nilai $\Delta\sigma > 0$. Kemudian akan dicari sebuah bilangan random r dengan $r = \lceil random(0,1)^{\beta} \times s \rceil$, β adalah parameter yang ditentukan dan s adalah banyaknya solusi dalam sub-lingkungan lalu akan dibandingkan $e(\sigma)$ dan $e(\sigma'_r)$. Penggunaan parameter β ini dimaksudkan agar solusi lingkungan yang terpilih adalah solusi lingkungan dengan nilai fungsi evaluasi yang relatif kecil, nilai β ditentukan sebesar 10 (Bent dan Hentenryck, 2003). Misalkan nilai $r = 1$, maka akan dibandingkan $e(\sigma)$ dan $e(\sigma'_1)$. Sama seperti yang sebelumnya didapat bahwa $e(\sigma) < e(\sigma'_1)$ atau $\Delta\sigma > 0$.

Selanjutnya akan diselidiki apakah σ'_1 dapat diterima dengan probabilitas $exp(-\Delta\sigma/H)$. Dicari sebuah bilangan P, bilangan acak diantara 0 dan 1, misalkan yang didapatkan bilangan adalah 0.00023. Dengan $e(\sigma) = \langle 3, -24, (218.26) \rangle$ dan $e(\sigma'_1) = \langle 3, -24, 226.5 \rangle$ maka $\Delta\sigma = 226{,}5 - 218{,}26 = 8.24$.

Jika bilangan $P \leq exp(-\Delta\sigma/1)$ maka σ_1 diterima sebagai solusi baru. Sebaliknya jika bilangan $P > exp(-\Delta\sigma/1)$ maka solusi σ tidak berubah.

– Untuk $\Delta\sigma = 8.24$

$exp(-\Delta\sigma/1) = 0.000264$ maka $0.00023 \leq exp(-\Delta\sigma/1)$.

Dari keterangan di atas maka σ'_1 diterima sebagai solusi baru. Sehingga solusi sekarang menjadi $\sigma = \{0-6-8-0, 0-3-1-7-5-\ 0, 0-4-5-0\}$ dengan

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

Rute 1 = 0 – 6 – 8 – 0

Rute 2 = 0 – 3 – 1 – 7 – 5 – 0

Rute 3 = 0 – 2 – 4 – 0

Banyak rute = 3

Total jarak = 226.5.

Iterasi ke – 2

Solusi sekarang adalah $\sigma = \{0-6-8-0, 0-3-1-7-5-0, 0-2-4-0\}$. Misalkan pasangan yang terpilih adalah 2 dan 4. Akan dibuat solusi sub-lingkungan dengan memindahkan 2 – 4 ke rute lain dan solusi sub-lingkungan yang terbentuk adalah

$$\sigma'_a = \{0-2-6-4-8-0, 0-3-1-7-5-0,\}$$

Kemudian akan dihitung nilai fungsi evaluasinya

- $e(\sigma) = \langle 3, -24, 226.5 \rangle$
- $e(\sigma'_a) = \langle 2, -32, 223.96 \rangle$

Dari nilai fungsi evaluasi terlihat bahwa $e(\sigma'_a) < e(\sigma)$ maka σ'_a diterima sebagai solusi. Maka solusi yang didapat dari algoritma *simulated annealing* adalah $\sigma = \{0-2-6-4-8-0, 0-3-1-7-5-0\}$ dengan

Rute 1 = 0 – 2 – 6 – 4 – 8 – 0

Rute 2 = 0 – 3 – 1 – 7 – 5 – 0

Banyak rute = 2

Total jarak = 223.96.

3.4 *Large Neighborhood Search* dalam Mengurangi Total Jarak

Algoritma *large neighborhood search* (LNS) merupakan algoritma yang digunakan pada tahap kedua. Algoritma ini digunakan untuk meminimumkan biaya perjalanan. Solusi awal yang digunakan dalam algoritma LNS adalah solusi akhir dari algoritma SA. Algoritma LNS dimulai dengan menentukan banyaknya pasangan pelanggan yang akan dihapus dari rute. Setelah itu akan dibentuk solusi sub-lingkungan dengan menggunakan metode penghapusan dan perbaikan yang akan dijelaskan pada Subbab 3.4.1, kemudian fungsi evaluasi yang digunakan

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

akan dijelaskan pada Subbab 3.4.2 serta implementasi dalam masalah pada Subbab 3.4.3.

3.4.1 Pembentukan Solusi Sub-lingkungan

Solusi sub-lingkungan dibentuk dengan cara metode penghapusan dan perbaikan terhadap solusi yang ada. Berikut akan dijelaskan metode penghapusan dan perbaikan yang digunakan pada tugas akhir ini :

1. Metode Penghapusan

   Metode penghapusan yang digunakan pada tugas akhir ini adalah dengan memilih pelanggan yang akan dihilangkan dari rute secara acak. Misalkan sebanyak x pasang pelanggan akan dihapus dari himpunan rute yang menjadi solusi awal. Maka akan ditentukan sebuah bilangan acak integer dari 1 hingga banyaknya pelanggan jemput. Bilangan ini akan disesuaikan dengan indeks pada tabel urutan pasangan pelanggan seperti pada Tabel 3.3. Setelah itu akan dilakukan penghapusan pasangan pelanggan yang terpilih dari rute. Langkah yang sama diulangi hingga sebanyak x pasang pelanggan terhapus dari rute. Langkah selanjutnya adalah pembuatan solusi lingkungan dengan metode perbaikan.

2. Metode Perbaikan

   Metode perbaikan dilakukan untuk membentuk kembali solusi yang sebelumnya telah berubah oleh metode penghapusan. Metode perbaikan yang digunakan pada tugas akhir ini adalah dengan menyisipkan kembali x pasang pelanggan satu persatu pasang dengan urutan acak. Penyisipan kembali pasangan-pasangan pelanggan yang telah dihapus sebelumnya akan menghasilkan solusi - solusi sub-lingkungan yang layak.

Solusi sub-lingkungan dinotasikan dengan $N(\sigma, x)$ yaitu solusi sub-lingkungan yang terbentuk dengan menyisipkan kembali x pasang pelanggan yang telah dihapus dari rute. Masing-masing solusi sub-lingkungan akan dihitung fungsi evaluasinya dan akan dipilih solusi sub-lingkungan dengan nilai fungsi evaluasi terkecil untuk dibandingkan dengan solusi sebelumnya.

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

3.4.2 Fungsi Evaluasi

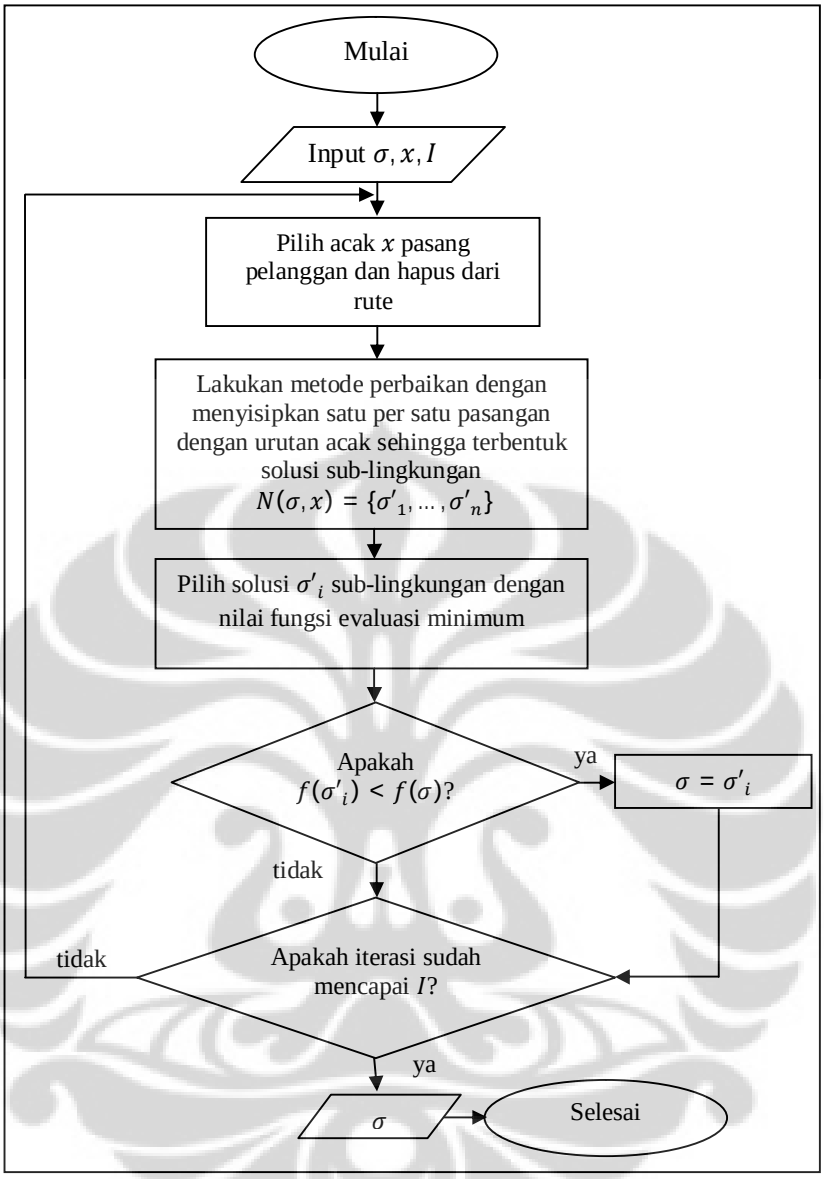
Pada tugas akhir ini, fungsi evaluasi yang digunakan pada algoritma LNS adalah fungsi evaluasi berdasarkan *lexicographic ordering*

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

dengan tujuan utama untuk meminimalkan biaya perjalanan (Bent dan Hentenryck, 2003). Penggunaan $|\sigma|$ pada fungsi evaluasi dikarenakan pada beberapa kasus, meminimumkan biaya perjalanan dapat menyebabkan pengurangan banyaknya rute. Fungsi evaluasi ini dipakai untuk menentukan apakah solusi sub-lingkungan yang terpilih dapat menggantikan solusi sebelumnya atau tidak. Solusi sub-lingkungan yang terpilih dapat diterima sebagai solusi baru bila solusi tersebut lebih baik dari solusi sebelumnya berdasarkan *lexicographic ordering*. Algoritma LNS akan berhenti pada iterasi yang ditentukan.

Berikut adalah *flowchart* algoritma LNS yang digunakan pada tugas akhir ini

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012



Keterangan : σ = solusi, x = banyak pasang pelanggan, $N(\sigma, x)$ = sub-lingkungan yang terbentuk dengan melakukan metode penghapusan dan menyisipkan kembali x pasangan pelanggan pada σ, I = banyaknya iterasi, σ'= solusi sub-lingkungan, $f(\sigma)$ = nilai fungsi evaluasi σ

Gambar 3.3 *Flowchart* Algoritma LNS

3.4.3 Implementasi dalam Masalah

Melanjutkan Contoh Masalah 3.1, solusi yang didapat dari algoritma SA adalah $\sigma = \{0-2-6-4-8-0, 0-3-1-7-5-0\}$. Solusi ini akan menjadi solusi awal pada algoritma LNS. Pertama tentukan parameter x yaitu banyaknya

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

pasangan pelanggan yang akan dikenakan metode penghapusan dan perbaikan serta banyaknya iterasi. Misalkan $x = 2$, maka akan dipilih secara acak 2 pasang pelanggan yang akan dihapus solusi. Dari pemilihan secara acak, misalkan 2 pasang pelanggan yang terpilih adalah $1 - 5$ dan $6 - 8$, maka 2 pasangan pelanggan tersebut akan dihapus dari solusi σ sehinggan solusi σ sekarang menjadi $\sigma = \{0 - 2 - 4 - 0,0 - 3 - 7 - 0\}$.

Selanjutnya akan dipilih secara acak antara pasangan $1 - 5$ dan $6 - 8$ untuk disisipkan kembali pada $\sigma = \{0 - 2 - 4 - 0,0 - 3 - 7 - 0\}$. Misalkan yang terpilih pertama adalah $6 - 8$. Maka kemungkinan yang terbentuk dengan menyisipkan $6 - 8$ adalah $\{0 - 2 - 6 - 4 - 8 - 0,0 - 3 - 7 - 0\}$ dan $\{0 - 2 - 4 - 0,0 - 3 - 6 - 7 - 8 - 0\}$. Lalu akan dilakukan penyisipan $1 - 5$ pada kedua kemungkinan tersebut. Dan solusi sub-lingkungan yang terbentuk adalah $\sigma'_1 = \{0 - 2 - 6 - 4 - 8 - 0,0 - 3 - 1 - 7 - 5 - 0\}$ dan $\sigma'_2 = \{0 - 1 - 2 - 4 - 5 - 0,0 - 3 - 6 - 7 - 8 - 0\}$. Akan dihitung nilai fungsi evaluasi untuk σ'_1 dan σ'_2.

- Untuk σ'_1

  $|\sigma'_1| = 2$

  $\sum_{r \in \sigma'_1} t(r) = 114,84 + 109.12 = 223.96$

  $f(\sigma'_1) = \langle 2,\ 223.96 \rangle$

- Untuk σ'_2

  $|\sigma'_2| = 2$

  $\sum_{r \in \sigma'_2} t(r) = 114.66 + 84.84 = 199.5$

  $f(\sigma'_2) = \langle 2,\ 199.5 \rangle$

Maka solusi sub-lingkungan yang terpilih adalah σ'_2. Lalu σ'_2 akan dibandingkan dengan σ. Karena $\sigma = \sigma'_1$ maka nilai $f(\sigma) = \langle 2, 223.96 \rangle$. Berdasarkan *lexicographic ordering* $f(\sigma'_2) < f(\sigma)$ sehingga σ'_2 diterima sebagai solusi baru dan solusi yang baru adalah $\sigma = \{0 - 3 - 6 - 7 - 8 - 0,0 - 1 - 2 - 4 - 5 - 0\}$ dengan

Rute 1 = $0 - 3 - 6 - 7 - 8 - 0$

Rute 2 = $0 - 1 - 2 - 4 - 5 - 0$

Banyak rute = 2

Total Jarak = 199.5

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

Terlihat dengan 1 kali iterasi, total jarak tempuh berkurang dari 223.96 menjadi 199.5. Selanjutnya ulangi langkah yang sama dimulai dari pemilihan sebanyak x pasang pelanggan untuk dihapus dari rute hingga iterasi yang ditentukan.

Dari pembahasan di atas dengan Contoh Masalah 3.1, dapat terlihat bahwa algoritma hibrida dua tahap dapat digunakan untuk menyelesaikan masalah PDPTW. Penggunaan algoritma hibrida dua tahap pada Contoh Masalah 3.1 menghasilkan solusi dengan banyaknya rute sebanyak 2 dan total jarak 199.5. Metode *insertion heuristic* membentuk solusi awal yang terdiri dari 3 rute dengan total jarak 218.26, kemudian tahap algoritma SA mengurangi banyaknya rute dengan 2 kali iterasi dan menghasilkan solusi yang terdiri dari 2 rute dengan total jarak 223.96. Kemudian tahap algoritma LNS mengurangi total jarak dengan 1 kali iterasi dan menghasilkan solusi yang terdiri dari 2 rute dengan total jarak 199.5.

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

BAB 4
IMPLEMENTASI DAN HASIL

Pada bab ini akan dijabarkan hasil implementasi algoritma hibrida dua tahap untuk menyelesaikan *pickup and delivery vehicle routing problem with time windows* pada data *benchmark* menggunakan program Matlab 7.8 (R2009a) serta penjabaran hasil yang didapat. Data yang digunakan adalah data *benchmark* Li dan Lim dengan 100 dan 200 pelanggan (Li & Lim *benchmark*). Pada Subbab 4.1 akan ditentukan parameter yang digunakan dan Subbab 4.2 akan diperlihatkan solusi yang didapat dengan menjalankan algoritma hibrida dua tahap pada data *benchmark* 100 pelanggan tipe LR101. Dan pada Subbab 4.3 akan berikan hasil implementasi algoritma hibrida dua tahap pada data *benchmark* Li dan Lim lainnya.

4.1 Penetapan Parameter

Berikut adalah parameter – parameter yang ada pada setiap metode yang digunakan :

1. Metode *Insertion Heuristic*

   Pada tahap *insertion heuristic* digunakan variabel μ, α_1, α_2 dan α_3 untuk menghitung biaya penyisipan. Nilai yang digunakan pada tugas akhir ini mengacu pada Solomon (1987), nilai – nilai tersebut adalah sebagai berikut :

   - μ = 1
   - α_1 = 0.5
   - α_2 = 0.5
   - α_3 = 0

2. Tahap *Simulated Annealing*

   Pada tahap *simulated annealing* digunakan variabel suhu awal H_0, suhu akhir H_t, iterasi tiap suhu L_t, *cooling rate* α dan parameter β. Nilai yang digunakan pada tugas akhir ini mengacu pada Bent dan Hentenryck (2003), nilai – nilai tersebut adalah sebagai berikut :

   - H_0 = 2000
   - H_t = 0.01

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

- L_t = 2500
- α = 0.95
- β = 10

3. Tahap *Large Neighborhood Search* (LNS)

   Pada tahap ini digunakan dua variabel yaitu x dan I. Variabel x adalah jumlah pasangan pelanggan yang akan dihapus dan disisipkan kembali pada rute sedangkan I adalah banyaknya iterasi yang ditetapkan untuk melakukan tahap LNS. Nilai – nilai yang digunakan pada tugas akhir ini adalah

   - x = 3
   - I = 500

4.2 Spesifikasi Perangkat Lunak

Percobaan dilakukan pada program Matlab 7.8(R2009a) dengan spesifikasi perangkat lunak sebagai berikut :

Sistem operasi : Windows 7 *Professional*

Processor : Intel(R) Pentium(R) Dual CPU T2390 @1.86GHz 1.87 GHz

RAM : 2.00 GB (1.87 Usable)

System Type : 32-bit *Operating System*

4.3 Hasil Percobaan pada Data LR101

Penggunaan algoritma hibrida dua tahap pada masalah *benchmark* tipe LR101 dimulai dengan pembuatan solusi awal dengan menggunakan metode *insertion heuristic*. Solusi awal yang didapat solusi dengan rute sebanyak **25** dan total jarak **2151.6** dengan rincian rute sebagai berikut :

Rute 1 : 0 – 65 – 81 – 68 – 103 – 24 – 80 – 0

Rute 2 : 0 – 63 – 64 – 49 – 48 – 0

Rute 3 : 0 – 36 – 47 – 19 – 8 – 46 –17 – 0

Rute 4 : 0 – 2 – 73 – 35 – 77 – 0

Rute 5 : 0 – 27 – 79 – 66 – 1 – 0

Rute 6 : 0 – 71 – 50 – 0

Rute 7 : 0 – 39 – 23 – 104 – 67 – 55 – 25 – 0

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

Rute 8 : 0 – 5 – 44 – 105 – 85 – 43 – 13 – 0

Rute 9 : 0 – 62 – 11 – 90 – 20 – 32 – 70 – 0

Rute 10 : 0 – 14 – 38 – 74 – 102 – 0

Rute 11 : 0 – 92 – 42 – 15 – 87 – 57 – 97 – 0

Rute 12 : 0 – 33 – 29 – 78 – 34 – 0

Rute 13 : 0 – 72 – 21 – 41 – 56 – 4 – 58 – 0

Rute 14 : 0 – 45 – 83 – 16 – 84 – 37 – 91 – 0

Rute 15 : 0 – 59 – 95 – 75 – 22 – 96 – 100 – 0

Rute 16 : 0 – 28 – 12 – 51 – 101 – 0

Rute 17 : 0 – 31 – 30 – 9 – 10 – 0

Rute 18 : 0 – 98 – 61 – 86 – 93 – 0

Rute 19 : 0 – 82 – 18 – 60 – 89 – 0

Rute 20 : 0 – 88 – 7 – 0

Rute 21 : 0 – 99 – 94 – 0

Rute 22 : 0 – 76 – 53 – 106 – 54 – 0

Rute 23 : 0 – 69 – 3 – 0

Rute 24 : 0 – 52 – 6 – 0

Rute 25 : 0 – 40 – 26 – 0

Selanjutnya solusi dari *insertion heuristic* tersebut akan digunakan sebagai solusi awal pada tahap *simulated annealing* dengan menggunakan parameter yang telah ditetapkan pada Subbab 4.1. Solusi yang didapat adalah solusi dengan rute sebanyak **19** dan total jarak **1719** dengan rincian rute sebagai berikut :

Rute 1 : 0 – 63 – 64 – 49 – 48 – 0

Rute 2 : 0 – 36 – 47 – 19 – 8 – 46 – 17 – 0

Rute 3 : 0 – 92 – 42 – 15 – 87 – 57 – 97 – 0

Rute 4 : 0 – 65 – 71 – 81 – 50 – 68 – 103 – 0

Rute 5 : 0 – 39 – 23 – 104 – 67 – 56 – 4 – 0

Rute 6 : 0 – 62 – 11 – 90 – 20 – 32 – 70 – 0

Rute 7 : 0 – 14 – 44 –105 – 38 – 43 – 13 – 0

Rute 8 : 0 – 52 – 6 – 0

Rute 9 : 0 – 33 – 29 – 78 – 34 – 35 – 77 – 0

Rute 10 : 0 – 2 – 21 – 73 – 41 – 74 – 102 – 0

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

Rute 11 : 0 – 45 – 82 – 18 – 84 – 60 – 89 – 0

Rute 12 : 0 – 75 – 22 – 55 – 25 – 0

Rute 13 : 0 – 27 – 69 – 76 – 79 – 3 – 54 – 24 – 80 – 0

Rute 14 : 0 – 5 – 83 – 61 – 85 – 37 – 93 – 0

Rute 15 : 0 – 72 – 99 – 94 – 58 – 0

Rute 16 : 0 – 59 – 95 – 98 – 16 – 86 – 96 – 91– 100 – 0

Rute 17 : 0 – 30 – 51 – 101 – 9 – 66 – 1 – 0

Rute 18 : 0 – 31 – 88 – 7 – 10 – 0

Rute 19 : 0 – 28 – 12 – 40 – 53 – 106 – 26 – 0

Terlihat bahwa penggunaan algoritma *simulated annealing* berhasil menurunkan banyaknya rute dari 25 rute menjadi 19 rute serta total jarak dari 2151.6 menjadi 1719.

Selanjutnya solusi yang didapat dari algoritma *simulated annealing* akan digunakan sebagai solusi awal pada tahap selanjutnya yaitu algoritma *large neighborhood search* dengan iterasi sebanyak 500. Solusi yang didapat adalah solusi dengan rute sebanyak **19** dan total jarak **1650.8** dengan rincian rute sebagai berikut :

Rute 1 : 0 – 63 – 64 – 49 – 48 – 0

Rute 2 : 0 – 36 – 47 – 19 – 8 – 46 – 17 – 0

Rute 3 : 0 – 92 – 42 – 15 – 87 – 57 – 97 – 0

Rute 4 : 0 – 65 – 71 – 81 – 50 – 68 – 103 – 0

Rute 5 : 0 – 39 – 23 – 104 – 67 – 55 – 25 – 0

Rute 6 : 0 – 62 – 11 – 90 – 20 – 32 – 70 – 0

Rute 7 : 0 – 14 – 44 – 105 – 38 – 43 – 13 – 0

Rute 8 : 0 – 52 – 6 – 0

Rute 9 : 0 – 33 – 29 – 78 – 34 – 35 – 77 – 0

Rute 10 : 0 – 2 – 21 – 73 – 41 – 56 – 4 – 0

Rute 11 : 0 – 45 – 82 – 18 – 84 – 60 – 89 – 0

Rute 12 : 0 – 72 – 75 – 22 – 74 – 102 – 58 – 0

Rute 13 : 0 – 27 – 69 – 76 – 79 – 3 – 54 – 24 – 80 – 0

Rute 14 : 0 – 5 – 83 – 61 – 85 – 37 – 93 – 0

Rute 15 : 0 – 59 – 99 – 94 – 96 – 0

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

Rute 16 : 0 – 95 – 98 – 16 – 86 – 91 – 100 – 0

Rute 17 : 0 – 30 – 51 – 101 – 9 – 66 – 1 – 0

Rute 18 : 0 – 31 – 88 – 7 – 10 – 0

Rute 19 : 0 – 28 – 12 – 40 – 53 –106 – 26 – 0

Solusi tersebut sama dengan *best known solution* yang telah ada (Li & Lim *benchmark*). Dengan demikian penggunaan algoritma dua tahap dalam menyelesaikan *pickup and delivery vehcle routing problem with time windows* terbukti menghasilkan solusi terbaik untuk masalah *benchmark* tipe LR101.

4.4 Hasil Implementasi Data Lainnya

Pada subbab ini akan diberikan hasil implementasi algoritma hibrida dua tahap pada data *benchmark* Li dan Lim lainnya. Berikut akan disajikan hasil penggunaan algoritma hibrida dua tahap pada beberapa data *benchmark* Li dan Lim 100 pelanggan.

Tabel 4.1 Hasil Penggunaan Algoritma Hibrida Dua Tahap pada Data *Benchmark* 100 Pelanggan

Masalah	Solusi awal (*insertion heuristic*)		Tahap SA		Tahap LNS		*Best Known*	
	|V|	TD	|V|	TD	|V|	TD	|V|	TD
LC101	17	2714.431	11	1081.718	10	828.937	10	828.94
LC102	14	2975,602	10	975.502	10	828.937	10	828.94
LC105	15	2911.866	10	922.064	10	828.937	10	828.94
LC106	14	2903.280	10	920.122	10	828.937	10	828.94
LC107	13	2809.078	11	1475.643	10	828.937	10	828.94
LC108	13	3004.662	11	1555.564	10	826.4	10	826.44
LC201	4	1743.249	3	595.1165	3	591.5566	3	591.56
LR201	6	2845.558	4	1903.50	4	1592.406	4	1253.23
LRC101	21	2480.552	14	1774.275	14	1708.801	14	1708.8

Tabel di atas memperlihatkan solusi yang dihasilkan dari penggunaan algoritma hibrida dua tahap untuk masalah *pickup and delivery vehicle routing problem with time windows* setiap tahapnya untuk beberapa masalah *benchmark* 100 pelanggan lainnya dengan |V| adalah banyaknya kendaraan dan TD adalah

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

total jarak tempuh. Hasil di atas didapat dengan melakukan percobaan sebanyak 1 kali. Terlihat bahwa *simulated annealing* (SA) berhasil meminimalkan banyaknya rute dari solusi awal yang didapat dengan metode *insertion heuristic* sedangkan algoritma *large neighborhood search* (LNS) pada tahap dua berhasil mengurangi total jarak tempuh secara signifikan sehingga pengabungan dua algoritma tersebut menghasilkan solusi optimal. Dengan menggunakan algoritma hibrida dua tahap terlihat bahwa 8 dari 9 masalah di atas memberikan solusi yang sama dengan *best known solution* yang telah ada (Li & Lim *benchmark*).

Berikut akan disajikan hasil penggunaan algoritma hibrida dua tahap pada beberapa data *benchmark* Li dan Lim 200 pelanggan.

Tabel 4.2 Hasil Penggunaan Algoritma Hibrida Dua Tahap pada Data *Benchmark* 200 Pelanggan

Masalah	Solusi awal (*insertion heuristic*)		Tahap SA		Tahap LNS		*Best Known*	
	|V|	TD	|V|	TD	|V|	TD	|V|	TD
LC1_2_1	35	8714.475	20	2777.797	20	2704.568	20	2704.568
LC2_2_1	11	1171.259	7	2929.589	7	1982.544	6	1931.44
LR1_2_1	32	9234.821	20	5546.789	20	5447.486	20	4819.12
LR2_2_1	9	10342.12	5	6182.286	5	5595.019	5	4073.10
LRC1_2_1	28	7127.424	19	4829.61	19	3825.133	19	3606.06

Dari tabel di atas terlihat bahwa algoritma SA memberikan hasil yang baik dalam mengurangi jumlah rute dan algoritma LNS juga efektif dalam mengurangi total jarak. Hasil di atas didapat dengan menjalankan percobaan sebanyak 1 kali. Dari 5 tipe masalah yang diambil, 1 diantaranya berhasil memberikan solusi optimal yang sama dengan *best known solution* sedangkan 4 lainnya belum memberikan solusi yang sama dengan *best known solution*. Hal ini terjadi karena percobaan hanya dilakukan 1 kali sedangkan pada umumnya solusi optimal pada metode *heuristic* mungkin didapatkan dengan melakukan percobaan beberapa kali.

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Tugas akhir ini membahas mengenai penggunaan algoritma hibrida dua tahap untuk menyelesaikan *pickup and delivery vehicle routing problem with time windows* (PDPTW). Algoritma yang digunakan pada tahap pertama adalah *simulated annealing* (SA) dan pada tahap kedua adalah *large neighborhood search* (LNS). Algoritma SA digunakan dengan tujuan meminimumkan banyaknya rute dan algoritma LNS digunakan dengan tujuan meminimumkan biaya perjalanan yang ekivalen dengan total jarak tempuh. Penentuan solusi awal pada algoritma SA didapatkan dengan metode *insertion heuristic*, lalu solusi dari algoritma SA digunakan sebagai solusi awal pada algoritma LNS.

Dari hasil yang didapat pada Bab 4, terlihat bahwa algoritma hibrida dua tahap berhasil menyelesaikan masalah *pickup and delivery vehicle routing problem with time windows* serta memberikan solusi yang optimal pada beberapa masalah *benchmark* 100 dan 200 pelanggan. Metode *insertion heuristic* dalam penentuan solusi awal memberikan hasil yang sudah cukup baik. Selanjutnya, algoritma SA pada tahap pertama berhasil mengurangi banyaknya rute dengan sangat baik dan LNS pada tahap kedua terbukti efektif dalam mengurangi total jarak tempuh. Dengan demikian, penggunaan algoritma hibrida dua tahap ini yaitu menggabungkan algoritma SA dan LNS dapat memberikan solusi yang optimal dalam penyelesaian *pickup and delivery vehicle routing problem with time windows*.

5.2 Saran

Saran yang dapat diberikan untuk pengembangan tugas akhir ini antara lain :

- Pada tugas akhir ini hanya menggunakan parameter yang tetap untuk setiap algoritma yang digunakan sehingga belum dapat terlihat pengaruh tiap parameter terhadap hasil yang didapatkan. Penulis berharap pengaruh masing-

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

masing parameter pada setiap iterasi dapat dibahas pada pengembangan selanjutnya.

- Tugas akhir ini hanya mengambil beberapa masalah *benchmark* Li dan Lim dengan 100 dan 200 pelanggan karena keterbatasan waktu, untuk pengembangan selanjutnya, algoritma yang digunakan dalam tugas akhir ini dapat diimplementasikan pada data *benchmark* lain dengan pelanggan yang lebih besar.

- Penyelesaian *pickup and delivery vehicle routing problem with time windows* dapat dicapai tidak hanya dengan algoritma yang digunakan pada tugas akhir ini saja, sehingga penggunaan algoritma lain dalam menyelesaikan PDPTW masih dapat dilakukan.

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

DAFTAR PUSTAKA

Bent, R., Hentenryck, P. V. (2003). A Two-Stage Hybrid Algorithm for Pickup and Delivery Vehicle Routing Problem with Time Window. 123-137.

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

Campbell, A. M., & Savelsbergh, M. (2004). Efficient Insertion Heuristics for Vehicle Routing and Scheduling Problems. *Transportation Science*, 38(3), 369-378.

Desaulniers, G., dkk. (2002). VRP with Pickup and Delivery. *SIAM Monographs and Discrete Mathematics and Applications*, 225 - 242.

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, Communications & Control VI* , 246-257.

Dumas, Y., Desrosiers, J., Soumis, F. (1991). The Pickup and Delivery Problem with Time Windows. *European Journal of Operational Research*, 54, 7-22.

Joubert JW, Claasen SJ. (2006). A sequential insertion heuristic for the initial.*ORiON : The Journal of ORSSA*, 22, 105–116.

Kallehauge, B., Larsen, J., Madsen, O.B.G., (2001). Lagrangian Duality Applied on Vehicle Routing Problem with Time Windows. *Technical Report, IMM*, Technical University of Denmark.

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

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

Li & Lim *benchmark*. Diakses tanggal 28 Maret 2012 dari http://www.sintef.no/Projectweb/TOP/Problems/PDPTW/Li--Lim-benchmark/

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

S, Mitrovic-Minic. (1998) . Pickup and Delivery Problem with Time Windows: A Survey. *Technical Report 1998-21, SCS Simon Fraser University.*

Solomon, M. M. (1987). Algorithms for The Vehicle Routing and Scheduling Problems with Time Window Constrains. *Operations Research*, 35, 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*.

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

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

LAMPIRAN

Lampiran 1 *Source Code* MATLAB

File semua.m

```
clear;
clc;
fprintf('Masukkan data yang di inginkan\n');
[file path] = uigetfile('*.txt','Pilih data benchmark');
fprintf('File %s dipilih\n',file);
fprintf('================================================')
fprintf('\nsilahkan masukkan parameter yang anda inginkan\n')
fprintf('================================================')
fprintf('\nKapasitas Kendaraan yang tersedia')
kapasitas = input('\nmasukkan kapasitas kendaraan ');
fprintf('\nInsertion Heuristic\n');
fprintf('====================');
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('====================');
fprintf('\nSimuated Annealing\n');
fprintf('====================');
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('============================');
fprintf('\nLarge Neighborhood Search\n');
fprintf('============================');
iterasi_LNS = input('\nMasukkan iterasi untuk LNS ');
UK = input ('\nMasukkan banyaknya pasang pelanggan yang akan
direlokasi ');
insertion
SAnih
LNSnih
Total_waktu = waktu_insertion + waktu_SA + waktu_LNS
```

File insertion.m

```
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)];
```


Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

```
        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;    %kendala kapasitas
                i=1;
                b=cek_kendala(r2,m,kapasitas);
                %kendala time windows
                if b<=kapasitas
                    output=cek_timewindows2(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
```

File SAnih.m

```
tic;
```


Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

```
%% 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_timewindows2(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 masing2 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;
```


Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

```
            end
        end
        Q_place1(fgh,:)=[];
    end
end
Q_baru;
Q_sementara = Q_baru{1};

%membandingkan 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(3)
                            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
```

File LNSnih.m

```
tic;
hasil=zeros(1,2);
jumlah_rute = size(R_SA,2);
total_jarak = distance_sa;
R = R_SA;
%% LNS
for cak = 1 : iterasi_LNS
    cak;
    D = R;
    sigma_R = [jumlah_rute total_jarak];
    c_copy = c;
    rel_custo = zeros(1,2);
```


Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

```
    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
        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
```


Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

```
        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',hasil(cak,1));
fprintf('\nTotal jarak adalah %d\n',hasil(cak,2));
disp ('Rute-rutenya adalah');
for i=1:length(R)
    disp(R{i}-1);
end
waktu_LNS = toc
```


Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

Terdapat beberapa fungsi yang digunakaN, antara lain :

File ins.m

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

File 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
```

File 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
```

File 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),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
```

Lampiran 2 Solusi *benchmark* 100 pelanggan

LC 101	Rute 1	0 – 67 – 65 – 63 – 62 – 74 – 72 – 61 – 64 – 102 – 68 – 66 – 69 – 0
	Rute 2	0 – 81 – 78 – 104 – 76 – 71 – 70 – 73 – 77 – 79 – 80 – 0
	Rute 3	0 – 98 – 96 – 95 – 94 – 92 – 93 – 97 – 106 – 100 – 99 – 0
	Rute 4	0 – 20 – 24 – 25 – 27 – 29 – 30 – 28 – 26 – 23 – 103 – 22 – 21 – 0
	Rute 5	0 – 57 – 55 – 54 – 53 – 56 – 58 – 60 – 59 – 0
	Rute 6	0 – 43 – 42 – 41 – 40 – 44 – 46 – 45 – 48 – 51 – 101 – 50 – 52 – 49 – 47 – 0
	Rute 7	0 – 90 – 87 – 86 – 83 – 82 – 84 – 85 – 88 – 89 – 91 – 0
	Rute 8	0 – 5 – 3 – 7 – 8 – 10 – 11 – 9 – 6 – 4 – 2 – 1 – 75 – 0
	Rute 9	0 – 32 – 33 – 31 – 35 – 37 – 38 – 39 – 36 – 105 – 34 – 0
	Rute 10	0 – 13 – 17 – 18 – 19 – 15 – 16 – 14 – 12 – 0
LC 102	Rute 1	0 – 13 – 17 – 18 – 19 – 15 – 16 – 14 – 12 – 0
	Rute 2	0 – 90 – 87 – 86 – 83 – 82 – 84 – 85 – 88 – 89 – 91 – 0
	Rute 3	0 – 32 – 33 – 31 – 35 – 37 – 38 – 39 – 104 – 36 – 34 – 0
	Rute 4	0 – 57 – 55 54 53 56 58 60 59 – 0
	Rute 5	0 – 5 – 3 – 7 – 8 – 10 – 11 – 9 – 6 – 4 – 2 – 1 – 75 – 0
	Rute 6	0 – 81 – 78 – 105 – 76 – 71 – 70 – 73 – 77 – 79 – 80 – 0
	Rute 7	0 – 43 – 42 41 40 44 46 45 48 51 50 106 52 49 – 47 – 0
	Rute 8	0 – 98 – 96 – 95 – 94 – 92 – 93 – 97 – 101 – 100 – 99 – 0
	Rute 9	0 – 67 – 65 – 63 – 62 – 74 – 72 – 61 – 64 – 68 – 102 – 66 – 69 – 0
	Rute 10	0 – 20 – 24 – 25 – 27 – 29 – 30 – 28 – 103 – 26 – 23 – 22 – 21 – 0
LC 105	Rute 1	0 – 90 – 87 – 86 – 83 – 82 – 84 – 85 – 88 – 89 – 91 – 0
	Rute 2	0 – 43 – 42 – 41 – 40 – 44 – 46 – 45 – 48 – 51 – 50 – 52 – 104 – 49 – 47 – 0
	Rute 3	0 – 81 – 78 – 76 – 71 – 70 – 73 – 77 – 79 – 102 – 80 – 0
	Rute 4	0 – 98 – 96 – 95 – 94 – 92 – 93 – 97 – 100 – 99 – 105 – 0
	Rute 5	0 – 57 – 55 – 54 – 53 – 56 – 58 – 60 – 59 – 0
	Rute 6	0 – 67 – 65 – 103 – 63 – 62 – 74 – 72 – 61 – 64 – 68 – 66 – 69 – 0
	Rute 7	0 – 13 – 17 – 18 – 19 – 15 – 16 – 14 – 12 – 0
	Rute 8	0 – 32 – 33 – 31 – 35 – 37 – 38 – 39 – 36 – 34 – 101 – 0
	Rute 9	0 – 5 – 3 – 7 – 8 – 10 – 11 – 9 – 6 – 4 – 2 – 1 – 75 – 0
	Rute 10	0 – 20 – 24 – 25 – 27 – 29 – 30 – 28 – 26 – 23 – 106 – 22 – 21 – 0
LC 106	Rute 1	0 – 67 – 65 – 63 – 62 – 74 – 72 – 61 – 64 – 68 – 66 – 103 – 69 – 0
	Rute 2	0 – 57 – 55 – 54 – 53 – 56 – 58 – 60 – 59 – 0
	Rute 3	0 – 13 – 17 – 18 – 19 – 15 – 16 – 14 – 12 – 0
	Rute 4	0 – 20 – 24 – 25 – 27 – 29 – 30 – 28 – 26 – 23 – 22 – 105 – 21 – 0

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

	Rute 5	0 – 98 – 96 – 95 – 94 – 92 – 93 – 97 – 100 – 106 – 99 – 0
	Rute 6	0 – 32 – 33 – 31 – 35 – 37 – 38 – 39 – 36 – 102 – 34 – 0
	Rute 7	0 – 81 – 104 – 78 – 76 – 71 – 70 – 73 – 77 – 79 – 80 – 0
	Rute 8	0 – 5 – 3 – 7 – 8 – 10 – 11 – 9 – 6 – 4 – 2 – 1 – 75 – 0
	Rute 9	0 – 43 – 42 – 41 – 40 – 44 – 46 – 45 – 48 – 51 – 50 – 101 – 52 49 – 47 – 0
	Rute 10	0 – 90 – 87 – 86 – 83 – 82 – 84 – 85 – 88 – 89 – 91 – 0
LC 107	Rute 1	0 – 98 – 96 – 95 – 94 – 92 – 93 – 97 – 100 – 106 – 99 – 0
	Rute 2	0 – 20 – 24 – 25 – 27 – 29 – 30 – 28 – 26 – 23 – 102 – 22 – 21 – 0
	Rute 3	0 – 81 – 104 – 78 – 76 – 71 – 70 – 73 – 77 – 79 – 80 – 0
	Rute 4	0 – 67 – 65 – 63 – 62 – 74 – 72 – 61 – 103 – 64 – 68 – 66 – 69 – 0
	Rute 5	0 – 57 – 55 – 54 – 53 – 56 – 58 – 60 – 59 – 0
	Rute 6	0 – 32 – 33 – 31 – 35 – 37 – 38 – 39 – 101 – 36 – 34 – 0
	Rute 7	0 – 90 – 87 – 86 – 83 – 82 – 84 – 85 – 88 – 89 – 91 – 0
	Rute 8	0 – 5 – 3 – 7 – 8 – 10 – 11 – 9 – 6 – 4 – 2 – 1 – 75 – 0
	Rute 9	0 – 43 – 42 – 41 – 40 – 44 – 46 – 45 – 48 – 51 – 50 – 52 – 49 47 – 105 – 0
	Rute 10	0 – 13 – 17 – 18 – 19 – 15 – 16 – 14 – 12 – 0
LC 108	Rute 1	0 – 81 – 78 – 76 – 71 – 70 – 73 – 77 – 79 – 103 – 80 – 0
	Rute 2	0 – 57 – 55 – 54 – 53 – 56 – 58 – 60 – 59 – 68 – 102 – 0
	Rute 3	0 – 98 – 96 – 95 – 94 – 92 – 93 – 97 – 101 – 100 – 99 – 2 – 75 – 0
	Rute 4	0 – 90 – 87 – 86 – 83 – 82 – 84 – 85 – 88 – 89 – 91 – 0
	Rute 5	0 – 20 – 24 – 25 – 27 – 106 – 29 – 30 – 28 – 26 – 23 – 22 – 21 – 0
	Rute 6	0 – 43 – 42 – 41 – 40 – 44 – 46 – 45 – 48 – 51 – 50 – 52 – 49 – 105 – 47 – 0
	Rute 7	0 – 67 – 65 – 63 – 62 – 74 – 72 – 61 – 64 – 66 – 69 – 0
	Rute 8	0 – 5 – 3 – 7 – 8 – 10 – 11 – 9 – 6 – 4 – 1 – 0
	Rute 9	0 – 32 – 33 – 31 – 35 – 37 – 38 – 39 – 104 – 36 – 34 – 0
	Rute 10	0 – 13 – 17 – 18 – 19 – 15 – 16 – 14 – 12 – 0
LC 2 01	Rute 1	0 – 67 – 63 – 62 – 74 – 72 – 61 – 64 – 66 – 69 – 68 – 65 – 49 – 55 – 54 – 53 – 56 – 58 – 60 – 59 – 57 – 40 – 44 – 46 – 45 – 51 – 50 – 52 – 47 – 43 – 42 – 41 – 48 – 0
	Rute 2	0 – 93 – 5 – 75 – 2 – 1 – 99 – 100 – 97 – 92 – 94 – 95 – 98 – 7 – 3 – 4 – 89 – 91 – 88 – 84 – 86 – 83 – 82 – 102 – 85 – 76 – 71 – 70 – 73 – 80 – 79 – 81 – 78 – 77 – 96 – 87 – 90 – 0
	Rute 3	0 – 20 – 22 – 24 – 27 – 30 – 29 – 6 – 32 – 33 – 31 – 35 – 37 – 38 – 39 – 36 – 34 – 28 – 26 – 101 – 23 – 18 – 19 – 16 – 14 – 12 – 15 – 17 – 13 – 25 – 9 – 11 – 10 – 8 – 21 – 0

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

LR 201	Rute 1	0 – 33 – 63 – 92 – 42 – 14 – 44 – 98 – 99 – 61 – 8 – 18 – 86 – 84 – 102 – 49 – 48 – 17 – 93 – 60 – 89 – 0
	Rute 2	0 – 95 – 59 – 5 – 45 – 82 – 47 – 36 – 64 – 11 – 19 – 62 – 88 7 – 16 – 38 – 85 – 94 – 6 – 46 – 96 – 37 – 43 – 91 – 100 – 13 – 58 – 0
	Rute 3	0 – 2 – 72 – 39 – 21 – 15 – 23 – 67 – 75 – 73 – 40 – 53 – 87 – 22 – 41 – 57 – 97 – 26 – 56 – 74 – 54 – 55 – 4 – 25 – 101 – 80 – 77 – 0
	Rute 4	0 – 65 – 83 – 31 – 69 – 52 – 27 – 28 – 12 – 29 – 76 – 30 – 71 – 78 – 79 – 81 – 9 – 51 – 90 – 34 – 3 – 50 – 20 – 10 – 32 – 66 – 35 – 68 – 24 – 1 – 70 – 0
LRC 101	Rute 1	0 – 33 – 28 – 29 – 30 – 26 – 34 – 32 – 93 – 0
	Rute 2	0 – 63 – 76 – 104 – 51 – 22 – 49 – 20 – 24 – 0
	Rute 3	0 – 5 – 45 – 2 – 7 – 6 – 8 – 3 – 1 – 105 – 4 – 0
	Rute 4	0 – 69 – 98 – 88 – 53 – 78 – 60 – 100 – 70 – 0
	Rute 5	0 – 83 – 23 – 21 – 19 – 18 – 48 – 25 – 101 – 0
	Rute 6	0 – 82 – 11 – 15 – 16 – 9 – 10 – 13 – 17 – 0
	Rute 7	0 – 72 – 36 – 38 – 41 – 40 – 43 – 37 – 35 – 0
	Rute 8	0 – 65 – 52 – 99 – 57 – 86 – 74 – 0
	Rute 9	0 – 64 – 103 – 90 – 84 – 56 – 66 – 0
	Rute 10	0 – 39 – 42 – 44 – 61 – 81 – 96 – 54 – 68 – 0
	Rute 11	0 – 92 – 95 – 62 – 67 – 71 – 94 – 50 – 80 – 0
	Rute 12	0 – 27 – 31 – 85 – 102 – 89 – 91 – 0
	Rute 13	0 – 59 – 75 – 87 – 97 – 58 – 77 – 0
	Rute 14	0 – 14 – 47 – 12 – 73 – 106 – 79 – 46 – 55 – 0

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

Lampiran 3 Solusi *benchmark* 200 pelanggan

LC 1_2_1	Rute 1	0 – 133 – 48 – 26 – 152 – 40 – 153 – 169 – 89 – 105 – 15 – 59 – 198 – 0
	Rute 2	0 – 113 – 155 – 78 – 175 – 13 – 43 – 2 – 90 – 67 – 39 – 107 – 212 – 0
	Rute 3	0 – 93 – 55 – 135 – 58 – 202 – 184 – 199 – 37 – 81 – 138 – 0
	Rute 4	0 – 148 – 103 – 197 – 203 – 124 – 141 – 69 – 200 – 0
	Rute 5	0 – 62 – 131 – 44 – 102 – 146 – 208 – 68 – 76 – 0
	Rute 6	0 – 114 – 159 – 38 – 150 – 22 – 151 – 16 – 140 – 204 – 187 – 142 – 111 – 63 – 56 – 0
	Rute 7	0 – 177 – 3 – 88 – 8 – 186 – 127 – 98 – 157 – 137 – 183 – 0
	Rute 8	0 – 21 – 23 – 182 – 75 – 163 – 194 – 145 – 195 – 52 – 92 – 0
	Rute 9	0 – 170 – 134 – 50 – 156 – 112 – 168 – 79 – 205 – 29 – 87 – 42 – 123 – 0
	Rute 10	0 – 32 – 171 – 65 – 86 – 115 – 94 – 51 – 174 – 136 – 189 – 0
	Rute 11	0 – 57 – 118 – 83 – 143 – 176 – 36 – 206 – 33 – 121 – 165 – 188 – 108 – 0
	Rute 12	0 – 20 – 41 – 85 – 80 – 31 – 25 – 172 – 77 – 110 – 162 - 0
	Rute 13	0 – 101 – 144 – 119 – 166 – 35 – 126 – 71 – 9 – 1 – 99 – 53 – 201 – 0
	Rute 14	0 – 30 – 120 – 19 – 192 – 196 – 97 – 14 – 96 – 130 – 28 – 74 – 149 – 0
	Rute 15	0 – 190 – 5 – 10 – 193 – 46 – 128 – 106 – 167 – 207 – 34 – 95 – 158 – 0
	Rute 16	0 – 60 – 211 – 82 – 180 – 84 – 191 – 125 - 4 – 72 – 17 - 0
	Rute 17	0 – 73 – 116 – 12 – 129 – 11 – 6 – 122 – 139 – 0
	Rute 18	0 – 164 – 210 – 66 – 147 – 160 – 47 – 91 – 70 – 0
	Rute 19	0 – 161 – 104 – 18 – 54 – 185 – 132 – 7 – 181 – 117 – 49 – 0
	Rute 20	0 – 45 – 178 – 27 – 173 – 154 – 209 – 24 – 61 – 100 – 64 – 179 – 109 – 0
LC 2_2_1	Rute 1	0 – 104 – 64 – 137 – 198 – 164 – 174 – 115 – 140 – 26 – 18 – 57 – 113 – 131 – 169 – 192 – 84 – 27 – 53 – 56 – 185 – 180 – 37 – 123 – 121 – 124 – 55 – 85 – 88 – 135 – 1 – 111 – 157 – 34 – 176 – 114 – 68 – 0
	Rute 2	0 – 28 – 136 – 171 – 63 – 173 – 94 – 99 – 70 – 162 – 46 – 187 - 60 – 30 – 108 – 133 - 16 – 61 – 72 – 62 – 122 – 24 – 92 – 49 - 160 – 117 – 82 – 17 – 7 – 170 – 165 – 58 – 66 – 52 – 105 – 0
	Rute 3	0 – 188 – 196 – 142 – 107 - 35 – 3 – 199 – 189 – 47 – 175 – 89 – 179 – 9 – 19 – 146 – 202 – 186 – 161 – 54 – 147 - 195 – 156 – 79 – 4 – 11 – 23 – 0
	Rute 4	0 - 200 - 41 - 183 - 65 - 116 - 15 - 155 - 36 – 0
	Rute 5	0 - 144 - 42 - 110 - 6 - 203 - 184 - 148 - 29 - 2 - 138 - 143 - 126 - 128 - 48 - 145 - 25 - 90 - 193 - 20 - 71 - 127 - 5 - 194 - 190 - 80 - 158 - 101 - 91 - 197 - 44 - 106 - 159 - 83 - 14 – 0

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

	Rute 6	0 – 178 – 167 – 40 – 39 - 119 – 38 – 166 – 102 – 98 – 76 – 8 – 78 – 177 – 134 – 22 – 191 – 150 – 43 – 151 – 69 – 97 – 31 – 181 – 172 – 87 – 163 – 75 – 168 – 154 – 21 – 51 – 201 – 0
	Rute 7	0 – 67 – 132 – 10 – 153 – 59 – 86 – 93 – 118 – 182 – 45 - 77 – 33 – 13 – 12 - 73 – 32 – 50 – 96 - 139 – 103 – 141 – 74 – 100 – 81 – 130 - 112 – 120 – 125 – 109 - 149 – 152 – 204 - 95 – 129 – 0
LR 1_2_1	Rute 1	0 – 158 – 51 – 143 – 187 - 171 – 122 – 181 – 101 – 154 - 148 – 119 – 201 – 0
	Rute 2	0 – 140 – 57 – 173 – 104 – 95 – 58 – 29 – 96 – 174 – 97 – 0
	Rute 3	0 – 157 – 93 – 199 – 135 – 80 – 64 – 109 – 182 – 202 – 10 – 0
	Rute 4	0 – 196 – 13 – 20 – 163 – 100 – 136 – 142 - 159 – 102 – 10 - 205 – 198 – 0
	Rute 5	0 – 7 – 41 – 36 – 86 – 131 – 130 – 79 – 156 – 172 – 91 – 0
	Rute 6	0 – 72 – 147 – 153 – 166 – 175 – 56 – 33 – 144 – 186 – 179 – 203 – 190 – 0
	Rute 7	0 – 87 – 145 – 35 – 60 – 99 – 62 – 84 – 118 – 183 – 200 – 209 – 59 – 0
	Rute 8	0 – 168 – 92 – 48 – 146 – 121 – 32 – 138 – 83 – 45 – 24 – 0
	Rute 9	0 – 149 – 81 – 18 – 150 – 137 – 14 – 165 – 69 – 134 – 204 – 126 – 197 – 0
	Rute 10	0 – 117 – 5 – 176 – 31 – 103 – 9 – 28 – 75 – 0
	Rute 11	0 – 132 – 115 – 139 - 162 – 170 – 78 – 0
	Rute 12	0 – 63 – 44 – 50 – 66 – 160 – 180 – 167 – 70 – 0
	Rute 13	0 – 46 – 184 – 206 – 71 – 193 – 151 – 129 – 141 – 191 – 110 – 0
	Rute 14	0 – 106 – 85 – 94 – 12 – 19 - 178 - 23 - 22 - 47 - 82 - 74 - 108 - 0
	Rute 15	0 – 112 – 161 – 2 – 8 – 76 – 17 – 40 – 105 – 0
	Rute 16	0 – 4 – 188 – 15 – 54 – 89 – 125 – 114 – 39 – 37 – 164 – 65 – 43 – 0
	Rute 17	0 – 55 – 67 – 3 – 68 – 133 – 207 – 195 – 25 – 116 – 210 – 11 – 49 – 27 – 61 – 0
	Rute 18	0 – 155 – 120 – 26 – 123 – 113 – 21 – 124 – 185 – 169 – 53 – 192 – 42 – 0
	Rute 19	0 – 73 – 30 – 208 – 152 – 34 – 38 – 194 – 90 – 88 – 111 – 128 – 52 – 0
	Rute 20	0 – 1 – 16 – 98 – 6 – 189 – 177 – 127 – 77 – 0

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

LR 2_2_1	Rute 1	0 – 62 – 192 – 36 – 33 – 179 – 137 – 117 – 177 – 35 – 162 – 134 – 63 – 135 – 170 – 165 – 90 - 14 – 142 – 66 – 166 – 105 – 123 – 130 – 198 – 17 – 16 – 89 – 70 – 174 – 58 – 5 – 55 – 111 – 82 – 2 – 110 – 12 – 109 – 20 – 182 – 54 – 190 – 25 – 153 – 148 – 86 – 171 – 202 – 0
	Rute 2	0 – 74 – 34 – 4 – 100 – 73 – 79 – 91 – 161 – 151 – 150 – 94 – 78 – 57 - 11 – 141 – 133 – 124 – 144 – 180 – 157 – 84 – 64 - 189 – 98 – 106 – 28 – 76 – 30 – 121 – 10 – 50 – 126 – 169 – 99 – 39 – 163 - 0
	Rute 3	0 – 18 – 168 – 164 - 183 – 181 – 8 – 60 – 61 – 77 – 152 – 143 – 119 – 71 – 197 – 95 – 194 – 21 – 6 – 40 – 32 – 128 – 44 – 200 – 191 – 178 – 146 – 193 – 27 – 68 – 38 – 13 – 196 – 102 – 75 – 145 – 69 – 122 – 107 – 138 – 186 – 159 – 187 - 7 – 156 – 136 – 24 – 114 – 127 – 112 – 80 – 53 – 154 – 85 – 46 – 87 – 201 – 0
	Rute 4	0 – 131 – 23 – 26 – 45 – 47 – 65 – 139 – 88 – 195 – 15 – 51 – 104 – 3 – 176 – 158 – 149 – 113 – 93 – 83 – 173 – 22 – 96 – 0
	Rute 5	0 – 67 - 41 – 116 – 56 – 199 – 118 – 31 – 48 – 167 – 184 – 19 – 43 – 103 – 132 – 92 – 120 – 172 – 81 – 59 – 29 – 147 – 9 – 49 – 37 – 129 – 125 – 160 – 101 – 185 – 42 – 108 – 188 – 155 – 175 – 97 – 52 – 1 – 140 – 115 – 72 – 0
LRC 1_2_1	Rute 1	0 – 153 – 47 – 200 – 103 – 90 – 79 – 204 – 96 – 34 – 175 – 0
	Rute 2	0 – 37 – 177 – 65 – 99 – 94 – 112 – 114 – 86 – 55 – 81 - 107 – 154 – 0
	Rute 3	0 – 140 – 100 – 143 – 173 – 188 – 29 – 139 – 21 – 166 – 87 – 121 – 208 – 115 – 11 – 30 – 67 – 0
	Rute 4	0 – 42 – 181 – 35 – 164 – 104 – 25 – 10 – 211 – 77 – 122 – 141 – 185 – 0
	Rute 5	0 – 128 – 168 – 53 – 151 – 160 – 109 – 189 – 212 – 125 – 17 – 0
	Rute 6	0 – 156 – 102 – 7 – 124 – 43 – 180 – 3 – 110 – 80 – 178 – 93 – 193 – 19 – 146 – 0
	Rute 7	0 – 4 – 182 – 57 – 133 – 147 – 40 – 16 – 22 – 106 – 70 – 0
	Rute 8	0 – 88 – 155 – 144 – 71 – 0
	Rute 9	0 – 131 – 24 – 196 - 68 – 184 – 136 – 0
	Rute 10	0 – 84 – 199 – 127 – 118 – 105 – 69 – 176 – 113 – 191 – 58 – 134 – 174 – 0
	Rute 11	0 – 54 – 9 – 12 – 59 – 89 – 82 – 0
	Rute 12	0 – 165 – 85 - 45 – 56 – 108 – 38 – 32 – 203 – 23 – 117 – 50 – 33 – 66 – 207 – 0
	Rute 13	0 – 48 – 15 – 179 – 18 – 183 – 190 – 27 – 51 – 44 – 95 – 126 – 206 – 0
	Rute 14	0 – 142 – 152 – 167 – 61 – 111 – 198 – 91 – 202 – 145 – 116 – 83 – 201 – 148 – 36 – 0
	Rute 15	0 – 92 – 195 - 31 – 52 – 74 – 101 – 192 – 157 – 0

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012

	Rute 16	0 – 162 – 63 – 149 - 26 – 62 – 159 – 20 – 46 – 138 – 186 – 120 – 39 – 0
	Rute 17	0 – 163 – 28 – 205 – 6 – 49 – 14 – 132 – 78 – 171 – 130 – 187 – 150 – 137 – 123 – 209 – 13 – 8 – 1 – 60 – 75 - 0
	Rute 18	0 – 169 – 98 – 119 – 161 – 72 – 129 – 41 – 194 – 210 – 158 – 0
	Rute 19	0 – 172 – 197 – 5 – 170 – 73 – 135 – 2 – 76 – 97 – 64 – 0

Aplikasi algoritma..., Risya Priwarnela, FMIPA UI, 2012