TESIS TI142307

PENGEMBANGAN ALGORITMA *CROSS ENTROPY GENETIC ALGORITHM* UNTUK PENYELESAIAN *TWO DIMENSIONAL LOADING HETEROGENEOUS FLEET VEHICLE ROUTING PROBLEM*

DOMINICO LAKSMA PARAMESTHA
2514203203

DOSEN PEMBIMBING
Prof. Ir. BUDI SANTOSA, M.S., Ph.D.

PROGRAM MAGISTER

BIDANG KEAHLIAN MANAJEMEN LOGISTIK DAN RANTAI PASOK

JURUSAN TEKNIK INDUSTRI

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI SEPULUH NOPEMBER

SURABAYA

2016

TESIS TI142307

DEVELOPING CROSS ENTROPY GENETIC ALGORITHM FOR SOLVING TWO DIMENSIONAL LOADING HETEROGENEOUS FLEET VEHICLE ROUTING PROBLEM (2L-HFVRP)

DOMINICO LAKSMA PARAMESTHA
2514203203

SUPERVISOR
Prof. Ir. BUDI SANTOSA, M.S., Ph.D.

MAGISTER PROGRAM
SUPPLY CHAIN MANAGEMENT
DEPARTMENT OF INDUSTRIAL ENGINEERING
FACULTY OF INDUSTRIAL TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2016

PENGEMBANGAN ALGORITMA *CROSS ENTROPY GENETIC ALGORITHM* UNTUK PENYELESAIAN *TWO DIMENSIONAL LOADING HETEROGENEOUS FLEET VEHICLE ROUTING PROBLEM*

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Teknik (MT)
di
Institut Teknologi Sepuluh Nopember Surabaya

Oleh :

DOMINICO LAKSMA PARAMESTHA
NRP. 2514 203 203

Tanggal Ujian : 5 Januari 2017
Periode Wisuda : Maret 2017

Disetujui oleh Tim Penguji Tesis:

1. Prof. Ir. Budi Santosa, MS., Ph.D. (Pembimbing)
   NIP. 196905121994021001

2. Prof. Dr. Ir. Suparno, MSIE (Penguji)
   NIP. 194807101976031002

3. Nurhadi Siswanto, S.T, MSIE., Ph.D. (Penguji)
   NIP. 197005231996011001

an. Direktur Program Pascasarjana
Asisten Direktur

Prof. Dr. Ir. Tri Widjaja, M.Eng.
NIP. 19611021 198603 1 001

Direktur Program Pascasarjana,

Prof. Ir. Djauhar Manfaat, M.Sc., Ph.D.
NIP. 19601202 198701 1 001

LEMBAR PERNYATAAN KEASLIAN TESIS

Saya yang bertanda tangan di bawah ini:

Nama	: Dominico Laksma Paramestha
Program Studi	: Magister Teknik Industri ITS
NRP	: 2514203203

Menyatakan bahwa isi sebagian maupun keseluruhan tesis saya yang berjudul:

"PENGEMBANGAN ALGORITMA *CROSS ENTROPY GENETIC ALGORITHM* UNTUK PENYELESAIAN *TWO DIMENSIONAL LOADING HETEROGENEOUS FLEET VEHICLE ROUTING PROBLEM"

adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan, dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Seluruh referensi yang dikutip dan dirujuk telah saya tulis secara lengkap di daftar pustaka

Apabila di kemudian hari ternyata pernyataan saya ini tidak benar, maka saya bersedia menerima sanksi sesuai dengan peraturan yang berlaku.

Surabaya, Januari 2017

Yang membuat pernyataan

Dominico Laksma Paramestha

NRP. 2514203203

PENGEMBANGAN ALGORITMA *CROSS ENTROPY GENETIC ALGORITHM* UNTUK PENYELESAIAN *TWO DIMENSIONAL LOADING HETEROGENEOUS FLEET VEHICLE ROUTING PROBLEM*

Nama Mahasiswa : Dominico Laksma Paramestha
NRP : 2514203203
Pembimbing : Prof. Ir. Budi Santosa, M.S., Ph.D.

ABSTRAK

Two dimensional loading heterogeneous fleet vehicle routing problem (2L-HFVRP) merupakan permasalahan kombinatorial gabungan dari *vehicle routing problem* dan *bin packaging problem.* Pada 2L-HFVRP pesanan konsumen berupa muatan berbentuk persegi panjang yang harus dikirim dari depot dengan beberapa *vehicle* yang memiliki kapasitas container/box yang berbeda-beda. Tujuan dari 2L-HFVRP adalah meminimalkan total biaya perjalanan dari depot ke seluruh konsumen dan kembali ke depot. Setiap rute yang terbentuk harus *feasible* dengan kapasitas dan proses penataan muatan untuk dalam setiap kendaraan.

Proses penataan muatan menggunakan 5 heuristik yang digunakan oleh Zachariadis *et.al* (2009). Penelitian ini menggunakan dua skenario dalam penataan muatan ke dalam kontainer. Skenario pertama (*sequential*) adalah menata muatan sesuai dengan urutan kunjungan dan bongkar muat, agar proses bongkar muat dapat dilakukan dengan mudah. Skenario kedua (*unrestricted*) adalah dengan menata muatan dengan memasukan muatan tanpa mempertimbangkan urutan kunjungan. Rotasi muatan dimungkinkan dalam proses penataan muatan untuk memaksimalkan utilitas ruang dari kontainer.

Sebagai gabungan dari dua *NP-hard problem* penelitian ini menggunakan algoritma *cross entropy genetic algorithm* (CEGA) untuk menyelesaikan 2L-HFVRP. Algoritma ini menutupi kekurangan dari algoritma CE dan GA jika digunakan sendiri dengan kelebihan dari masing-masing algoritma. Proses pencarian solusi awal menggunakan mekanisme matriks probabilitas transisi dibantu dengan mekanisme *local improvement* yang dikembangkan oleh Ai & Kachitvichyanukul (2009) untuk mendapatkan solusi awal yang baik. Hasil akan dibandingkan dengan algoritma lain dari penelitian sebelumnya untuk melihat performansi dari algoritma yang dikembangkan dalam penelitian ini.

Kata kunci: Metaheuristik, *Cross Entropy, Genetic Algorithm, Vehicle Routing Problem, Two-loading constraint*

DEVELOPING *CROSS ENTROPY GENETIC ALGORITHM* FOR SOLVING *TWO DIMENSIONAL LOADING HETEROGENEOUS FLEET VEHICLE ROUTING PROBLEM* (2L-HFVRP)

Name : Dominico Laksma Paramestha
NRP : 2514203203
Supervisor : Prof. Ir. Budi Santosa, M.S., Ph.D.

ABSTRACT

Two dimensional loading heterogeneous fleet vehicle routing problem (2L-HFVRP) is a combination of Heterogeneous Fleet VRP and a packing problem well-known as Two Dimensional Bin Packing Problem (BPP). 2L-HFVRP is a Heterogeneous Fleet VRP in which these costumer demands are formed by a set of two-dimensional rectangular weighted item. These demands must be served by a heterogeneous fleet of vehicles with a fix and variable cost from the depot. The objective function 2L-HFVRP is to minimize the total transportation cost which relies on the type of vehicle and the route while satisfying all of the customer demands. All formed routes must be consistent with the capacity and loading process of the vehicle.

In this paper, the loading process follows the 5 heuristic which was used by Zachariadis et.al (2009) and allowed to 90^{o} rotation position. Allowing rotation position on loading process is used to maximize the utility of vehicle. Sequential and unrestricted scenarios are considered on this paper. Sequential scenario is a scenario that deals with loading and unloading item into vehicle. The sequence of the loading process is affected by the sequence of the route. Whereas unrestricted scenario is a scenario that only deals with feasible loading of item into the vehicle.

We propose a metaheuristic which is a combination of the Genetic Algorithm (GA) and the Cross Entropy (CE) named Cross Entropy Genetic Algorithm (CEGA) to solve the 2L-HFVRP. The mutation concept on GA is used to speed up the algorithm CE to find the optimal solution. The mutation mechanism was based on local improvement that was developed by Ai & Kachitvichyanukul (2009). The probability transition matrix mechanism on CE is used to avoid getting stuck in local optimum. The effectiveness of CEGA was tested on benchmark instance based 2L-HFVRP. The result of experiments show a competitive result compared with another algorithm.

Keywords: Metaheuristik, *Cross Entropy, Genetic Algorithm, Vehicle Routing Problem, Two-loading constraint*

KATA PENGANTAR

Bersama dengan ini penulis mengucapkan puji syukur yang tiada henti kepada Tuhan Yang Maha Esa karena dengan segala rahmat karunia dan petunjuk-Nya penulis mampu menyelesaikan Tesis ini dengan baik.

Laporan tesis ini dugunakan sebagai syarat untuk menyelesaikan program studi Magister Teknik di bidang Logistik dan Rantai Pasok Jurusan Teknik Industri dengan judul "**PENGEMBANGAN ALGORITMA *CROSS ENTROPY GENETIC ALGORITHM* UNTUK PENYELESAIAN *TWO DIMENSIONAL LOADING HETEROGENEOUS FLEET VEHICLE ROUTING PROBLEM*"**.

Selama pelaksanaan dan penyusunan laporan Tugas Akhir ini saya telah menerima bantuan dari berbagai pihak. Oleh sebab itu, pada kesempatan ini penulis ingin menyampaikan terima kasih dan penghargaan kepada:

1. Kedua orang tua sekaligus "investor utama" Bapak Ignatius Suwarto dan Ibu Sudjarwati atas segala doa-doa yang tulus serta dukungan moral dan finansial yang telah diberikan.
2. Maria Laksmi Parahita dan Dionysius Arya yang selalu memberikan semangat dan nasihat untuk penulis dalam menyelesaikan kuliah dan tesis ini.
3. Bapak Prof. Ir. Budi Santosa, M.S., Ph.D. selaku dosen pembimbing utama penulis yang telah memberikan ilmu dan pengalamannya hingga penelitian ini dapat diselesaikan.
4. Bapak Prof. Ir. Suparno, MSIE., Ph.D. dan Bapak Nurhadi Siswanto, S.T, MSIE., Ph.D. selaku penguji yang dengan pengalamannya memberikan masukan terhadap penulis, sehingga penelitian ini dapat diselesaikan.
5. Bapak Erwin Widodo, S.T., M.Eng, D.Eng. selaku Ketua Jurusan Program Pasca Sarjana ITS yang selalu memberikan kami motivasi kami, mahasiswa pasca sarjana ITS dalam menyelesaikan studinya.
6. Seluruh dosen pengajar Program Pasca Sarjana Jurusan Teknis Industri ITS atas ilmu yang telah diberikan selama penulis menempuh studi, serta seluruh staf dan karyawan di jurusan Teknik Industri ITS, terima kasih atas bantuannya dalam kepengurusan hingga tesis ini selesai.

7. The Jin Ai, ST., MT., Dr. Eng. yang selalu memberikan perhatian dan inspirasi bagi penulis dalam menyelesaikan tesis dan kuliah di S2 TI ITS.
8. Seluruh teman S2 TI ITS angkatan 2014 genap: Rohman, Afifah, Satiti, Linda, Cici, Risma, dan Amir yang selalu memberikan dukungan penulis dalam menyelesaikan kuliah selama 4 semester dan pengerjaan tesis ini dengan menyenangkan.
9. Para penunggu residen, Mas Tonggeng, Aan, Danu, Isna, Hendra, Sony, Mba Opin, Hima, Putu, Dian, dll yang senantiasa memberikan semangat dan masukan penulis dalam menyelesaikan kuliah.
10. Mbak Prita dan Nasir yang mulai dari pengerjaan proposal dan tesis selalu menemani dan membantu penulis di residen dalam kebuntuan.
11. Keluarga Keputih Perintis 1 no 19, Ongky, Yohakim dan Suprianto yang selalu menemani dan memberi semangat penulis dalam "bertahan hidup" di Surabaya.
12. Semua teman-teman TI UAJY 2010 yang selalu memberikan semangat dan kegembiraan kepada penulis.
13. Dan seluruh rekan, teman dan saudara penulis yang tidak memungkinkan untuk disebutkan satu-persatu, terima kasih.

Akhir kata, dengan segala kerendahan hati penulis meminta maaf apabila ada kesalahan di dalam penulisan tesis ini dan semoga tesis ini dapat bermanfaar bagi para pembaca dan penelitian selanjutnya.

Surabaya, Januari 2017

Dominico Laksma Paramestha

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Proses *packing* dan pengiriman barang atau distribusi ke konsumen merupakan kegiatan yang penting pada suatu perusahaan manufaktur. Banyak usaha yang dilakukan untuk meminimasi biaya yang terjadi pada kegiatan tersebut. Sebelumnya masalah perencanaan rute dan masalah *packing* dalam distribusi menjadi objek terpisah dalam beberapa penelitian. Beberapa tahun terakhir banyak yang melakukan optimasi terhadap kombinasi objek penelitian tersebut. Optimasi dilakukan oleh banyak peneliti untuk mendapatkan solusi yang lebih efisien sehingga biaya dapat diminimalkan (Zachariadis dkk., 2009; Fuellerer dkk., 2009; Fuellerer dkk., 2010; Fernández dkk., 2013; Leung dkk., 2013; Bortfeldt, 2013; Li dan Zhang, 2015;Junqueira dan Morabito, 2015; Mahvash dkk., 2015; Hokama dkk., 2016; Männel dan Bortfeldt, 2016). Sehingga penelitian ini berusaha untuk mendekati kasus nyata dengan menambah *constraint* yang relevan dengan permasalahan nyata yang ada di perusahaan.

Terdapat beberapa jenis kasus distribusi yang popular sebagai obyek penelitian oleh beberapa peneliti. Salah satunya adalah *Vehicle Routing Problem* (VRP). VRP adalah permasalahan kombinatorial berupa penentuan rute perjalanan dalam distribusi oleh beberapa kendaraan. VRP memiliki peran penting dalam bidang logistik dan distribusi. Penelitian tentang VRP banyak dilakukan sejak tahun1959 ketika Danzig dan Ramser memperkenalkan *Traveling Salesman Problem* (TSP) (Yeun *et.al,* 2008). Kendaraan/*vehicle* yang digunakan memiliki kapasitas terbatas. Rute harus dimulai dan berakhir di titik (depot) yang sama. VRP sama dengan permasalahan yang sering terjadi di bagian distribusi pada sebuah perusahaan manufaktur dan distributor. Terdapat berbagai macam kegiatan distribusi yang terjadi di dunia nyata. Hal ini membuat banyak peneliti yang melakukan beberapa penyesuaian terhadap VRP murni terhadap kondisi tertentu. Sehingga banyaknya varian VRP pada berbagai penelitian yang sudah pernah dibuat. Variasi VRP biasanya tergantung pada *constraint* yang digunakan. Varian

paling sederhana adalah Capacitated Vehicle Routing Problem (CVRP) (Toth & Vigo, 2002a). CVRP sebagai varian VRP paling sederhana memiliki karakteristik berupa kapasitas yang tetap dan sama untuk semua kendaraan. Namun sering juga ditemukan perusahaan memiliki beberapa kendaraan dengan kapasitas yang berbeda-beda. Kondisi ini yang membuat munculnya varian VRP dengan kapasitas kendaraan yang berbeda-beda (Leung *et al.* 2013). Varian ini biasa disebut Heterogeneous Fleet Vehicle Routing Problem (HFVRP). Sebagai *NP-Hard* problem, VRP banyak diselesaikan dengan metode metaheuristik. Metode metaheuristik digunakan karena menghasilkan solusi yang baik dalam hal kualitas dan waktu komputasi. Rute adalah solusi yang dihasilkan dari sebuah permasalahan VRP. Rute yang terbentuk harus meminimalkan total biaya perjalanan dengan tidak melanggar *constraint* yang digunakan.

Permintaan dari pelanggan dapat berupa apa saja. Muatan dapat berupa curah ataupun berupa barang dalam kemasan. Muatan dapat berupa barang pecah belah yang tidak dapat ditumpuk ataupun berbentuk *box* dan palet yang dapat disusun bertingkat dalam proses penataan muatan pada kendaraan. Proses penataan muatan ke dalam kapasitas kendaraan menjadi permasalahan tersendiri. Permasalahan ini dapat digolongkan dalam disebut *bin packaging problem* (BPP). BPP merupakan permasalahan optimasi kombinatorial *NP-hard* dengan tujuan meminimalkan jumlah *bin* berbentuk persegi panjang yang digunakan untuk meletakan sekumpulan muatan berbentuk persegi panjang (Zachariadis *et al.* 2009). Banyak peneliti yang mengembangkan metode heuristic untuk menyelesaikan masalah ini. Beberapa peneliti yang mengembangkan metode heuristik ini adalah Chazelle, (1983), Lodi *et al.* (1999), Leung et al., (2013) Zachariadis et al., (2009), dan Li & Zhang, (2015),

Dalam VRP tradisional kapasitas kendaraan memiliki ukuran berupa jumlah unit dan berat. Jadi unit yang dibawa tidak boleh melebihi kapasitas kendaraan, jika kapasitas kendaraan berupa jumlah unit maksimum yang mampu dibawa kendaraan. Sedangkan jumlah berat unit yang dibawa tidak boleh melebihi kemampuan angkut kendaraan jika kapasitas kendaraan diartikan dalam ukuran berat total. Kenyataannya kapasitas kendaraan dapat berupa luas atau volume dari kendaraan jika kendaraan berjenis truk box atau container. VRP *with loading*

constraint adalah varian dari VRP yang mempertimbangkan hal tersebut. Dalam penelitian ini yang akan di bahas adalah *Two-dimensional loading* HFVRP (2L-HFVRP). *Two- dimensional* berarti muatan yang dibawa oleh kendaraan berbentuk persegi panjang tanpa mempertimbangkan tinggi muatan, dan kapasitas kendaraan berupa luasan permukaan dari *box* atau container yang dibawa kendaraan untuk mengangkut muatan permintaan konsumen. *Heterogeneous fleet* berarti terdapat jenis kendaraan berbeda yang dapat digunakan dalam melayani seluruh konsumen. 2L-HFVRP pada dasarnya adalah gabungan dari HFVRP dan BPP, karena dalam solusi rute yang terbentuk selain meminimalkan total ongkos perjalanan namun juga mempertimbangkan penyusunan muatan yang berdimensi panjang dan lebar ke dalam permukaan angkut kendaraan.

Beberapa peneliti sudah melakukan penelitian pada area ini. Seperti yang dilakukan oleh Li & Zhang, (2015), Fuellerer *et al* (2009) (2010), Leung *et al*. (2013), Zachariadis et al. (2009),dan Miao *et al.* (2012) yang menggunakan metaheuristik dalam menyelesaikan varian dari VRP *with loading constraint*. Sebagai gabungan dari *NP-Hard problem,* VRP *with loading constraint* banyak diselesaikan dengan metode *metaheuristic*. Walaupun tidak selalu menghasilkan solusi optimum, namun memiliki waktu komputasi yang relatif cepat. Beberapa metode metaheuristik yang sudah pernah digunakan adalah *tabu search* (Zachariadis et al., 2009), *ant colony* (Fuellerer et al., 2009), *simulated annealing* (Leung et al., 2013), *deferential evolution* (Li & Zhang, 2015) dan *Genetic algorithm* (Miao et al., 2012). Pada penelitian ini diusulkan algoritma metaheuristik *cross entropy genetic algorithm* (CEGA).

Cross entropy genetic algorithm (CEGA) adalah algoritma metaheuristik *hybrid* kombinasi dari algoritma *Cross entropy* dengan algoritma genetic (Santosa dkk., 2011). Algoritma *cross entropy* biasa digunakan untuk menyelesaikan kasus optimasi kontinyu. Pada masalah kombinatorial (diskret) algoritma ini menggunakan matriks transisi dalam menginisiasi solusi. Prinsip pada algoritma *cross entropy* adalah memusatkan solusi menuju ke solusi optimal. Pemusatan dilakukan dengan memperbarui parameter untuk setiap iterasi untuk meningkatkan kemungkinan kemunculan solusi yang memiliki solusi yang optimal. Algoritma genetik (GA) adalah algoritma yang didasarkan pada prinsip-prinsip genetika dalam

menghadapi seleksi alam. Individu yang memiliki kualitas yang baik akan bertahan terhadap seleksi alam. Proses mutasi dan kawin silang terjadi dalam proses seleksi alam. Begitu pula pada GA mekanisme mutasi dan kawin silang dilakukan untuk mencari solusi yang optimal (Santosa & Willy, 2011). Algoritma CE memiliki kelebihan dalam eksploitasi solusi. Kelebihan ini membuat CE sering terjebak dalam *local optimum* jika algoritma ini dijalankan sendiri. Sedangkan GA dengan mutasi dan kawin silang membuat algoritma ini memiliki eksplorasi yang baik (Lopez-garcia dkk., 2016). Penggunaan algoritma CE untuk menyelesaikan kasus dengan skala besar membutuhkan waktu komputasi yang tinggi. Mekanisme mutasi pada algoritma GA digunakan untuk mempercepat menemukan solusi yang optimal (Santosa & Willy, 2011). Gabungan konsep algoritma ini bertujuan untuk menutupi kelemahan dari masing-masing algoritma, sehingga mampu menghasilkan solusi yang optimal.

Solusi awal (*Initial Solution*) dalam metaheuristik menjadi kunci dalam menentukan kualitas solusi dan waktu komputasi yang dihasilkan. Inisialisasi solusi awal dilakukan menggunakan mekanisme matriks probabilitas transisi dengan beberapa penyesuaian. Mekanisme ini akan dilanjutkan dengan beberapa *local improvement* seperti pada penelitian Ai & Kachitvichyanukul, (2009). *Local improvement* yang digunakan adalah 2-*opt*, 1-1 *exchange*, dan 1-0 *exchange*. *Local improvement* ini didesain untuk memperbaiki rute pada kasus VRP sehingga penggunaan mekanisme ini akan menghasilkan solusi awal yang baik.

Penelitian ini melakukan pengujian CEGA dalam menyelesaikan kasus 2L-HFVRP. Sampai saat ini belum ada yang pernah menggunakan CEGA dalam menyelesaikan kasus 2L-HFVRP. Penelitian yang paling dekat dengan penelitian ini adalah penelitian yang dilakukan oleh Leung *et al*. (2013). Pada penelitiannya 2L-HFVRP diselesaikan menggunakan algoritma *Simulated Annealing* dengan 6 heuristik untuk menyelesaikan masalah BPP. Pada penelitian tersebut mempertimbangkan 2 versi dari 2L-HFVRP yaitu *Unrestricted* dan *Sequential. Sequential* adalah dimana proses bongkar muat harus mempertimbangkan urutan pelanggan yang akan dikunjungi. Sehingga pada saat bongkar muat di suatu konsumen tidak perlu memindahkan muatan untuk konsumen lainnya. *Unrestricted* adalah kondisi yang berlawanan dengan *sequential* dimana susunan dari muatan

tidak mempertimbangkan urutan pelanggan yang akan dikunjungi. Pada penelitian ini juga membahas *Unrestricted* dan *sequential* 2L-HFVRP. Algoritma CEGA digunakan dalam menyelesaikan permasalahan ini dengan 5 heuristik yang digunakan oleh Zachariadis et al., (2009). Jika pada penelitian Leung et al., (2013) muatan memiliki orientasi yang tetap, pada penelitian ini memungkinkan muatan dirotasi 90^o agar utilitas dari kontainer dapat dimaksimalkan.

1.2 Perumusan Masalah

Rumusan masalah yang dibahas pada penelitian ini adalah:

1. Bagaimana melakukan pengembangan algoritma *Cross Entropy Genetic Algorithm* dalam menyelesaikan kasus *two-dimensional loading constraint heterogeneous fleet vehicle routing problem*?
2. Apakah algoritma *Cross Entropy Genetic Algorithm* dapat menghasilkan solusi dan waktu komputasi yang kompetitif dengan solusi yang dihasilkan metode lain dalam penelitian sebelumnya?

1.3 Tujuan penelitian

Dalam Penelitian ini, tujuan yang ingin dicapai adalah sebagai berikut :

1. Mengembangkan algoritma *Cross Entropy Genetic Algorithm* agar dapat menyelesaikan kasus *two-dimensional loading constraint heterogeneous fleet vehicle routing problem*.
2. Menghasilkan solusi yang kompetitif dibanding metode lain dengan menggunakan algoritma *Cross Entropy Genetic Algorithm*.

1.4 Batasan masalah

1. Penelitian ini berfokus pada pengembangan algoritma CEGA dalam menyelesaikan kasus kombinatorial berupa 2L-HFVRP.
2. Muatan yang dimiliki oleh setiap konsumen merupakan muatan 2 dimensi dan menggunakan 5 heuristik dalam proses memasukan muatan ke kontainer.
3. Data set yang digunakan dalam menguji algoritma adalah data *instance* 1-15 yang sama dipakai pada penelitian Leung *et al.* (2013).

4. Fungsi tujuan yang digunakan dalam penelitian ini adalah meminimalkan biaya perjalanan yang terdiri dari *fix cost* dan *variable cost*. Dimensi dan berat muatan menjadi komponen konstrain dalam menyusun solusi.

1.5 Asumsi

Asumsi yang digunakan pada penelitian ini adalah:

1. Jumlah kendaraan yang digunakan memiliki kapasitas yang berbeda-beda dengan jumlah yang tidak terbatas dalam penggunaanya.
2. Total *Variable cost* hanya dipengaruhi oleh jarak tempuh antar dua titik.
3. *Fix cost* hanya dipengaruhi oleh jumlah dan jenis kendaraan yang digunakan.
4. Jarak dua titik diukur dengan jarak *Euclidian.*
5. Muatan dapat dirotasi sebesar 90^{o}.

1.6 Manfaat Penelitian

Manfaat penelitian ini untuk bidang keilmuan memiliki manfaat dalam mengeksplorasi metode metaheuristik CEGA untuk menyelesaikan kasus kombinatorial berupa 2L-HFVRP. Penelitian ini dapat digunakan sebagai acuan penelitian yang akan datang yang akan menggunakan algoritma CEGA.

1.7 Sistematika Penulisan

Pada penelitian ini menggunakan sistematika penulisan laporan sebagai berikut :

BAB I PENDAHULUAN

Pendahuluan berisi tentang beberapa hal yang mendasari dilakukannya penelitian. Bab ini terdiri dari beberapa sub bab seperti latar belakang, perumusan masalah, tujuan penelitian, batasan masalah, asumsi, dan manfaat penelitian.

BAB II TINJAUAN PUSTAKA

Tinjauan pustaka berisi tentang teori, temuan dan penelitian lain yang relevan dan digunakan sebagai acuan dan landasan dalam melaksanakan penelitian.

BAB III METODOLOGI PENELITIAN

Metodologi penelitian menjelaskan langkah-langkah secara sistematis yang dilakukan di penelitian untuk mencapai tujuan penelitian.

BAB IV DESKRIPSI MODEL dan PENGEMBANGAN ALGORITMA

Bab ini menjelaskan deskripsi model dari objek penelitian, *tools* yang digunakan secara rinci dan algoritma usulan yang dikembangkan dalam penelitian ini.

BAB V EKSPERIMEN dan ANALISA

Eksperimen dan analisa berisi tentang program yang dibuat dan *output*-nya. Hasil output dibandingkan dengan penelitian sebelumnya untuk dilakukan analisa.

BAB VI KESIMPULAN dan SARAN

Bab ini menjelaskan kesimpulan yang diperoleh dari hasil analisa yang dilakukan terhadap *output* yang dihasilkan algoritma. Saran diberikan untuk acuan pengembangan penelitian selanjutnya.

BAB 2
TINJAUAN PUSTAKA

Bab ini akan menguraikan teori, temuan dan penelitian sebelumnya yang akan mendukung kegiatan penelitian. Pada bagian terakhir di bab ini terdapat posisi penelitian untuk melihat posisi penelitian yang dilakukan dibanding dengan penelitian yang sudah ada.

2.1 *Vehicle Routing Problem* (VRP)

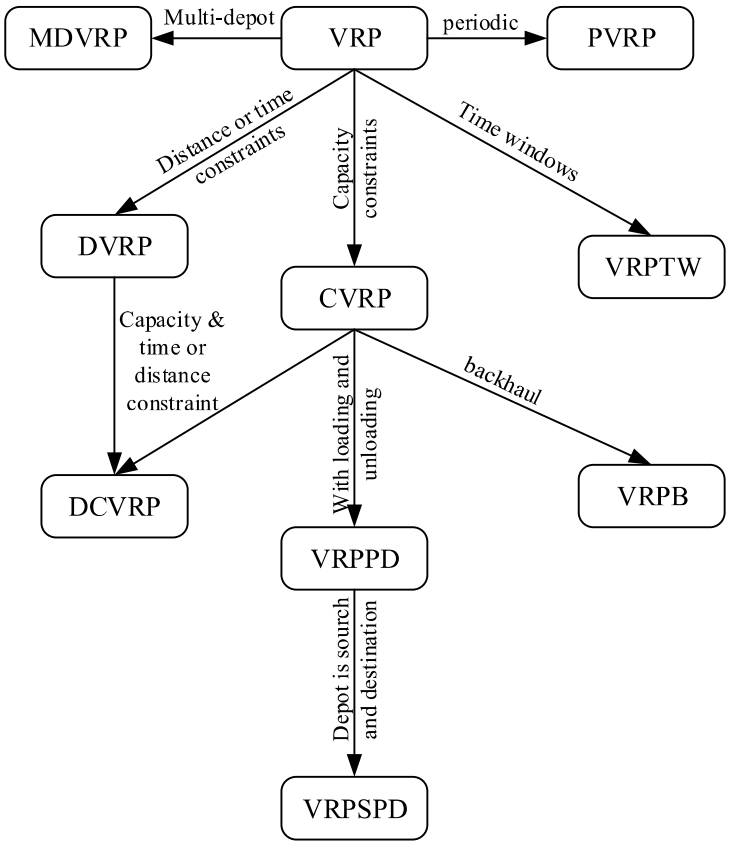
Vehicle Routing Problem (VRP) adalah suatu kasus distribusi barang dari depot (*distribution center*). Dantzig dan Ramser (1959) adalah yang pertama mengenalkan "*Truck Dispatching Problem"* suatu model tentang beberapa truk dengan kapasitas homogen melayani permintaan minyak dari sejumlah stasiun pengisian bahan bakar dari suatu depot dengan jarak perjalanan minimum sebagai fungsi tujuannya. Lima tahun berikutnya Clark dan Wright (1964) mendefinisikan bentuk umum optimasi linear dari kasus tersebut yang secara umum termasuk dalam bidang logistic dan transportasi. Hal ini yang sekarang biasa disebut sebagai VRP, merupakan salah satu topik yang secara luas dipelajari dalam bidang riset operasi (Braekers, *et.al* 2016). Berbeda dengan *Traveling Salesman Problem* (TSP), pada VRP terdapat beberapa *vehicle* dengan kapasitas tertentu. Kapasitas itu akan diisi oleh permintaan dari para konsumen yang harus dihantarkan. Secara lebih detail kasus VRP terdiri dari sekumpulan konsumen yang harus dilayani oleh sekumpulan *vehicle* yang berada di satu atau lebih depot. Setiap *vehicle* dioperasikan oleh sekumpulan pengendara yang bergerak dalam suatu *road network* (Toth & Vigo, 2002b). Setiap *vehicle* yang bergerak akan membentuk suatu rute yang berangkat pada satu depot dan berakhir di depot yang sama. Rute yang terbentuk harus memenuhi konstrain yang sada. Fungsi tujuan dari VRP adalah meminimasi total biaya transportasi. Biaya transportasi bisa terdiri dari *fix cost* dan *variable cost*. Biaya tersebut juga bisa dipengaruhi oleh beberapa hal seperti jarak perjalanan, lama perjalanan, muatan yang dibawa, jenis kendaraan, kapasitas kendaraan, jumlah kendaraan, dll. Fungsi tujuan lain yang biasanya ada pada VRP

adalah meminimasi jumlah kendaraan yang digunakan untuk melayani semua kendaraan, menyeimbangkan rute, waktu perjalanan dan beban *vehicle*, dan dapat juga biaya penalti yang mungkin terjadi. Kombinasi dari beberapa fungsi tujuan biasa digunakan dalam beberapa penelitian tentang VRP atau biasa disebut *multi objective*.

VRP memiliki sangat banyak varian. Hal ini sangat dipengaruhi oleh situasi sistem dari kasus distribusi yang akan diselesaikan. Varian dari VRP biasanya menggambarkan dari *constraint* yang digunakan. Lebih dari beberapa dekade VRP dan varianya berkembang dan semakin popular dalam literatur ilmiah (Braekers, *et.al*, 2016). Salah satu varian dari VRP adalah VRP dengan depot tunggal dan VRP dengan sejumlah depot. Dalam literatur tentang VRP terdapat 4 varian dari VRP depot tunggal yang paling penting yaitu: *Capacitated Vehicle Routing Problem* (CVRP), *Heterogeneous Fleet Vehicle Routing Problem* (HVRP), *Vehicle Routing problem with Time Windows* (VRPTW). *Vehicle Routing Problem with Simultaneous Pickup dan Delivery* (VRPSPD) (Ai, 2008).

Weise *et al.* (2010) dalam penelitiannya menjelaskan (Gambar 2.1) beberapa varian dari VRP dan hubungan antar varian. Setiap varian dibedakan dengan *constraint* yang ada, Kondisi dan situasi dari perusahaan logistik yang terkadang rumit membuat semakin banyaknya varian VRP.

Varian paling sederhana dari VRP adalah *Capacitated* VRP (CVRP). Sehingga jika hanya menyebut VRP maka VRP tersebut berasosiasi dengan CVRP. Pada CVRP semua pengiriman berdasarkan *demand* dari konsumen yang bersifat deterministik. Semua *vehicle* yang digunakan adalah identik dan berasal dari depot tunggal. Pada CVRP terdapat konstrain tunggal berupa kapasitas *vehicle*. Fungsi tujuan pada CVRP adalah meminimasi total biaya perjalanan yang muncul dalam melayani semua konsumen.



Gambar 2.1 Perbedaan dari beberapa varian VRP dan hubungannya (Weise, *et.al* 2010)

Kasus CVRP dapat digambarkan dalam sebuah grafik $G = (V, A)$, dimana $V = [0,1,2, \dots, n]$ adalah sejumlah *vertex* dan A adalah sejumlah *arc*. *Vertex* $i = 1,2, \dots, n$ diasosiasikan dengan konsumen, dengan *vertex* 0 adalah depot. Karena rute akan berakhir di depot, maka depot dapat diasosiasikan dengan *vertex* $n + 1$. Biaya perjalanan dari *vertex* i ke *vertex* j dinotasikan sebagai c_{ij}. Besar nilai c_{ij} akan berbanding lurus dengan jarak antar kedua *vertex.* Dalam kasus yang sederhana *vertex* digambarkan oleh suatu titik koordinat, sehingga jarak antara *vertex i* dan *j* dihitung dengan jarak *Euclidian*. Jika pada grafik G sebagaian atau setiap *arc*-nya memiliki arah maka hal ini dapat disebut *asymmetric* CVRP sehingga $c_{ij} \neq c_{ji}$, dan sebaliknya disebut *symmetric* CVRP. Setiap *vertex*

(konsumen) i diasosiasikan dengan *demand* yang diketahui d_i, untuk dikirim dan karena depot tidak memiliki *demand* maka $d_0 = 0$. Sejumlah K *vehicle* identik, memiliki total kapasitas C, dan berada di depot. Diasumsikan $d_i \leq C$ untuk memastikan semua *vertex* dapat terlayani. Setiap *vehicle* melakukan paling sedikit satu rute perjalanan (Toth & Vigo, 2002b).

Formulasi dari CVRP memiliki fungsi tujuan meminimalkan total biaya perjalanan dengan beberapa kendala yang harus terpenuhi. Dimisalkan x_{ij} adalah variable biner (0,1) yang akan menunjukan apakah terjadi perjalanan antara *vertex* i ke *vertex* j dan memicu munculnya biaya c_{ij}. Berikut adalah formulasi dari kasus CVRP (Toth & Vigo, 2002b):

Fungsi tujuan $$min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \quad (2.1)$$

Fungsi kendala

$$\sum_{i \in V} x_{ij} = 1 \qquad \forall j \in V \setminus \{0\}, \quad (2.1)$$

$$\sum_{j \in V} x_{ji} = 1 \qquad \forall i \in V \setminus \{0\}, \quad (2.2)$$

$$\sum_{i \in V} x_{i0} = K, \quad (2.3)$$

$$\sum_{j \in V} x_{0j} = K, \quad (2.4)$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - r(S) \qquad \forall S \subseteq V \setminus \{0\}, S \neq \emptyset, \quad (2.5)$$

$$x_{ij} \in \{0,1\} \qquad \forall i, j \in V. \quad (2.6)$$

Pada formulasi kendala (2.1) dan (2.2) memastikan bahwa setiap arc (masuk dan keluar) hanya dilewati satu kali. Kendala (2.3) dan (2.4) memastikan bahwa arc yang berasar dan menuju depot adalah sejumlah *vehicle* yang digunakan. Kendala (2.5) adalah *sub tour elimination*.

Ada banyak metode untuk menyelesaikan kasus VRP yang diperkenalkan dan dikembangkan oleh peneliti pada literatur ilmiah. VRP termasuk dalam *NP-hard problem* karena sebagai permasalahan kombinatorial, untuk mendapatkan solusi yang optimal membutuhkan waktu yang sangat lama (Kumar & Panneerselvam, 2012). Waktu komputasi untuk menyelesaikan *NP-hard problem* akan semakin tinggi secara *non deterministic polynomial* seiring dengan meningkatnya ukuran (*size*) dari masalah (Santosa & Willy, 2011). Tingkat kesulitan dari VPR akan meningkat secara eksponensial seiring dengan bertambahnya jumlah n (Braekers et al., 2016). Ada 3 metode yang digunakan dalam menyelesaikan VRP yaitu: metode *exact, heuristic* dan *hybrid* (*metaheuristic*). Metode exact seperti yang dilakukan oleh Hokama, *et al.* (2016) yang menggunakan algoritma *branch and cut* untuk menyelesaikan kasus VRP dengan *two and tri dimensional parallelepiped items*. Salah satu contoh metode *heuristic* yang digunakan adalah *packing first and route second* yang digunakan Bortfeldt (2013) dalam menyelesaikan kasus *vehicle routing and loading problem*. Metaheuristik menjadi yang paling banyak digunakan dalam menyelesaikan kasus VRP, diantaranya adalah *tabu search* (Zachariadis et al., 2009), *ant colony* (Fuellerer et al., 2009), *simulated annealing* (Leung et al., 2013), *deferential evolution* (Li & Zhang, 2015) dan *Genetic algorithm* (Miao et al., 2012). Metaheuristik menjadi popular digunakan karena metode ini memiliki kemampuan mencari solusi dengan waktu komputasi yang cepat.

2.1.1. Heterogeneous Fleet Vehicle Routing Problem (HFVRP)

HFVRP merupakan salah satu varian dari VRP. Kapasitas menjadi faktor pembeda antara HFVRP dan CVRP. Jika pada CVRP terdapat *K vehicle* yang identik, maka pada HFVRP menggunakan *vehicle* yang berbeda dengan kapasitas yang berbeda. Telah 3 tahun sejak HFVRP di perkenalkan, perkembangan yang signifikan telah dibuat dan variannya. Dalam prakteknya sebagian besar masalah distribusi, *demand* dari konsumen dilayani oleh *vehicle* dengan kapasitas yang berbeda. Penentuan keputusan jenis *vehicle* termasuk jumlah yang dibutuhkan menjadi topic yang dominan terjadi dalam bidang logistik (Jabali & Laporte, 2016).

HFVRP secara umum dapat berupa kasus dengan jumlah kendaraan yang terbatas atau tidak terbatas. Setiap *vehicle* memiliki *fix cost* yang membuat pemilihan jenis *vehicle* menjadi hal yang penting dalam mempengaruhi fungsi tujuan. Setahun terakhir menurut Jabali & Laporte (2016) sebagian riset banyak yang membahas tentang varian dari HFVRP standar, diantaranya adalah *time windows, multiple depots, external carriers, pickup and delivery operation, container loading and backhauls*. Namun demikian masih banyak potensi untuk riset pada varian HFVRP yang masih banyak. Jabali & Laporte juga menyebutkan bahwa tidak ada algoritma eksak yang digunakan dalam menyelesaikan kasus HFVRP.

HFVRP dimodelkan dengan grafik lengkap $G = (N, A)$, dimana $N = \{0,1,2 \ldots . n\}$ adalah sekumpulan *node*, dan *node* 0 adalah depot, dan $A = \{(i,j): 0 \leq i,j \leq n, i \neq j\}$ adalah sekumpulan *arc*. Konsumen di notasikan oleh $N_0 = N \backslash \{0\}$. Tetapkan c_{ij}^h adalah biaya perjalanan pada *arc* $(i,j) \in A$ oleh tipe *vehicle* h, dan tetapakan f_{ij}^h sebagai sejumlah *demand* yang dikirimkan pada *arc* $(i,j) \in A$ oleh tipe *vehicle* h. x_{ij}^h adalah variable biner yang akan bernilai 1 ketika *vehicle* $h \in H$ melakukan perjalanan pada *arc* $(i,j) \in A$. Setiap konsumen i memiliki sejumlah *demand* q_i. Tetapkan $H = \{1,2, \ldots . k\}$ sebagai tipe *vehicle* yang tersedia, dan t^hdan Q_h adalah *fix vehicle cost* dan kapasitas untuk *vehicle* tipe $h \in H$, dan m_h adalah jumlah *vehicle* tipe h yang tersedia.

Berikut adalah formulasi dari *single-commodity flow* untuk HFVRP berdasarkan Baldacci, *et.al* (2008) :

$$\textbf{\textit{Fungsi Tujuan}} \quad min \sum_{h \in H} \sum_{j \in N_o} t^h x_{0j}^h + \sum_{h \in H} \sum_{(i,j) \in A} c_{ij}^h \, x_{ij}^h \tag{2.7}$$

Fungsi Kendala

$$\sum_{j \in N_0} x_{0j}^h \leq m_h \qquad h \in H \tag{2.8}$$

$$\sum_{h \in H} \sum_{j \in N} x_{ij}^h = 1 \qquad i \in N_0 \tag{2.9}$$

$$\sum_{h\in H}\sum_{i\in N} x_{ij}^{h} = 1 \qquad j \in N_0 \tag{2.10}$$

$$\sum_{h\in H}\sum_{j\in N} f_{ji}^{h} - \sum_{h\in H}\sum_{j\in N} f_{ij}^{h} = q_i \qquad i \in N_0 \tag{2.11}$$

$$q_i x_{ij}^{h} \le f_{ij}^{h} \le (Q_h - q_i) x_{ij}^{h} \qquad (i,j) \in A, h \in H \tag{2.12}$$

$$x_{ij}^{h} \in \{0,1\} \qquad (i,j) \in A, h \in H \tag{2,13}$$

$$f_{ij}^{h} \ge 0 \qquad (i,j) \in A, h \in H \tag{2,14}$$

Pada formula diatas, fungsi tujuan yang ingin dicapai adalah minimasi jumlah total *vehicle fix cost* dan total biaya perjalanan. Batasan maksimal jumlah kendaraan yang tersedia untuk masing-masing tipe *vehicle* ada pada konstrain (2.8). Pada kasus dengan jumlah *vehicle* yang tidak terbatas untuk masing-masing tipe maka konstrain (2.8) menjadi *redundant*. Konstrain (2.9) dan (2.10) digunakan untuk memastikan bahwa setiap konsumen hanya dikunjungi satu kali. Konstrain (2.11) menunjukan aliran barang/ komoditas. Konstraint (2.13) dan (2.14) merupakan memastikan variabel yang terbentuk merupakan variabel interger dan tidak negatif.

2.2 *Routing Problem with Loading Constraint*

Terdapat dua kasus pada bidang transportasi dan logistik yaitu membentuk rute untuk kendaraan dan memasukan muatan ke dalamnya. Optimasi untuk keduanya termasuk dalam *NP-hard*, yang secara praktis sangat sulit untuk diselesaikan. Sehingga dulu pada umumnya dua kasus ini diselesaikan dengan cara terpisah, hingga pada beberapa tahun terakhir mulai muncul algoritma untuk mengombinasikan dua problem tersebut. Menggabungkan dua problem tersebut membuat semakin sulit untuk diselesaikan, namun hal ini memungkinkan menghasilkan solusi yang lebih baik berkaitan dengan target dalam logistik (Iori & Martello, 2010).

Dalam CVRP, *demand* konsumen direpresentasikan sebagai total berat dari barang yang akan dikirimkan. Hal tersebut tidak selalu terjadi di dunia nyata. *Demand* dari konsumen tidak hanya dilihat dari beratnya, namun juga dari bentuknya. Bentuk dari barang nantinya juga akan berpengaruh dengan proses

bongkar muat dan *material handling*, sehingga terdapat dua hal yang harus diperhatikan. Pertama adalah semua barang dapat dialokasikan secara feasible masuk ke dalam *vehicle*. Kedua adalah fakta bahwa proses *unloading* harus dapat dilakukan tanpa merubah susunan dari barang lainnya, sehingga ini akan mempengaruhi urutan dan posisi dari barang tersebut dimasukan ke dalam *vehicle*.

Pada beberapa aplikasi transportasi dimungkinkan mengangkut suatu barang berbentuk persegi panjang yang dalam penyusunannya tidak mempertimbangkan tingginya dan dalam penyusunannya barang tersebut tidak dapat ditumpuk oleh barang lain. Hal in dapat disebabkan oleh sifat barang tersebut yang mudah pecah, beratnya yang membuatnya seperti itu atau bentuk dari benda tersebut. Contohnya adalah ketika pengiriman suatu perangkat dapur seperti kulkas, meja kaca, mesin CNC, dll. Hal ini harus dipertimbangkan dalam CVRP, sehingga jika pada bentuk klasik dari CVRP hanya menggunakan konstrain berat, maka berdasarkan pembahasan diatas perlu dipertimbangkan aspek dua dimensi dari muatan (*two-dimensional loading*). Pada kondisi lain, *demand* dari konsumen dapat berupa kubus atau balok yang memiliki 3 dimensi dengan berat tertentu.

Penelitian tentang kasus seperti yang dijelaskan diatas memiliki relasi dengan *multi-dimensional packing problem,* yang merupakan perkembangan dari bentuk klasik *bin packing*. Banyak peneliti yang melakukan penelitian pada area ini seperti Zachariadis *et al.* (2009), Männel & Bortfeldt (2016) Fernández, *et al.* (2013), Junqueira & Morabito (2015), Alonso, *et al.* (2016), dll. Banyak penelitian yang membahas tentang optimasi kombinatorial *multi-dimensional packing problem*. Dari berbagai macam varian dari kasus *multi-dimensional packing problem*, pada dasarnya mengacu pada 4 kasus dibawah ini (Iori & Martello, 2010):

- *Two-Dimensional Bin Packing Problem* (2PBB): Pada kasus ini semua muatan berbentuk persegi panjang dimasukan ke dalam sejumlah kontak berbentuk persegi panjang yang identik, Pendekatan eksak untuk 2BPP pada umumnya didasarkan pada teknik *branch and bound* dan dapat menghasilkan solusi optimal hingga 100 unit muatan.
- *Two-Dimensional Strip Packing Problem* (2SPP): Muatan yang berbentuk persegi panjang dimasukan ke dalam kontainer terbuka dengan lebar tertentu

dan panjang yang tidak terbatas. Sehingga fungsi tujuan dari kasus ini adalah meminimalkan panjang kontainer yang dibutuhkan untuk dapat memuat semua muatan yang ada. Terdapat banyak pendekatan heuristik dan metaheuristik yang sudah teruji menyelesaikan kasus 2BPP dan 2SPP.

- *Three-Dimensional Bin Packing Problem* (3BPP): Muatan berbentuk balok dimasukan ke dalam kontainer 3 dimensi dengan ukuran yang identik. Dari pendekatan eksak, heuristik, hingga metaheuristik sudah pernah digunakan dalam menyelesaikan kasus ini.
- *Three-Dimensional Strip Packing Problem* (3SPP): Muatan dalam kasus ini berbentuk balok yang akan dimasukan ke dalam kontainer 3 dimensi dengan panjang yang tidak terbatas. Fungsi tujuan dari kasus ini adalah meminimalkan total panjang dari kontainer.

2.2.1. Capacitated vehicle routing problem with two-dimensional loading constraint (2L-CVRP)

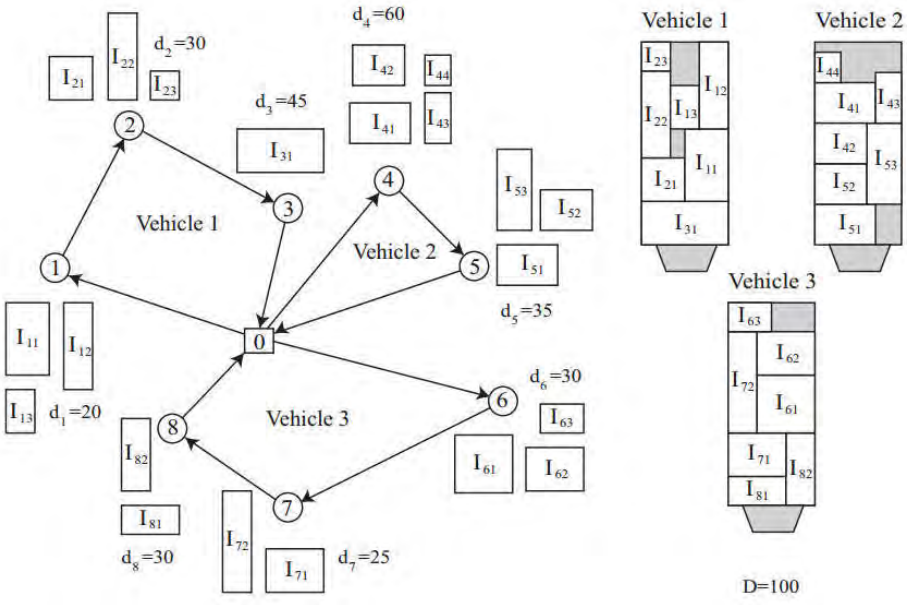
2L-CVRP memiliki deskripsi hampir sama dengan bentuk normal CVRP. Yang membedakan adalah atribut pada *vehicle* dan *demand* dari konsumen. Pada 2L-CVRP terdapat K *vehicle* yang identik, dengan kapasitas berat D dan kapasitas area muatan berbentuk persegi panjang dengan lebar W dan panjang H. *Demand* konsumen i $(i = 1,2, \dots, n)$ terdiri dari m_i unit dengan total berat d_i. Setiap muatan $I_{i\ell}$ $(\ell = 1,2, \dots ., m_i)$ memiliki lebar $w_{i\ell}$ dan panjang $h_{i\ell}$. Pada kondisi normal semua muatan memiliki orientasi tetap.

Ketika *vehicle* k ditugaskan untuk menjalankan suatu rute yang melayani sejumlah konsumen $S(k) \subseteq \{1,2, \dots, n\}$, dua konstrain harus dipenuhi :

- Total berat $\sum_{i \in S(k)} d_i$ tidak melebihi dari kapasitas *vehicle* D_i.
- Dipastikan bahwa penyusunan muatan *feasible* untuk semua muatan yang merupakan *demand* dari konsumen $S(k)$ ke dalam $W \times H$ luasan kontainer.

Selain mencari rute dengan total biaya yang minimum, 2L-CVRP juga mencari cara agar semua muatan diangkut dengan tidak lebih dari K.

Kasus diatas juga dikenal dengan *unrestricted* 2L-CVRP. Kasus lain dapat terjadi ketika *demand* dari konsumen berupa barang yang sangat berat, ukuran yang sangat besar atau mudah pecah, sehingga menjadi tidak mungkin untuk melakukan perubahan posisi di dalam kontainer. Hal ini disebut sebagai *Sequential* 2L-CVRP. Pada kasus *sequential* 2L-CVRP semua muatan akan disusun secara LIFO (*Last in First Out*). Semua muatan untuk konsumen yang akan dikunjungi terakhir akan dimasukan terlebih dahulu agar peletakannya berada di bagian paling dalam di kontainer, dan sebaliknya. Sehingga muatan konsumen yang akan dikunjungi lebih awal dapat dikeluarkan dengan mudah tanpa memindahkan mutan lain. Gambar 2.2 sudah menunjukkan bagaimana *sequential* 2L-CVRP diselesaikan.



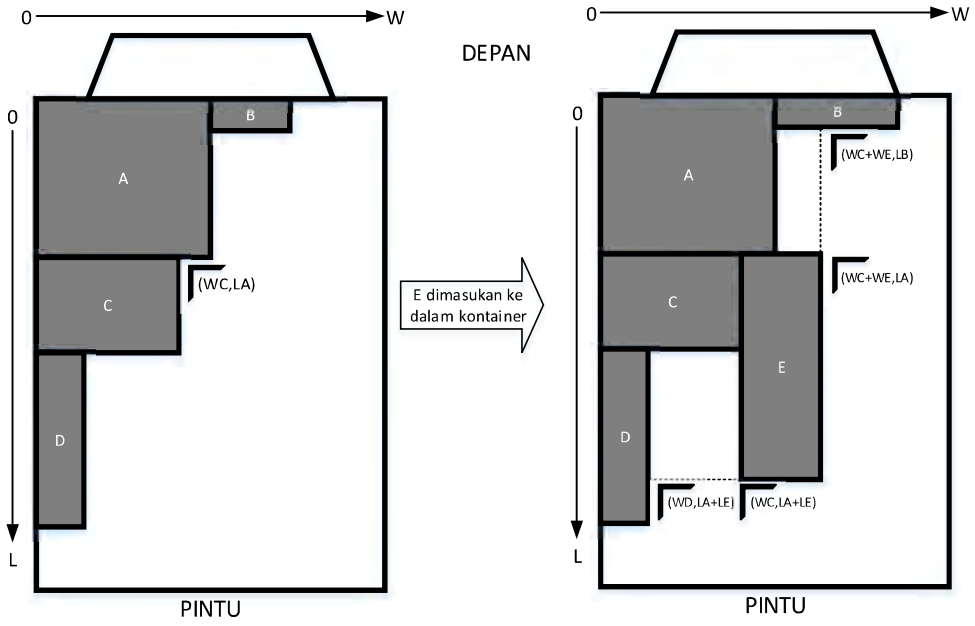
Gambar 2.2 Ilustrasi dari 2L-CVRP (Iori & Martello, 2010).

Varian dari 2L-CVRP dapat berupa *Heterogeneous Fleet* 2L-VRP atau dapat disebut 2L-HFVRP. Seperti pada penelitian Leung *et al.* (2013) yang menggunakan algoritma *Simulated Annealing with heuristic local search* (SA_HLS) untuk menyelesaikan 2L-HFVRP. Dalam penelitiannya Leung *et al.* (2013) mencari solusi untuk dua skenario yaitu *unrestricted* dan *sequential*. Dengan

4 jenis *vehicle* yang berbeda dan jumlah yang tidak terbatas, memiliki jenis kendaraan dan jumlahnya menjadi salah satu variabel keputusan di penelitiannya.

2.2.2 Packing Heuristic

Seperti yang sudah disebutkan diatas, bahwa pada 2L-CVRP setiap rute harus *feasible* terhadap penyusunan muatan ke dalam kontainer *vehicle* yang digunakan. Zachariadis *et al.* (2009) dalam penelitian merancang 5 *packing heuristic* yang dikumpulkannya dari beberapa penelitian sebelumnya untuk menyelesaikan 2L-CVRP. Pada *sequential* 2L-CVRP urutan memasukan muatan ke kontainer adalah urutan terbalik dari kunjungan konsumen. Sedangkan pada kasus *unrestricted* urutan memasukan muatan ke kontainer diurutkan berdasarkan dimensi yang terbesar.



Gambar 2.3 Ilustrasi memasukan muatan kedalam kontainer (Zachariadis et al., 2009) .

Dimisalkan *posList* sebagai daftar posisi peletakan yang mungkin untuk muatan yang akan dimasukan. Pada awalnya hanya ada satu posisi peletakan yaitu pada sudut kiri depan, sehingga *posList* = {(0, 0)}. Dengan dimasukkannya muatan

maka *posList* yang dipilih akan dihapus dan akan muncul *posList* baru. Gambar 2.3 menggambarkan mekanisme dalam memasukan muatan ke kontainer. Kotak E akan dimasukan ke *posList* (WC, LA).Ketika E dimasukan maka *posList* tersebut akan dihapus dan digantikan oleh *posList* yang baru. Sehingga *posList* diperbaharui menjadi *posList*=$(posList - \{(WC, LA)\}) \cup \{(WC, LA + LE), \ (WC + WE, LA), \ (WD, LA + LE), \ (WC + WE, LB)\}$. Jika terdapat *posList* yang sama maka salah satu akan dihapus. Posisi peletakan yang dipilih harus dari posisi yang tersedia di *posList* dan tidak melanggar konstrain (*overlapping* atau *sequential*).

Posisi peletakan muatan di kontainer ditentukan dengan *packing heuristic*. Zachariadis *et al.* (2009) pada penelitiannya menggabungkan 5 *packing heuristic* dari beberapa penelitian sebelumnya. Berikut adalah 5 *packing heuristic* yang digunakan dalam memilih posisi peletakan muatan di kontainer:

- ***Heur1:*** *Bottom-Left Fill (W-axis)* (Chazelle, 1983)

  Dari daftar posisi yang tersedia di *posList,* posisi yang dipilih adalah posisi dengan nilai sumbu W paling kecil. Gambar 2.4 sebelah kiri menggambarkan *Heur1. PosList1* dipilih karena berada di koordinat dengan nilai sumbu W paling kecil.

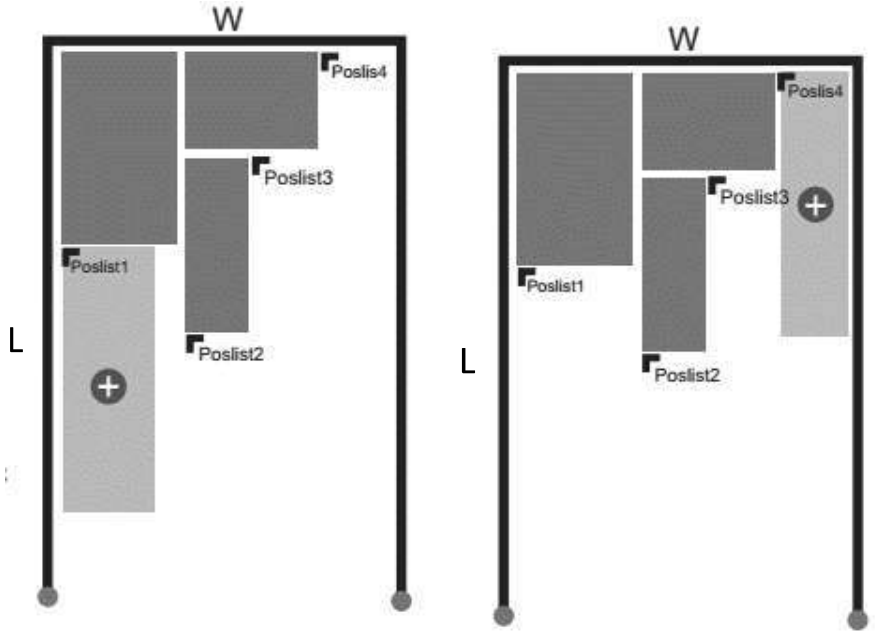
- ***Heur2:*** *Bottom-Left Fill (L-axis)* (Chazelle, 1983)

  Dari daftar posisi yang tersedia di *posList* posisi yang dipilih adalah posisi dengan nilai sumbu L paling kecil. Gambar 2.4 sebelah kanan menggambarkan *Heur2. PosList4* dipilih karena berada di koordinat dengan nilai sumbu L paling kecil.

- ***Heur3:*** *Max Touching Perimeter heuristic* (Lodi, *et al.,*1999)

  Untuk setiap posisi yang tersedia di *posList,* hitung total *touching perimeter* dari masing-masing posisi ketika muatan diletakkan di posisi tersebut. *Touching perimeter* adalah jumlah panjang dari sisi muatan yang menempel pada sisi muatan lain dan dinding kontainer. Posisi di *posList* dengan total *touching perimeter* terbesar akan dipilih sebagai lokasi peletakan muatan. Gambar 2.5

paling kanan menggambarkan *Heur3*. *PosList4* dipilih karena memiliki total *touching perimeter* yang paling besar.



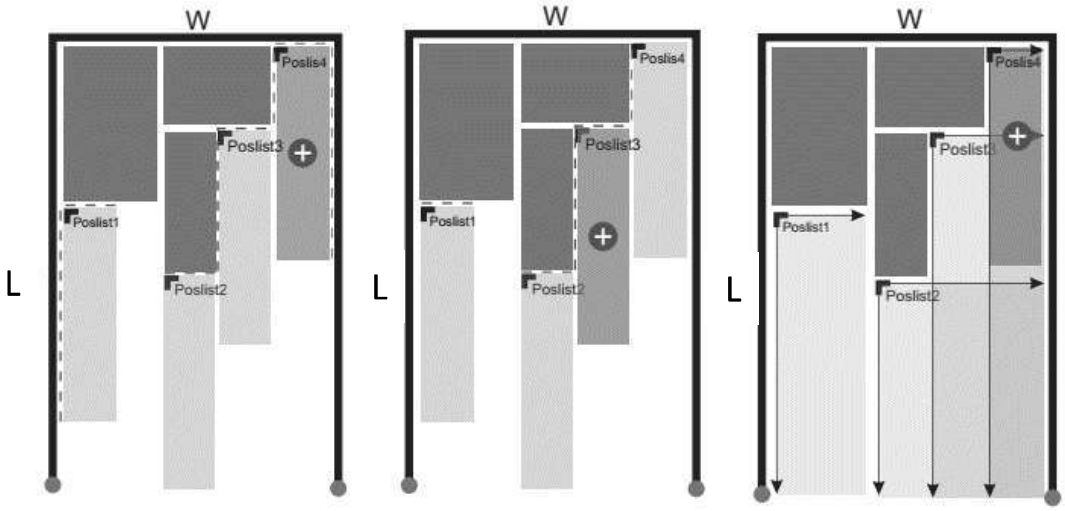
Gambar2.4 Ilustrasi *Heur1* dan *Heur2*

- ***Heur4:*** *Max Touching Perimeter No Walls heuristic* (Lodi, *et al.,*1999)

Heur4 juga menggunakan *touching perimeter*, namun yang membedakannya adalah pada *heur4* sisi yang menempel pada dinding kontainer tidak dihitung. Gambaran dari *heur4* dapat dilihat dari Gambar 2.5 pada gambar kedua. Pada Gambar 2.5.dipilih *posList3* bukan *posList 4* seperti di *heur3* karena pada *posList4* sisi dari muatan yang menempel di dinding kontainer tidak dihitung yang membuat *posList3* memiliki total *touching perimeter* yang lebih besar.

- ***Heur5:*** *Min Area heuristic* (Zachariadis et al., 2009)

Untuk setiap posisi yang tersedia di *posList,* luas permukaan yang berbentuk persegi panjang dihitung. Seperti pada Gambar2.5 paling kanan, bagaimana luasan permukaan dari *posList4* adalah yang paling kecil. Sehingga *posList4* yang dipilih untuk meletakan muatan. Heuristik ini bertujuan untuk mencapai utilitas yang tinggi dari luasan permukaan yang tersedia.



Gambar2.5 Ilustrasi *Heur3, Heur4* dan *Heur5*

Dari 5 heuristik yang dijelaskan diatas diurutkan dari *heur1* sampai *heur5*. Setiap proses memasukan muatan maka yang akan dijalankan adalah dimulai dari heuristic yang paling sederhana yaitu *heur1*. Jika *heur1* gagal menghasilkan solusi yang *feasible* maka akan dijalankan *heur2* dan seterusnya hingga *heur5.* Jika masih tidak menemukan solusi yang *feasible* maka ubah rute yang terbentuk. Menerapkan heuristik dengan urutan dari yang paling sederhana akan mengurangi waktu komputasi.

2.3 *Metaheuristik*

Metaheuristik adalah suatu pendekatan dalam optimasi yang dikembangkan dalam rangka menyelesaikan kasus optimasi yang tidak bisa diselesaikan dengan teknik kalkulus diferensial. Pendekatan ini biasanya di inspirasi dari berbagai fenomena alam seperti: fenomena evolusi, perilaku hewan, proses pendinginan baja, dll (Santosa & Willy, 2011).

Metaheuristik biasa digunakan dalam menyelesaikan kasus *hard combinatorial* dengan skala yang besar. Hal ini dikarenakan sebagai pendekatan

pencarian solusi dengan memperbaiki hasil setiap iterasinya membuat metaheuristic menjadi memiliki waktu komputasi yang lebih cepat dibanding metode lainnya. Metaheuristik mencoba mencari solusi yang optimal atau mendekati optimal. Pengguna pendekatan metaheuristic biasanya lebih mengutamakan waktu komputasi dibandingkan hasil yang optimal (*global optimum*). Karena untuk kasus kombinatorial mencari solusi yang optimal membutuhkan waktu komputasi yang sangat lama. Metaheuristik dapat menyelesaikan kasus kombinatorial dengan waktu yang relatif cepat dengan hasil yang mendekati optimal. Dalam dunia nyata, terkadang diperlukan waktu yang cepat dalam proses pengambilan keputusan, sehingga menunggu hasil optimum bukan merupakan pilihan yang *feasible*.

Dalam penerapannya dimungkinkan untuk melakukan kombinasi/*hybrid* dari antar metode. Hal ini dilakukan untuk menutupi kekurangan dari suatu metode dengan metode lain, dan sebaliknya. Penelitian ini mencoba untuk melakukan kombinasi antar 2 metode yaitu *Cross Entropy* (CE) dan *Genetic Algorithm* (GA).

2.3.1 Cross Entropy (CE)

Menurut Rubinstein (1997) metode CE pada awalnya digunakan untuk mengestimasi probabilitas dari kejadian langka (*rare event*) dengan penerapan algoritma adaptive pada peristiwa stokastik yang kompleks, dengan meminimasi variasi (Santosa & Willy, 2011). Pada dasarnya dalam algoritma CE membangkitkan sejumlah sampel dan pada iterasi berikutnya dibangkitkan sampel random dengan menggunakan parameter yang dihasilkan dari iterasi sebelumnya. Algoritma CE dijalankan dengan prosedur iterasi, dimana setiap iterasi nya dibagi menjadi dua bagian, yaitu:

- Membangkitkan sampel random, dengan menggunakan mekanisme atau distribusi tertentu.
- Memperbaharui parameter, parameter diambil dari sampel elit untuk digunakan dalam membangkitkan bilangan random di iterasi berikutnya.

Sampel elit adalah berapa persen dari sampel yang digunakan untuk memperbaiki dan mengupdate parameter yang akan digunakan.

Algoritma CE tidak hanya bisa digunakan untuk optimasi kontinyu namun juga bisa digunakan untuk optimasi diskret (kombinatorial) seperti penjadwalan, TSP ,VRP, dll. Untuk itu perlu dilakukan modifikasi terhadap algoritma CE agar sesuai dengan format permasalahan tersebut. Agar algoritma CE dapat diaplikasikan pada kasus TSP atau VRP perlu ditentukan beberapa langkah berikut:

- Bagaimana cara membangkitkan rute/tour dengan random.
- Bagaimana memperbaharui parameter pada setiap iterasi.

Penerapan CE pada kasus TSP atau VRP, menggunakan matriks transisi P sebagai parameter, yang didalamnya terdapat bilangan random yang menunjukan probabilitas munculnya perjalanan dari titik i ke titik j. Sehingga setiap iterasinya rute yang terbentuk akan semakin menuju ke rute yang terpendek. Diagonal dari P bernilai 0, karena peluang mengunjungi dari titik i ke titik i adalah 0. Jumlah dari setiap baris dari P adalah 1. Algoritma 1.1 menjelaskan bagaimana cara membangkitkan rute dengan menggunakan matriks transisi.

Algoritma 1.1 Algoritma membangkitkan rute (Santosa & Willy, 2011).

1. Tentukan $P^{(1)} = P$ dan $X_1 = 1$, set k=1
2. Dapatkan $P^{(k+1)}$ dari $P^{(k)}$ dengan cara membuat kolom ke X_k dari $P^{(k)}$ sama dengan 0 dan normalisasi semua baris sehingga jumlah total peluangnya 1. Bangkitkan X_{k+1} dari distribusi yang di bentuk oleh baris ke X_k dari matrik $P^{(k+1)}$.
3. Jika $k = n - 1$ berhenti atau set $k = k + 1$ dan ulangi langkah 2
4. Evaluasi panjang dari rute yang terbentuk

Agar kota yang sudah dikunjungi X_k tidak dikunjungi lagi, maka peluang dari X_k setelah dikunjungi harus 0. Sehingga peluang pada kolom X_k, dibuat 0. Setelah diperoleh rute maka hitung jarak dari rute yang terbentuk. Urutkan nilai-nilai jarak total mulai dari yang terbesar. Dari N rute akan dipilih ρ 100% yang menghasilkan jarak terpendek untuk menjadi sampel elit. Sampel elit tersebut digunakan akan digunakan untuk memperbaharui probabilitas transisi pada iterasi berikutnya. Algoritma 1.2 menjelaskan langkah-langkah CE untuk menyelesaikan kasus TSP.

Algoritma 1.2 Algoritma CE untuk kasus TSP (Santosa & Willy, 2011).

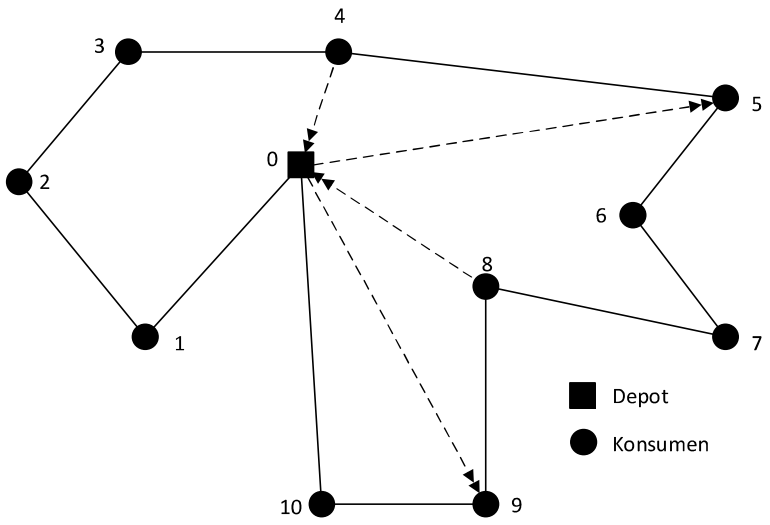
1. Pilih matriks transisi $\widehat{P_0}$, kecuali pada diagonal bernilai $\frac{1}{(n-1)}$

2. Bangkitkan sampel rute $X_1, \ldots, X_N$ dengan algoritma sebelumnya
3. Hitung fungsi tujuan $S(x)$
4. Urutkan dari yang terbesar $S(X_N) \ldots S(X_1)$
5. Update nilai P dengan sampel elit menggunakan rumus:

$$P^{k+1} = \alpha\omega + (1-\alpha)P^k \quad (2.15)$$

Pada rumus (2.15), ω adalah probabilitas transisi yang dihitung dari sampel elit dan P^k adalah probabilitas transisi iterasi sebelumnya. Nilai α $(0{,}5 \leq \alpha \leq 1)$ merupakan porsi dari probabilitas sampel elit (ω) dan probabilitas sebelumnya (P^k) yang digunakan dalam mencari P baru.

Penggunaan algoritma CE untuk kasus TSP hampir sama jika digunakan untuk kasus VRP. Pada kasus VRP terdapat K vehicle yang memiliki kapasitas tertentu. Sehingga VRP juga dapat disebut sebagai K-TSP, terdapat sejumlah K kasus TSP dalam suatu VRP.



Gambar2.6 Ilustrasi rute dalam CVRP (Santosa & Willy, 2011)

Gambar 2.6 menjelaskan konsep VRP sebagai K-TSP. Pada Gambar 2.6 ruter berawal dari titik 0 dan berjalan hingga titik 4 harus kembali ke 0 karena sudah melewati batas kapasitas, dan melanjutkan jalan dari titik 0 ke titik 5 dan seterusnya

hingga semua titik terlayani. Sehingga hampir sama dengan TSP saat CE digunakan untuk menyelesaikan VRP maka pembangkitan sampel rute perlu diperhatikan kendala berupa kapasitas kendaraan. Jika akan melakukan random untuk titik berikutnya maka perlu dievaluasi apakah kapasitas sudah terlewati, jika sudah maka akan kembali ke depot dan lakukan pembangkitan bilangan random untuk titik sesudah depot.

Beberapa peneliti menggunakan pendekatan CE untuk menyelesaikan kasus VRP dan variannya. VRP dengan *soft time windows* diselesaikan dengan *multi-objective* CE oleh Hauman & Bekker (2014). Beberapa perbandingan dilakukan dan menunjukkan solusi yang dihasilkan menuju ke titik tunggal area objektif. VRP dengan *stochastic demand and simultaneous delivery and pickup* pernah diselesaikan oleh Wang & Qiu (2012) dengan skema yang didasarkan pada algoritma CE. Eksperimen numerical menghasilkan solusi yang mengindikasikan bahwa metode ini dapat menyelesaikan masalah tersebut dengan efektif. Algorima CE juga digunakan oleh Cabo & Edgar (2011) untuk menyelesaikan kasus *classical* VRP. Walaupun memakan waktu namun algoritma CE dapat menghasilkan solusi yang baik.

2.3.2 *Genetic Algorithm* (GA)

Algoritma GA merupakan algoritma yang didasarkan pada prinsip-printip genetika dan seleksi alam. Elemen-elemen yang ada pada genetika dan seleksi alam diadopsi dalam GA untuk menyelesaikan permasalahan optimasi. Elemen tersebu adalah : reproduksi, kawin silang, mutasi dan elitisme. Selain mampu menyelesaikan permasalahan optimasi kontinyu, GA juga banyak dipakai dalam menyelesaikan masalah-masalah kombinatorial seperti TSP, VRP,*crew scheduling* untuk *airline* hingga permasalahan *control.* Dalam prosedur GA prosedur pencarian solusi hanya didasarkan pada nilai fungsi tujuan, tidak menggunakan *gradient* atau teknik kalkulus. (Santosa & Willy, 2011).

Algoritma GA menggunakan beberapa mekanisme yang terinspirasi dari prinsip-prinsip genetika, berikut adalah istilah-istilah yang ada dalam algoritma GA yang :

- **Kromosom**

  Kromosom dalam GA adalah individu yang mewakili suatu solusi. Dalam beberapa kasus kromosom dapat langsung mewakili solusi yang diinginkan, namun jika tidak maka diperlukan proses pengkodean (*encoding* dan *decoding*). Pengkodean dilakukan untuk menerjemahkan struktur data yang bisa diolah oleh GA menjadi solusi yang sesuai dengan permasalahan (*encoding*) dan sebaliknya (*decoding*). Langkah pertama dalam GA adalah membangkitkan populasi yang merupakan sekumpulan dari kromosom. Setiap kromosom disusun oleh gen-gen yang merupakan elemen dari vektor solusi.

- ***Fitness***

  Fungsi *fitness* digunakan untuk mengukur tingkat kualitas dari solusi yang didasarkan pada fungsi tujuan yang digunakan. Setiap solusi yang dihasilkan akan dievaluasi dengan fungsi *fitness*. Fungsi *fitness* bisa berkaitan langsung dengan fungsi tujuan atau bisa juga sedikit modifikasi terhadap fungsi tujuan. *Output* dari fungsi *fitness* akan digunakan untuk memilih kromosom elit.

- **Elitisme**

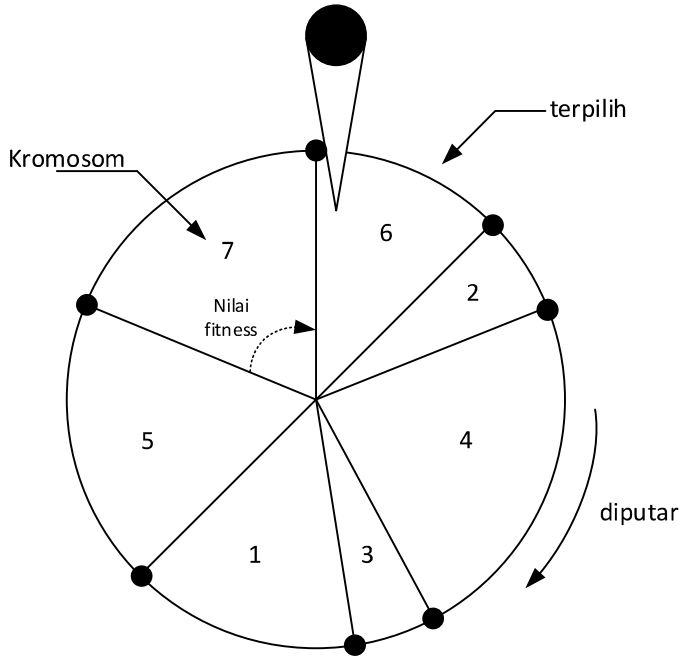
  Elitisme (*elitism*) dalam GA adalah konsep untuk menjaga agar individu-individu terbaik yang muncul di suatu generasi akan tetap ada pada generasi selanjutnya. Konsep ini dapat dilakukan dengan berbagai cara, namun cara paling sederhana adalah dengan melakukan penyalinan individu terbaik atau dapat juga melalui kombinasi antara solusi-solusi turunan.

- ***Crossover* atau Kawin silang**

  Crossover atau sering disebut kawin silang adalah usaha untuk menghasilkan individu (anak) dari kombinasi antar dua individu (induk). Ada dua macam *crossover* yang bisa diterapkan dalam GA, yaitu *crossover* sederhana dan *crossover* aritmatik. Dalam proses kawin silang, terdapat parameter yang disebut probabilitas kawin silang. Parameter ini akan menentukan banyak sedikitnya kromosom yang akan mengalami kawin silang. Contoh jika parameter bernilai 0,5 maka ada sekitar separuh populasi yang akan mengalami kawin silang.

- ***Selection***

  Dalam proses *crossover* akan dipilih kromosom atau individu yang akan menjadi induk. Proses seleksi ini yang akan menentukan kromosom mana yang akan menjadi induk. Proses seleksi menggunakan konsep roda lotre. Roda lotre akan menjadi area pemilihan kromosom. Kromosom yang memiliki nilai fitness yang lebih baik akan memiliki peluang terpilih yang lebih besar. Gambar 2.7 dapat dilihat bahwa kromosom dengan nilai *fitness* besar akan memiliki luasan yang lebih besar sehingga kemungkinan yang lebih besar dibandingkan yang lain.

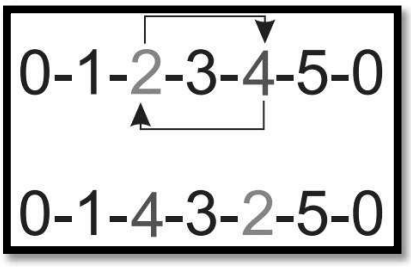


Gambar2.7 Ilustrasi roda lotre dalam proses seleksi

- **Mutasi**

  Pada setiap kromosom terdiri dari sekumpulan gen yang merupakan komponen dari vektor solusi. Mutasi adalah proses menukar posisi gen dalam kromosom untuk mendapat kromosom baru. Gen yang akan dipilih untuk diubah urutannya atau dimutasi akan dipilih secara random. Parameter yang digunakan dalam mutasi adalah probabilitas mutasi. Probabilitas ini akan menentukan

kromosom mana yang akan mengalami mutasi. Semakin besar nilai probabilitas mutasi maka akan semakin banyak kromosom dalam populasi yang akan mengalami mutasi. Jika nilai probabilitas mutasi 0.01 maka akan ada sekitar 1% dari seluruh kromosom dalam populasi yang akan mengalami mutasi. Gambar 2.8 menggambarkan mekanisme mutasi untuk kasus kombinatorial. Gen 2 dan 4 bertukar posisi sehingga terbentuk kromosom baru dengan urutan gen yang berbeda.



Gambar2.8 Ilustrasi mutasi pada kasus kombinatorial

Setelah dijelaskan beberapa istilah yang dipakai dalam algoritma GA, maka pada Algoritma 1.3 akan dijelaskan langkah-langkah dasar dari GA.

Algoritma 1.3 Algoritma dasar GA (Santosa & Willy, 2011).

1. Pembangkitan populasi awal (solusi awal): Tentukan ukuran populasi dan bangkitkan populasi awal secara random, evaluasi nilai setiap individu menggunakan fungsi *fitness.* Tentukan probabilitas kawin silang dan probabilitas mutasi. Pembangkitan populasi awal juga dapat menggunakan mekanisme tertentu.
2. Set iterasi $t = 1$.
3. Pilih individu terbaik untuk menjadi individu elit. Individu elit akan disalin sejumlah tertentu untuk mengganti individu lain.
4. Lakukan mekanisme seleksi untuk memilki kromosom dalam populasi untuk menjadi induk dalam proses *crossover.*
5. Lakukan proses *crossover* antar induk yang terpilih.

6. Menggunakan probabilitas mutasi, lakukan mutasi untuk beberapa individu dalam populasi.
7. Iterasi $t = t + 1$ jika belum mencapai konvergensi dan berhenti jika sebaliknya.
8. Kembali ke langkah 2.

2.3.3 *Cross Entropy Genetic Algorithm* (CEGA)

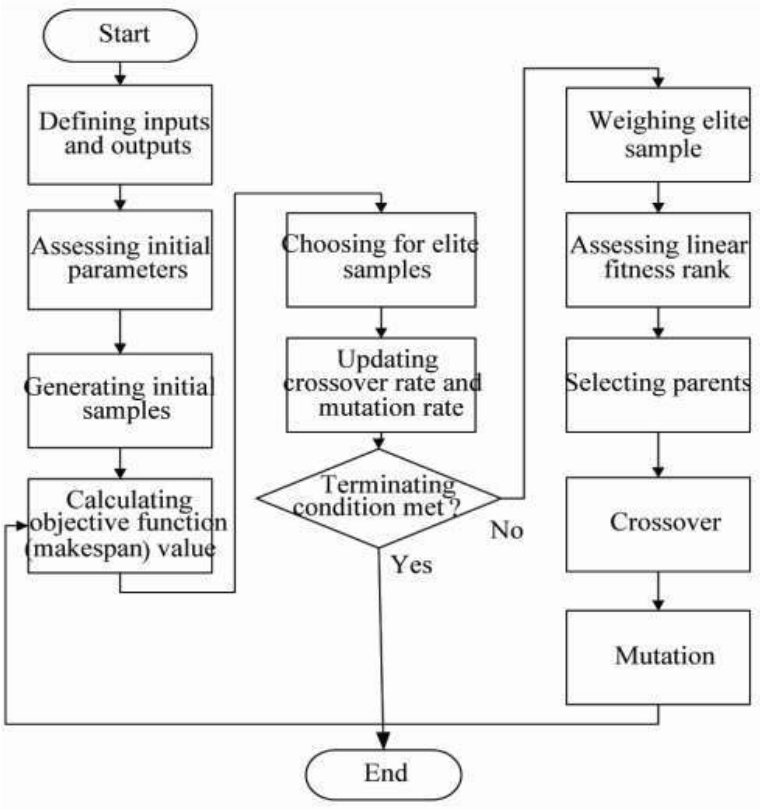
CEGA adalah algoritma hasil gabungan dari konsep *cross entropy* dan *genetic algorithm*. Algoritma CE digunakan sebagai dasar dari algoritma dengan mengadopsi mekanisme mutasi dan *cross over* pada algoritma GA. Tujuan dari penggabungan ini adalah mengeliminasi kelemahan dari masing-masing algoritma dengan kelebihan yang dimiliki masing-masing algoritma. Algoritma CE memiliki kecenderungan untuk terjebak di lokal optimum jika dijalankan sendiri. Algoritma CE memiliki kecenderungan itu jika memiliki *learning rat* yang tinggi. Sehingga CE digunakan untuk menutupi kekurangan eksploitasi solusi dari GA dan GA berperan untuk mengeksplorasi area solusi. Selain itu algoritma CE membutuhkan waktu yang lama untuk menyelesaikan kasus kombinatorial dengan skala besar. Dengan mengadopsi mekanisme mutasi dan *cross over* maka proses pencarian solusi akan lebih singkat (Santosa dkk., 2011).

Algoritma CEGA pertama kali dikembangkan oleh Santosa dkk. (2011) untuk kasus penjadwalan. Beberapa penelitian pernah melakukan pengujian algoritma CEGA untuk menyelesaikan beberapa kasus optimasi. Beberapa kasus adalah kasus logistik dan rantai pasok (Aini, 2012) (Lahdji, 2016) (Santosa, *et al.*, 2016). Algoritma 1.4 menjelaskan algoritma dasar dari CEGA.

Algoritma 1.4 Algoritma dasar CEGA

1. Penentuan parameter awal
2. Bangkitkan solusi awal
3. iterasi=1
4. Evaluasi fungsi tujuan
5. Pemilihan Sampel elit
6. Perbaharui parameter mutasi

7. Lakukan mutasi dan *crossover*. Kembali ke langkah 4 jika *stopping criteria* belum terpenuhi. (iterasi=iterasi+1)
8. .Jika *stopping criteria* sudah terpenuhi maka hentikan iterasi.



Gambar 2.9 Diagram alir CEGA (Santosa dkk., 2011)

Pada Gambar 2.9 menjelaskan diagram alir dari algoritma CEGA. Berikut adalah penjelasan dari beberapa proses yang dituliskan pada diagram alir CEGA.

Assessing Initial Parameters

Pada algoritma CEGA digunakan beberapa parameter yang ditentukan oleh pengguna. Parameter tersebut adalah:

- Ukuran populasi N, tidak ada batasan yang tertentu tentang ukuran populasi. Semakin besar skala dari permasalahan semakin besar pula jumlah populasi yang dibutuhkan.
- Rasio sampel elit ρ, biasanya direntang 1% - 10%.
- Koefisien *smoothing* α, berada di rentang 0 -1, dan secara empiris berada di rentang 0.4 – 0.9.
- *Crossover rate* (Pps).
- *Terminating criterion* $\varepsilon = 0.001$.

Pembangkitan sampel

1) **Pembobotan sampel elit**

   Pembobotan ini akan digunakan pada proses penentuan orang tua. Pilih orang tua pertama dari sampel elit berdasarkan bobot dari setiap sampel elit. Jika urutan yang terbentuk menghasilkan nilai fungsi tujuan yang lebih baik dibandingkan urutan terbaik sebelumnya maka bobot dari urutan tersebut adalah sama dengan jumlah dari sampel elit, selain itu bobot bernilai 1.

2) **Menghitung *Linear Fitness Rank***

   Linear Fitness Rank (LFR) pada setiap iterasi dihitung dari *fitness value* dari semua sampel yang ada di iterasi sebelumnya. Nilai LFR dihitung dengan persamaan (2.16).

$$LFR\big(I(N-1+1)\big) = F_{max} - (F_{max} - F_{min})\left(\frac{i-1}{N-1}\right) \qquad (2.16)$$

   Dengan nilai *fitness value* adalah 1/nilai fungsi tujuan. i adalah sampel ke- i dari sampel (bernilai antaran 1 dan N), dan I adalah indek dari matriks sub sampel (*gen/job/vertex*).

3) **Penentuan** orang tua

   Penentuan orang tua dilakukan dengan mekanisme *roulette wheel selection*. Sampel dengan *fitness value* yang tinggi akan memiliki probabilitas yang lebih tinggi untuk menjadi orang tua dibandingkan yang lain. Orang tua pertama dipilih dari sampel elit dan orang tua kedua dipilih dari semua sampel di iterasi terakhir dengan pembobotan LFR.

4) ***Crossover***

Crossover dilakukan dengan teknik *two-point order crossover technique*, dengan memilih titik secara acak dari kedua orang tua. Keturunan yang dihasilkan teknik ini memiliki bagian yang sama antara dua titik dari kedua orang tua.

5) **Mutasi**

Mutasi dilakukan dengan teknik mutasi *swapping*, dimana mutasi dilakukan dengan menukar urutan dalam sampel.

Pembaharuan *Crossover Rate* dan *Mutation Rate.*

Pembaharuan parameter dilakukan dengan menghitung rasio antara nilai fungsi tujuan pada iterasi saat itu dengan nilai fungsi tujuan terbaik dari semua iterasi, yang dinotasikan sebagai u. *Crossover rate* kemudian diperbaharui dengan persamaan (2.17) dan *mutation rate* dihitung dengan setengah dari nilai *crossover rate*.

$$P_i = \alpha u + (1 - \alpha) P_{i-1} \tag{2.17}$$

2.3.4 Algoritma *Local Improvement*

Prosedur 2-opt, 1-1 *exchange* dan 1-0 *exchange* adalah mekanisme mutasi yang dilakukan pada VRP. Prosedur ini kembangkan oleh Ai & Kachitvichyanukul (2009). Prosedur ini dilakukan untuk memperbaiki rute VRP yang terbentuk. Mutasi dilakukan jika mutasi tersebut menghasilkan jarak rute yang lebih pendek, jika tidak maka mutasi tersebut dibatalkan. 2-opt dilakukan dengan menukar urutan rute dari suatu *vehicle*. 1-1 *exchange* adalah prosedur pertukaran konsumen antar dua *vehicle*. Sedangkan 1-0 *exchange* adalah prosedur memindahkan konsumen ke *vehicle* lain. Gambar 2.10 mengilustrasikan dari ketiga prosedur diatas. Algoritma dari 2-opt, 1-1 *exchange* dan 1-0 *exchange* akan dijelaskan pada Algoritma 1.5, 1.6 dan Algoritma 1.7.

Algoritma 1.5 algoritma 2-opt

1. Tetapkan n sebagai jumlah konsumen dalam rute
2. Untuk setiap $i = 1, \ldots . (n - 2)$ dan $j = (i + 2), \ldots . n$

a. Ubah rute dengan menukar arah rute dari konsumen dengan urutan i, j, $(i+1)$, dan $(j+1)$.
b. Evaluasi fisibilitas dari rute yang telah diubah dan perubahan biaya perjalanan yang terjadi.
c. Jika rute tetap fisibel dan total biaya perjalanan menjadi lebih baik maka simpan perubahan rute. Jika tidak maka kembalikan rute ke kondisi semula sebelum langkah 2.a.

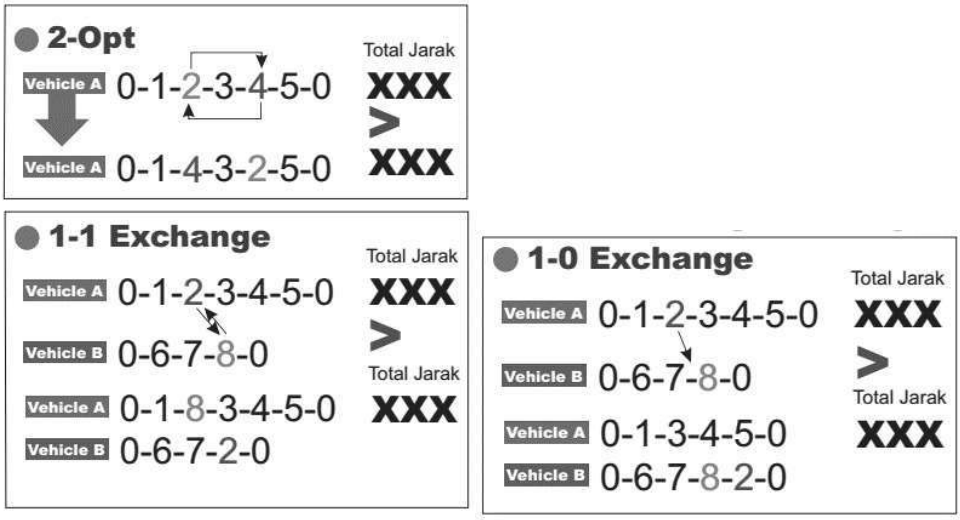
Algoritma 1.6 algoritma 1-1 *exchange*

1. Tetapkan n seabagai jumlah konsumen dalam rute pertama.
2. Tetapkan m sebagai jumlah konsumen dalam rute kedua.
3. Untuk setiap $i = 1,..n$ dan $j = 1,...m$
   a. Jika jarak antara konsumen i pada rute pertama dan konsumen j pada rute kedua ada dalam rentang δ lanjutkan ke langkah 3b, jika tidak maka kembali ke langkah 3a dengan nilai i=i+1 dan j=j+1.
   b. Ubah rute dengan menukar urutan konsumen i di rute pertama dengan konsumen j pada rute kedua.
   c. Evaluasi fisibilitas dari rute dan perubahan biaya perjalanan yang terjadi.
   d. Jika rute tetap fisibel dan total biaya perjalanan menjadi lebih baik maka simpan perubahan rute. Jika tidak maka kembalikan rute ke kondisi semula sebelum langkah 3.a.

Algoritma 1.7 algoritma 1-0 *exchange*

1. Tetapkan n seabagai jumlah konsumen dalam rute pertama.
2. Tetapkan m seabagai jumlah konsumen dalam rute kedua.
3. Untuk setiap $i = 1,..n$ dan $j = 1,...m$
   a. Jika jarak antara konsumen i pada rute pertama dan konsumen j pada rute kedua ada dalam rentang δ lanjutkan ke langkah 3b, jika tidak maka kembali ke langkah 3a dengan nilai i=i+1 dan j=j+1.
   b. Ubah rute dengan memindahkan konsumen i di rute pertama ke urutan setelah konsumen j di rute kedua.

c. Evaluasi fisibilitas dari rute dan perubahan biaya perjalanan yang terjadi.
d. Jika rute tetap fisibel dan total biaya perjalanan menjadi lebih baik maka simpan perubahan rute. Jika tidak maka kembalikan rute ke kondisi semula sebelum langkah 3.a.



Gambar 2.10 Ilustrasi prosedur 2-opt, 1-1 *exchange,* dan 1-0 *exchange.*

2.4 Posisi Penelitian

Pada penelitian ini akan mengembangkan algoritma CEGA untuk menyelesaikan kasus HFVRP dengan *two-dimensional loading constraint*. Beberapa penelitian yang sudah dilakukan untuk menyelesaikan varian dari kasus VRP *with loading constraint* dan varian nya. Tabel 2.1 menjelaskan beberapa penelitian yang memiliki kesamaan dengan penelitian ini, sehingga terlihat posisi dari penelitian yang dilakukan akan dilakukan.

2L-CVRP menjadi salah satu objek yang cukup banyak diteliti oleh beberapa peneliti. Beberapa metode telah digunakan untuk menyelesaikannya. Zachariadis *et al.,* (2009) menggunakan *guided tabu search* untuk menyelesaikan kasus 2L-CVRP. Wei *et al.*, (2015) memilih untuk menggunakan metode *Variable neighborhood search*. *Ant colony optimization* digunakan untuk menyelesaikan 2L-CVRP oleh Fuellerer *et al.,* (2009) dan 3L-CVRP oleh Fuellerer *et al.* (2010). Kasus

3L-CVRP pernah diselesaikan dengan dua metode yang berbeda oleh Ruan *et al.,* (2013) dan Junqueira & Morabito, (2015) yaitu *Classical routing and loading heuristics* dan metode metaheuristik *Honey Bee Mating Optimization*. Leung *et al.*, (2013) melakukan penelitian tentang varian dari 2L-CVRP yaitu 2L-HFVRP. Leung *et al.* menganggap dengan menambahkan *heterogeneous fleet* membuat kasus ini semakin mendekati realita di dunia logistik dan distribusi. Hokama et al., (2016) menggunakan metode *Guided Tabu Search* dan *Biased random-key genetic algorithms* untuk menyelesaikan kasus VRP *with loading constraint.* VRP *with loading constraint* berarti mengakomodir untuk 2L-CVRP dan 3L-CVRP. Fernández et al., (2013) dan Li & Zhang, (2015) meneliti tentang *Bin packing problem* (BPP). Fernández et al., (2013) menyelesaikan 2-*dimension* BPP menggunakan metode *Parallel multi-objective algorithm* sedangkan Li & Zhang, (2015) menyelesaikan 3-*dimension* BPP menggunakan *Hybrid differential evolution algorithm*.

Dari beberapa penelitian diatas hampir semua yang meneliti VRP *with loading constraint* baik 2 dimensi maupun yang 3 dimensi menggunakan dua skenario yaitu *sequential* dan *unrestricted*. Hal ini dilakukan karena di dunia nyata hampir semua kasus distribusi dalam menyusun muatannya akan dibuat sedemikian rupa agar mudah untuk dibongkar muat. Hal ini dilakukan dengan cara, muatan milik konsumen yang dikunjungi terakhir akan diletakkan paling dalam dan sebaliknya. Seperti inilah yang dilakukan dalam skenario *sequential* pada VRP *with loading constraint.*

6 dari 10 penelitian Zachariadis *et al.,* (2009), Fuellerer *et al.,* (2009 & 2010), Leung *et al.,* (2013), Wei *et al.*, (2015), dan Hokama *et al.,* (2016) pada objek penelitiannya diasumsikan bahwa muatan dari konsumen memiliki orientasi yang tetap (tidak dapat dirotasi). Asumsi tersebut mengacu pada barang-barang dengan berat yang sangat besar atau mudah pecah sehingga tidak bisa dilakukan rotasi. Namun ada juga yang mempertimbangkan rotasi pada penyusunan muatan di kontainer. Jika dimungkinkan maka rotasi ini akan membuat penataan dalam kontainer semakin lebih fleksibel dan utilitas dari luasan kontainer tersebut dapat dimaksimalkan.

Sebagian besar penelitian menggunakan metode *heuristic* dan *metaheuristic* untuk menyelesaikan kasus VRP *with loading constraint* dan BPP. Hal ini dikarenakan dua metode ini memiliki reputasi yang baik dalam menyelesaikan kasus yang dikategorikan sebagai *NP-hard problem* dengan waktu yang relatif cepat. *Initial solution* menjadi salah satu kunci dalam keberhasilan metode *heuristic* dan *metaheuristic* dalam menghasilkan solusi yang optimal. Sebagian besar penelitian yang ada menggunakan suatu teknik/mekanisme/ algoritma tersendiri untuk memunculkan *initial solution* dari masing-masing kasus yang mereka teliti.

Penelitian kali ini akan mencoba meneliti varian dari 2L-CVRP yaitu 2L-HFVRP menggunakan metode *metaheuristic hybrid* antara *cross entropy* dan *genetic algorithm.* Untuk lebih mendekati realita maka dipertimbangkan dua skenario yaitu *sequential* dan *unrestricted*. Rotasi pada penataan muatan dimungkinkan untuk memaksimalkan utilitas luasan kontainer. , dan menggunakan mekanisme matriks probabilitas transisi dan *local improvement* dari penelitian Ai & Kachitvichyanukul (2009) untuk mendapatkan *initial solution* yang baik.

Tabel 2.1 Tabel perbandingan dan posisi penelitian

	Zachariadis *et al.,* (2009)	Fuellerer et al., (2009)	Fuellerer et al. (2010)	Leung *et al.,* (2013)	Ruan et al., (2013)	Fernández et al., (2013)	Li & Zhang, (2015)	Junqueira & Morabito, (2015)	Wei et al., (2015)	Hokama et al., (2016)	Penelitian ini
Obyek penelitian											
2L-HFVRP				•							•
2L-CVRP	•	•							•		
3L-CVRP			•		•			•			
VRP with loading constraints										•	
three-dimensional bin packing problem (3D-BPP)							•				
Two-dimensional bin packing						•					
Metode											
Simulated Annealing (SA)				•				•			
SA with heuristic local search				•							
Guided Tabu Search	•										
Branch-and-cut										•	
Hybrid differential evolution algorithm							•				
Biased random-key genetic algorithms										•	

Tabel 2.1 Tabel perbandingan dan posisi penelitian **(lanjutan)**

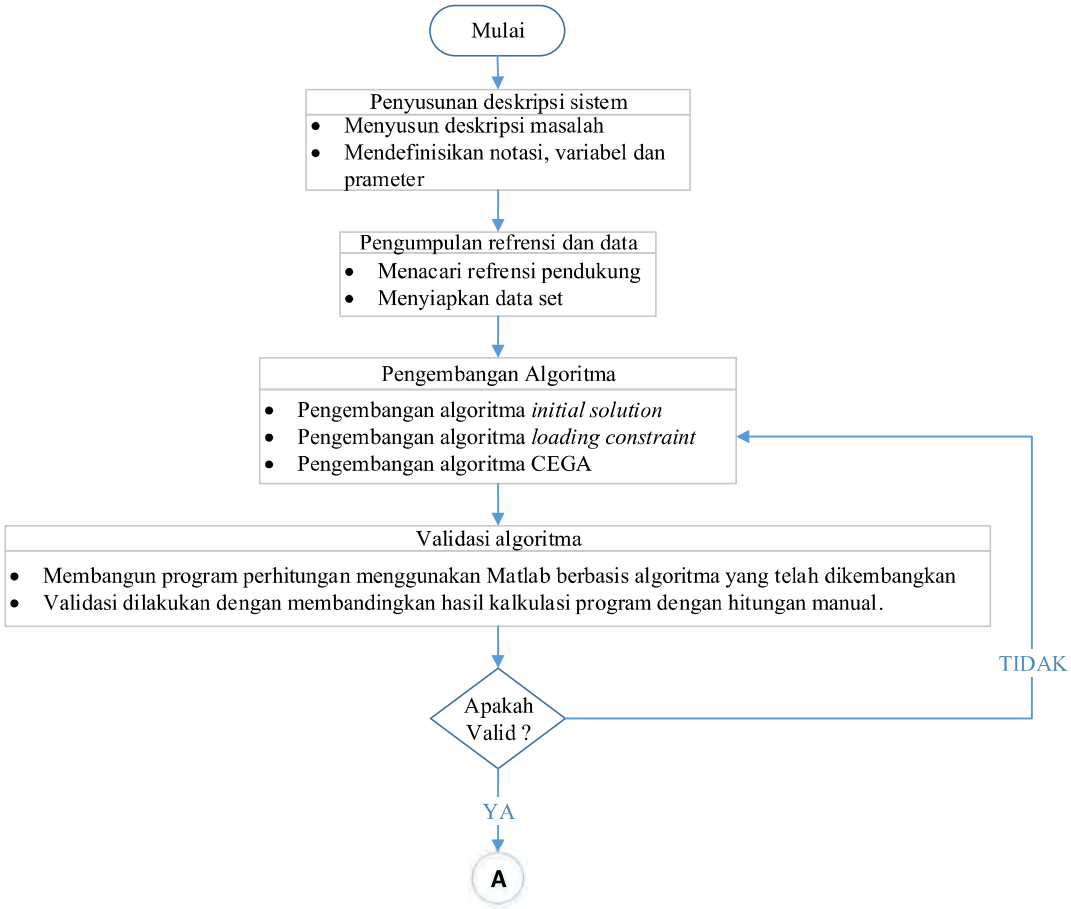
	Zachariadis *et al.,* (2009)	Fuellerer et al., (2009)	Fuellerer et al. (2010)	Leung *et al.,* (2013)	Ruan et al., (2013)	Fernández et al., (2013)	Li & Zhang, (2015)	Junqueira & Morabito, (2015)	Wei et al., (2015)	Hokama et al., (2016)	Penelitian ini
Ant colony optimization		•	•								
Classical routing and loading heuristics								•			
Honey Bee Mating Optimization				•	•						
Variable neighborhood search									•		
Parallel multi-objective algorithm						•					
Cross Entropy Genetic Algorithm											•
Karakteristik *demand*											
2-dimensi	•	•		•		•			•	•	•
3-dimensi			•		•		•	•		•	
Orientasi											
Fix	•			•					•	•	
Rotasi		•	•		•	•	•	•			•

Tabel 2.1 Tabel perbandingan dan posisi penelitian **(lanjutan)**

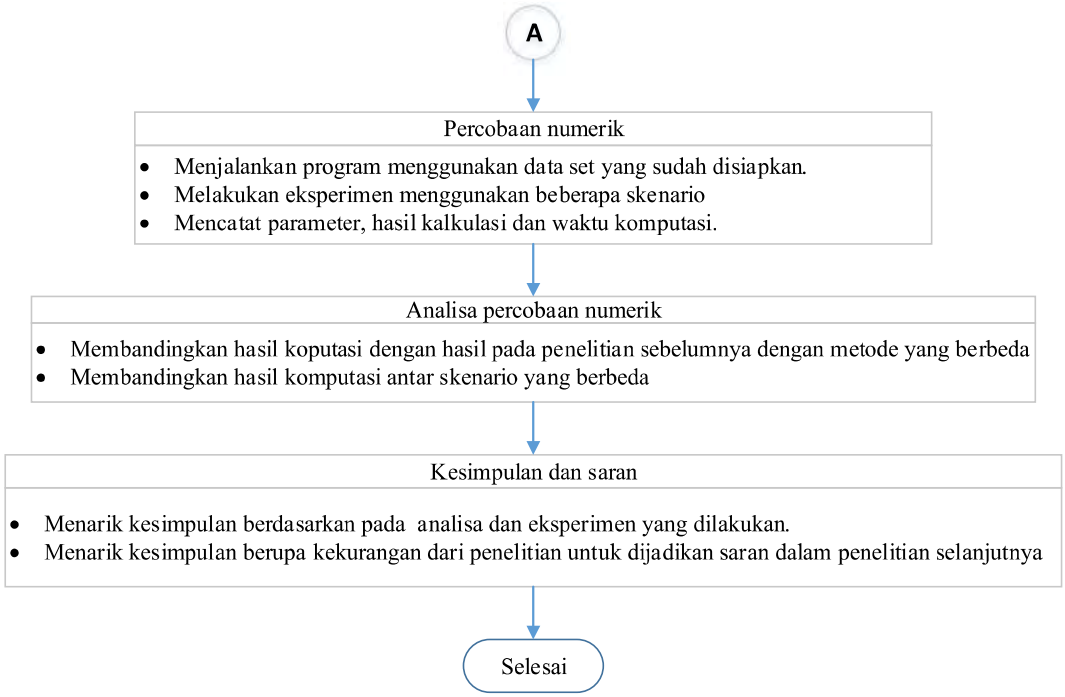
	Zachariadis *et al.,* (2009)	Fuellerer et al., (2009)	Fuellerer et al. (2010)	Leung *et al.,* (2013)	Ruan et al., (2013)	Fernández et al., (2013)	Li & Zhang, (2015)	Junqueira & Morabito, (2015)	Wei et al., (2015)	Hokama et al., (2016)	Penelitian ini
Karakteristik *loading constraint*											
squential	•	•	•	•	•			•	•	•	•
unrestricted	•	•		•		•	•		•	•	•
Initial solution											
random						•	•			•	
Savings Algorithm		•	•								
Constructive heuristics								•			
Best Fit Decreasing heuristic	•			•							
Iterative two-phase search method,					•						
Insertion-based mechanism									•		
Transition probabilistic matrix + local improvement mechanism											•

BAB 3
METODOLOGI PENELITIAN

Pada bab ini akan dijelaskan secara sistematis langkah-langkah (metodologi) yang akan dilakukan dalam penelitian ”Pengembangan Algoritma *Cross Entropy Genetic Algorithm* untuk *Two Dimensional Loading Constraint Heterogeneous Fleet Vehicle Routing Problem*”. Gambar 3.1 menjelaskan garis besar metodologi pada penelitian ini.



Gambar 3.1 Metodologi Penelitian



Gambar 3.1 Metodologi Penelitian (lanjutan)

3.1 Penyusunan Deskripsi Masalah

Langkah pertama setelah menentukan topik penelitian (*two-dimensional loading constraint heterogeneous fleet vehicle routing problem*) adalah menyusun deskripsi masalah. Deskripsi masalah ini adalah menuliskan secara rinci masalah yang menjadi objek penelitian. *Two-dimensional loading constraint heterogeneous fleet vehicle routing problem* digambarkan dengan notasi, variabel dan parameter dengan struktur matematis. Menetapkan fungsi tujuan, variabel keputusan, dan kendala-kendala yang digunakan.

3.2 Pengumpulan Referensi dan Data

Setelah deskripsi masalah dijabarkan langkah selanjutnya adalah melakukan pengumpulan referensi dan data. Berbeda dengan studi literatur yang dilakukan dalam penentuan topik penelitian, pada tahap ini referensi dan data yang dikumpulkan sudah lebih berfokus pada topic tertentu. Pengumpulan referensi dan data dilakukan untuk mendukung dalam hal teori untuk menyelesaikan penelitian

ini. Referensi yang dikumpulkan dapat berupa penelitian terdahulu dan literatur dari suatu topik tertentu. Dalam penelitian ini referensi yang dikumpulkan adalah penelitian terdahulu dan literatur tentang VRP *with loading constraint* dan metode-metode yang telah digunakan untuk menyelesaikan permasalahan tersebut. Tujuan dari kegiatan ini adalah untuk meyakinkan bahwa penelitian ini dapat dan layak untuk dilakukan.

Data yang dikumpulkan dalam penelitian ini berupa hasil komputasi dari percobaan pada penelitian terdahulu dan data set yang digunakan dalam menguji algoritma yang diusulkan. Data hasil komputasi dapat digunakan untuk pembanding dan data set digunakan untuk menguji solusi yang ditawarkan dalam penelitian ini. Referensi utama yang digunakan dalam penelitian ini adalah penelitian Leung, *et al.* (2013) begitu pula data komputasi nya yang digunakan sebagai pembanding. Data set yang digunakan adalah data set Iori, (2005) dengan modifikasi berupa karakteristik *vehicle* oleh Leung, *et al.* (2013). Penjelasan detail dari data set yang digunakan akan dijelaskan pada bab berikutnya.

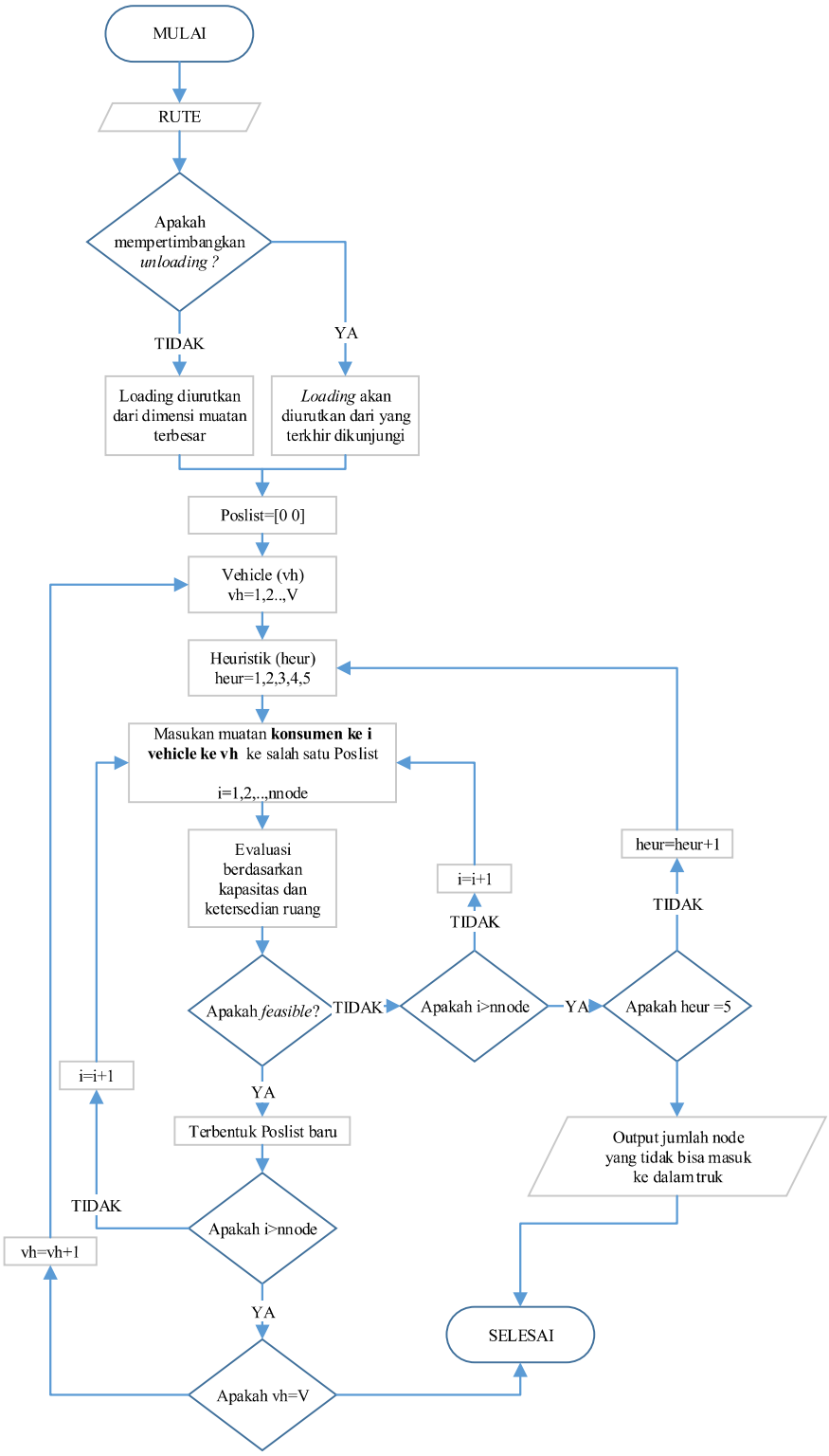
3.3 Pengembangan Algoritma

CEGA telah terbukti efektif dalam menyelesaikan beberapa kasus kombinatorial namun secara praktis algoritma ini belum pernah digunakan dalam menyelesaikan kasus *two-dimensional loading constraint heterogeneous fleet vehicle routing problem* (2L-HFVRP). Penelitian ini mencoba untuk mengembangkan algoritma CEGA untuk menyelesaikan kasus 2L-HFVRP dengan menggabungkan beberapa algoritma yang pendukung untuk menghasilkan hasil yang baik. Agar semua *demand* konsumen dapat *feasible* dalam luasan kontainer maka digunakan algoritma *loading constraint* yang di dalamnya terdapat 5 heuristik dalam menentukan posisi peletakan muatan. Matriks probabilitas transisi dan mekanisme *local improvement* digunakan untuk mendapatkan *initial solution* yang baik. *Initial solution* yang baik menjadi kunci dalam menghasilkan solusi yang baik dalam metode metaheuristik. Selanjutnya *initial solution* tersebut akan diproses lanjut dengan algoritma CEGA untuk mencari solusi yang optimum.

2.1.1 Loading constraint algorithm

Fungsi tujuan dari 2L-HFVRP adalah meminimalkan biaya perjalanan yang terdiri dari *fix cost* dan *variable cost*. Dengan dimensi dari *demand* konsumen menjadi salah satu konstrain. Sehingga rute yang terbentuk juga *feasible* untuk mengangkut semua *demand* ke dalam suatu luasan kontainer. Karena berdiri sebagai konstrain maka algoritma ini dijalankan setiap pembentukan solusi dan setiap mutasi yang terjadi. Gambar 3.2 menjelaskan bagaimana alur dalam menjalankan algoritma *loading constraint*. Dalam diagram alur input berupa rute, hal ini menunjukkan bahwa setiap terbentuk rute baru algoritma ini akan dijalankan untuk memeriksa apakah rute tersebut *feasible* terhadap *demand* konsumen yang dibawa *vehicle* dalam rute nya. Pada awal algoritma jika menggunakan skenario *sequential* maka rute akan diurutkan secara terbalik sehingga barang dari konsumen yang akan dikunjungi paling akhir akan dimasukan terlebih dahulu sehingga aliran bongkar muat-nya menjadi *last in first out* (LIFO) , sedangkan jika menggunakan skenario *unrestricted* maka rute akan diurutkan berdasarkan dimensi terbesar dari barang konsumen yang akan diangkut. Sehingga utilitas dari luasan kontainer dapat dimaksimalkan dengan baik.

Dalam algoritma *loading constraint* terdapat 5 heuristik untuk menentukan posisi mana yang akan dipilih untuk peletakan muatan ke dalam kontainer. Pada diagram alur juga digambarkan bahwa *heur1* akan dicoba, jika tidak berhasil menghasilkan solusi yang *feasible* maka dijalankan heuristik selanjutnya hingga *heur5* secara berurutan. Heuristik ini diurutkan berdasarkan kompleksitas, sehingga jika mampu dikerjakan dengan heuristic dengan kompleksitas rendah maka tidak perlu dikerjakan dengan heuristik dengan kompleksitas tinggi. Hal ini bertujuan untuk mengurangi waktu komputasi. Pada akhir diagram alur yang dilakukan adalah mencatat apakah ada konsumen yang belum terlayani. *Output* inilah yang menjadi indikator bagi rute, apakah rute tersebut *feasible* atau tidak. Karena rute dikatakan *feasible* ketika semua barang konsumen yang akan dikunjungi harus dapat masuk semua ke dalam kontainer.



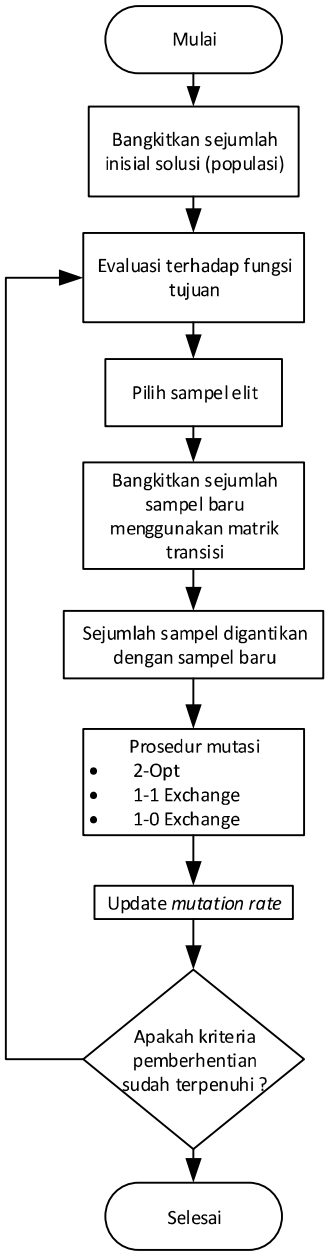
Gambar 3.2 Diagram alir algoritma *loading constraint*

2.1.2 Algoritma CEGA

Algoritma CEGA menjadi algoritma yang digunakan untuk mencari solusi optimal. Dalam penelitian ini algoritma CEGA tidak berjalan sendiri. Beberapa algoritma yang sudah dijelaskan sebelumnya akan mendukung algoritma CEGA dalam mencari solusi optimum. Algoritma 1.4 yang merupakan algoritma dasar dari CEGA dapat diterapkan di berbagai kasus kombinatorial. Urutan langkah-langkah tetap sama, yang berbeda adalah penjabaran secara spesifik terhadap beberapa langkahnya. Seperti pada Algoritma 1.4 langkah 2 pembangkitan solusi awal, banyak mekanisme yang dapat digunakan dalam membangkitkan solusi awal. Penelitian ini menggunakan Matriks probabilitas transisi untuk membangkitkan solusi awal sejumlah N sampel. Pada setiap iterasi sampel akan dikenakan mekanisme mutasi. Sebagai ganti mekanisme *crossover* pada setiap iterasi sejumlah random akan digantikan dengan sampel baru yang dihasilkan dari mekanisme matriks probabilitas transisi. Pada langkah 6, mutasi dilakukan dengan prosedur 2-opt, 1-1 *exchange* dan 1-0 *exchange* dengan beberapa penyesuaian. Perubahan rute yang terjadi akibat mutasi membuat mekanisme *loading constraint* bekerja untuk memastikan bahwa rute baru yang hasilkan dari mutasi tetap *feasible* dengan penataan barang di kontainer. Jika prosedur mutasi ternyata membuat rute menjadi tidak *feasible* maka mutasi pada gen yang terpilih akan dibatalkan dan mencoba untuk mutasi dengan gen lainnya. Gambar 3.3 menjelaskan algoritma CEGA yang digunakan dalam penelitian ini. Penjelasan lebih lanjut akan dijelaskan pada bab berikutnya.

3.4 Validasi Algoritma

Untuk memastikan algoritma valid dan dapat digunakan untuk menyelesaikan 2L-HFVRP maka dibangun suatu program yang didasarkan dengan beberapa algoritma yang sudah dijelaskan sebelumnya. Program dibuat menggunakan aplikasi MATLAB. Sampel dari hasil kalkulasi program akan dihitung secara manual untuk memastikan bahwa program sudah benar dalam menjalankan algoritma. Program dikatakan sudah benar jika solusi yang dihasilkan tidak melanggar semua konstrain.

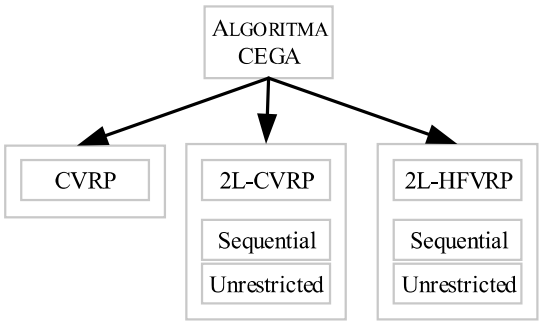


Gambar 3.4 Diagram alir algoritma CEGA

3.5 Percobaan Numerik

Percobaan numerik dilakukan dengan menjalakan program dengan input data set yang sudah disiapkan. Fungsi tujuan dan waktu komputasi akan dicatat

dengan untuk kemudian dilakukan analisa. Dalam percobaan numerik ini dilakukan beberapa skenario. Gambar 3.5 menjelaskan skenario yang akan dijalankan. Algoritma CEGA akan dicoba untuk menyelesaikan CVRP standar, 2L-CVRP dan 2L-HFVRP. Masing-masing dari 2L-CVRP dan 2L-HFVRP diselesaikan dengan skenario *sequential* dan *unrestricted.* Skenario ini dilakukan untuk kemampuan dari algoritma CEGA dalam menyelesaikan kasus *Vehicle Routing Problem*.



Gambar 3.5 Skema skenario percobaan

3.6 Analisa Percobaan Numerik

Analisis dilakukan dengan melihat hasil komputasi berupa total biaya perjalanan dan waktu komputasi. Analisis dilakukan berdasarkan perbandingan dengan hasil komputasi pada penelitian sebelumnya. Analisis juga dilakukan berdasarkan perbandingan antar skenario (*sequential* dan *unrestricted*). Analisis dilakukan untuk melihat kelemahan dan keunggulan dari metode yang digunakan.

3.7 Kesimpulan dan Saran

Tahap terakhir adalah pengambilan kesimpulan berdasarkan analisis yang sudah dilakukan sebelumnya. Kesimpulan ini akan menjawab pertanyaan-pertanyaan pada rumusan masalah. Saran dirumuskan berdasarkan kesimpulan yang diambil untuk penelitian yang akan datang.

BAB 4
DESKRIPSI MODEL DAN PENGEMBANGAN ALGORITMA

Bab ini akan menjelaskan deskripsi model yang akan diselesaikan dan pengembangan algoritma yang digunakan untuk menyelesaikan model.

4.1. *Two loading constraint Heterogeneous Fleet Vehicle Routing Problem*

2L-HFVRP digambarkan dengan grafik jaringan $G = (V, E)$, dimana $V = \{0,1,\dots,n\}$ adalah *vertex* yang merepresentasikan depot (*vertex* 0) dan sejumlah konsumen (*vertex* 1,2,…n) dan $E = \{e_{ij}; i,j \in V\}$ adalah sejumlah *arc.* Untuk setiap $e_{ij} \in E$ berasosiasi dengan jarak d_{ij} dan $d_{ii} = 0$. Terdapat P jenis kendaraan yang berbeda di depot, dengan jumlah kendaraan yang tidak terbatas untuk setiap jenis. Setiap kendaraan t $(t = 1,2,\dots,P)$ memiliki atribut seperti kapasitas Q_t, biaya tetap F_t,biaya variabel V_t, panjang L_t, dan lebar W_t. *Loading surface* dari kendaraan jenis t adalah $A_t = L_t \times W_t$. Setiap kendaraan dengan jenis yang berbeda memiliki atribut yang berbeda-beda dengan asumsi: $Q_1 \leq Q_1 \leq \cdots \leq Q_p$, sehingga $F_1 \leq F_1 \leq \cdots \leq F_p$, dan $V_1 \leq V_1 \leq \cdots \leq V_p$. Biaya perjalanan untuk setiap arc $e_{ij} \in E$ oleh kendaraan t adalah $c_{ijt} = V_t \times d_{ij}$. *cost.* Formula biaya perjalanan dari suatu rute untuk jenis kendaraan t adalah $C_R = F_t + \sum_{i=1}^{i<|R|} V_t \times d_{R(i),R(i+1)}$. Dimana R adalah rute yang dimulai dan berakhir di depot. 2L-HFVRP memiliki fungsi tujuan meminimalkan total biaya perjalanan. Fungsi tujuan didapatkan dengan meminimalkan total biaya perjalanan dapat dihitung dengan menjumlahkan C_R untuk semua rute R.

Setiap konsumen i $(i = 1,2,\dots,n)$ memiliki sejumlah *demand* m_i berupa barang berbentuk persegi panjang yang dinotasikan dengan IT_i dan berat total dari IT_i dinotasikan dengan D_i. Setiap barang $I_{ir} \in IT_i$ $(r = 1,2,\dots,m_i)$ memiliki ukuran yang spesifik yaitu panjang l_{ir} dan lebar w_{ir}. Sehingga $a_i = \sum_{r=1}^{m_i} w_{ir} \times l_{ir}$ adalah total luas dari barang-barang yang menjadi *demand* konsumen i.

Pada 2L-HFVRP, solusi dianggap fisibel jika memenuhi beberapa konstrain sebagai berikut:

- Semua barang dari suatu konsumen harus diangkut oleh kendaraan yang sama. Pemisahan pengiriman tidak diizinkan.
- Semua kendaraan harus berawal dan berakhir di depot
- Setiap konsumen hanya dapat dilayani satu kali.
- Kapasitas, panjang dan lebar dari semua *demand* dalam suatu rute tidak boleh melebihi kapasitas, panjang dan lebar kendaraan.
- Setiap barang tidak dapat ditumpuk pada *loading surface* kendaraan.

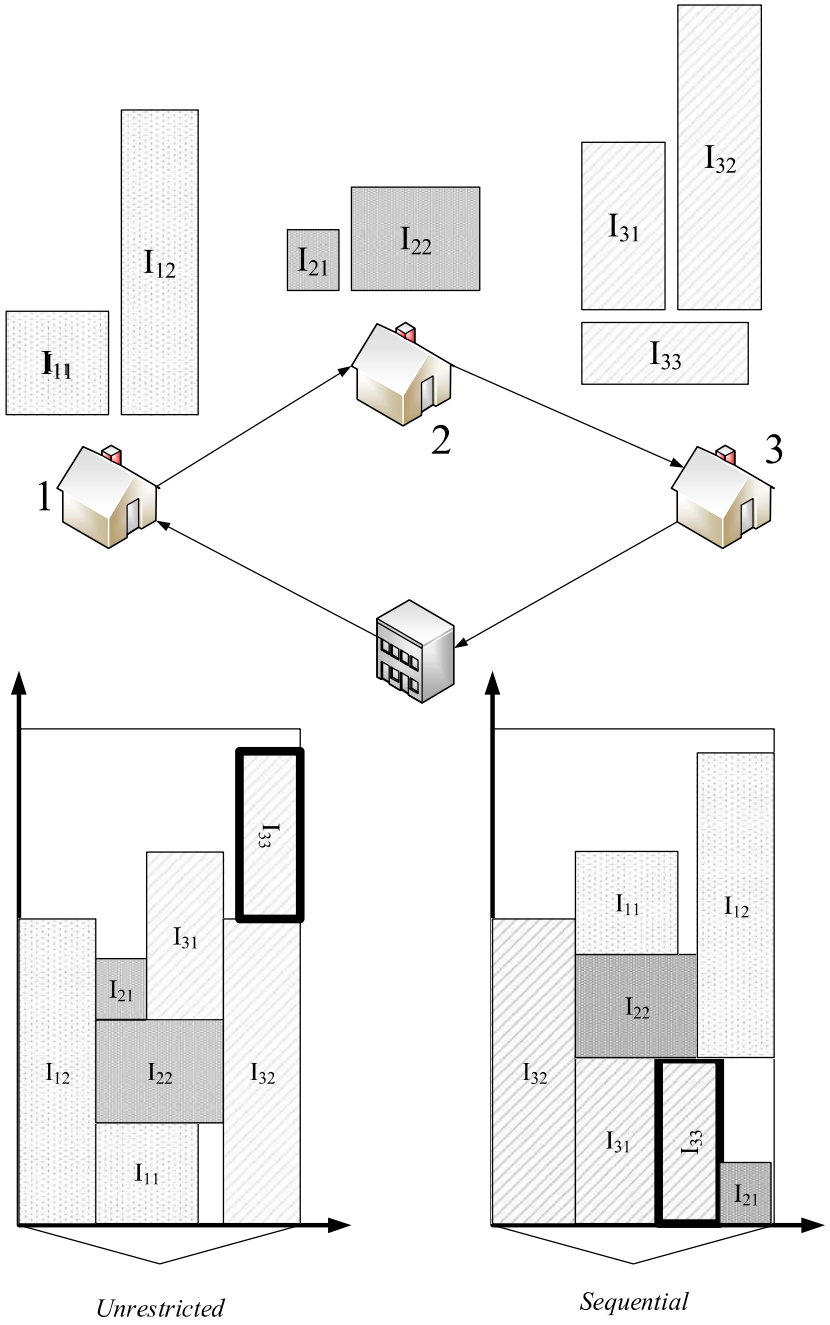
Pada penelitian ini diasumsikan bahwa barang dapat dirotasi 90^o dalam proses penataan pada kendaraan. Penelitian ini menggunakan dua versi dari 2L-HFVRP yaitu *sequential* dan *unrestricted*.

Gambar 4.1 menjelaskan 2 versi dari 2L-HFVRP. Pada *sequential* barang milik konsumen 3 (I_{31}, I_{32}, I_{33}) diikuti konsumen berikutnya hingga konsumen pertama dan sebaliknya untuk versi *unrestricted*. Pada gambar 4.1 barang yang bergaris tebal (I_{33}) dirotasi 90^o untuk mendapatkan utilitas yang lebih baik.

Dalam menguji algoritma yang diusulkan di penelitian ini, maka digunakan data set yang berisi contoh kasus dari 2L-HFVRP. Data set yang digunakan adalah data set yang digunakan oleh Iori (2005) dan data berupa atribut kendaraan yang diambil pada penelitian Leung *et al.* (2013). Data set terdiri dari 36 jenis contoh kasus dengan masing-masing jenis terdiri dari 5 kelas yang berbeda.

Tabel 4.1 menjelaskan beberapa contoh karakteristik data set yang digunakan. Dari kelima kelas dalam data set, kelas 1 berisi kasus murni CVRP, tidak ada data dimensi dari setiap *demand* konsumen. Sehingga hanya kelas 2-5 yang merepresentasikan kasus dari 2L-CVRP. Pada Tabel 4.1 kolom *it* menginformasikan jumlah barang yang harus dikirim, kolom *vh* menginformasikan jumlah kendaraan yang tersedia, dan kolom *r%* adalah persentase total luasan dari semua *demand* dibanding total luasan dari *loading surface* dari semua kendaraan. Agar data set sesuai dengan kasus 2L-HFVRP maka diperlukan data tambahan berupa data atribut kendaraan yang menunjukkan jenis kendaraan yang berbeda. Pada Tabel 4.2 menjelaskan bahwa setiap contoh kasus memiliki sejumlah jenis

kendaraan yang dapat digunakan. Tabel 4.2 menginformasikan atribut dari masing-masing jenis kendaraan berupa kapasitas, panjang, lebar, biaya variabel dan biaya tetap.



Gambar 4.1 Ilustrasi dua versi 2L-HFVRP

Tabel 4.1 Karakteristik data set 2L-CVRP

		Class 2			Class 3			Class 4			Class 5		
Inst	cus	it	vh	r%	it	vh	r%	it	vh	r%	it	vh	r%
1	15	24	3	78	31	3	82	37	4	70	45	4	61
2	15	25	5	52	31	5	59	40	5	53	48	5	39
3	20	29	5	68	46	5	77	44	5	62	49	5	54
4	20	32	6	58	43	6	58	50	6	63	62	6	47
5	21	31	4	72	37	4	75	41	4	76	57	5	53
6	21	33	6	54	40	6	63	57	6	72	56	6	46
7	22	32	5	71	41	5	66	51	5	67	55	6	49
8	22	29	5	63	42	5	71	48	5	68	52	6	38
9	25	40	8	57	61	8	61	63	8	60	91	8	53
10	29	43	6	74	49	6	66	72	7	73	86	7	63
11	29	43	6	77	62	7	74	74	7	83	91	7	63
12	30	50	9	62	56	9	52	82	9	66	101	9	58
13	32	44	7	69	56	7	68	78	7	77	102	8	59
14	32	47	7	65	57	7	65	65	7	61	87	8	49
15	32	48	6	76	59	6	84	84	8	72	114	8	72

Sumber: Zachariadis dkk. (2009)

Tabel 4.2 Data set jenis –jenis kendaraan

Inst	A					B					C					D				
	Q_A	L_A	W_A	F_A	V_A	Q_B	L_B	W_B	F_B	V_B	Q_C	L_C	W_C	F_C	V_C	Q_D	L_D	W_D	F_D	V_D
1	20	10	10	10	1	25	15	15	20	1.1	40	25	25	30	1.2	60	40	20	40	1.3
2	20	10	10	10	1	25	15	15	20	1.1	40	25	25	30	1.3	55	40	20	40	1.5
3	20	10	10	10	1	30	15	15	20	1.1	60	40	20	40	1.2					
4	20	10	10	10	1	40	20	20	20	1.1	60	40	20	30	1.2					
5	1000	10	10	10	1	2500	15	15	20	1.1	4000	25	25	30	1.3	6000	40	20	50	1.5
6	2000	10	10	10	1	2500	15	15	20	1.1	4000	40	20	30	1.3					
7	200	10	10	10	1	500	15	15	20	1.1	2000	25	25	120	6	4500	40	20	250	8
8	200	10	10	10	1	500	15	15	20	1.1	2000	25	25	120	5	4500	40	20	250	10
9	20	10	10	10	1	25	15	15	20	1.1	48	40	20	30	1.3					
10	200	10	10	10	1	500	15	15	20	1.1	2000	25	25	120	8	4500	40	20	250	10
11	200	10	10	10	1	500	15	15	20	1.1	2000	25	25	120	8	4500	40	20	250	10
12	20	10	10	10	1	25	15	15	20	1.1	40	40	20	30	1.3					
13	2000	10	10	10	1	5000	15	15	50	2	30000	40	20	200	10					
14	500	10	10	10	1	1500	15	15	50	2.1	3000	20	20	400	3.2	5000	40	20	800	5
15	500	10	10	10	1	1500	15	15	50	2.1	3000	20	20	400	3.2	5000	40	20	800	5

Sumber: Leung dkk. (2013)

Setiap contoh soal dalam data set berisi informasi berupa koordinat dari setiap *vertex* beserta *demand* yang melekat padanya. Setiap *demand* memiliki atribut berupa berat dan dimensi (panjang dan lebar). Gambar 4.2 menjelaskan informasi yang ada pada setiap contoh soal dalam data set. Untuk kasus 2L-HFVRP data *vehicle* pada Gambar 4.2 akan digantikan dengan data kendaraan seperti pada Tabel 4.2.

Instance: E016-05m.dat
Class: 2
15 --- number of customers (no depot)
5 --- number of vehicles
25 --- number of items

Vehicle		
Capacity	Height	Width
55	40	20

Node	X	Y	demand
0	30	40	0
1	37	52	7
2	49	49	30
3	52	64	16
4	20	26	9

Node	number of Item	Height	Width	Height	Width
0	0	0	0	0	0
1	1	7	16	0	0
2	2	12	7	24	3
3	2	17	10	26	2
4	2	6	11	13	8

Gambar 4.2 Data set 2L-CVRP

4.2. Algoritma *Packing Heuristic*

Algoritma *packing heuristic* digunakan untuk menyusun barang dari konsumen dalam satu rute 2L-HFVRP dengan tujuan memaksimalkan utilitas dari luas permukaan kontainer. Penelitian ini menggunakan 5 heuristik dalam proses penataan barang ke dalam kontainer. 5 heuristik ini berasal dari berbagai penelitian terdahulu yang dikumpulkan dan digunakan pada penelitian Zachariadis dkk.

(2009). 5 heuristik tidak dilakukan secara bersama-sama, namun dilakukan secara berurutan hingga mendapatkan solusi yang fisibel. Kelima heuristik pada penelitian Zachariadis dkk. (2009) sudah diurutkan berdasarkan tingkat kompleksitas. Heuristik mulai dijalankan mulai dari yang memiliki tingkat kompleksitas rendah (*heur1*) hingga yang paling kompleks (*heur5*) jika belum menemukan solusi yang fisibel. Hal ini dilakukan untuk mempercepat waktu komputasi.

Penjelasan dari kelima heuristik sudah dijelaskan pada bab sebelumnya. Tabel 4.3 menjelaskan dengan *pseucode* algoritma penggunaan *packing heuristic* pada penelitian ini.

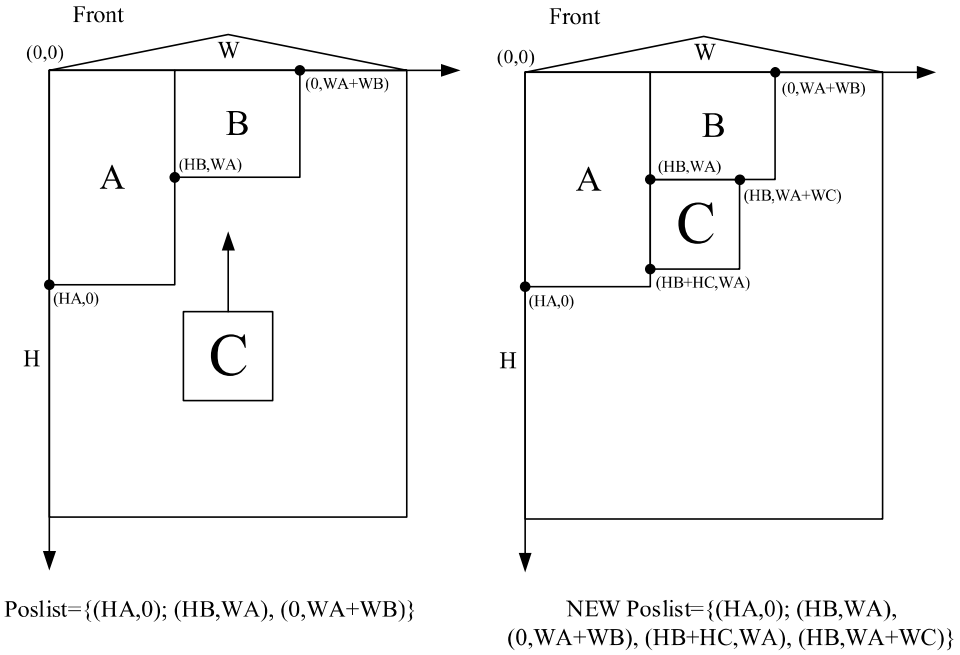
Tabel 4.3 Pseucode penggunaan *packing heuristic*

```
Version={sequential, unrestricted}; Heuristic={heur1, heur2, heur3, heur4, heur5}}
Ordering ={Ord1, Ord2}; route={1,2,…,n}

if Version = sequential
        Ord1 = route {n, n-1, n-2,…1}
elseif Version = unrestricted
        Ord2=route {1, 2,…n}
end
for heur=1:heur5
        poslist={0,0}
        for each item
                heur (placing item in a poslist)
                if not feasible
                        rotate the item
                end
                update new poslist
        end
        if route feasible
                break
        end
end
```

Tabel 4.3 juga memperlihatkan bahwa jika menggunakan versi *sequential* dan *unrestricted* maka urutan dari rute akan diubah. *Ord1* adalah urutan memasukan barang untuk versi *sequential* dan *Ord2* adalah urutan memasukan barang untuk versi *unrestricted.* Barang milik konsumen yang dikunjungi terakhir di rute akan dimasukan pertama kali dalam versi *sequential*. Namun pada versi

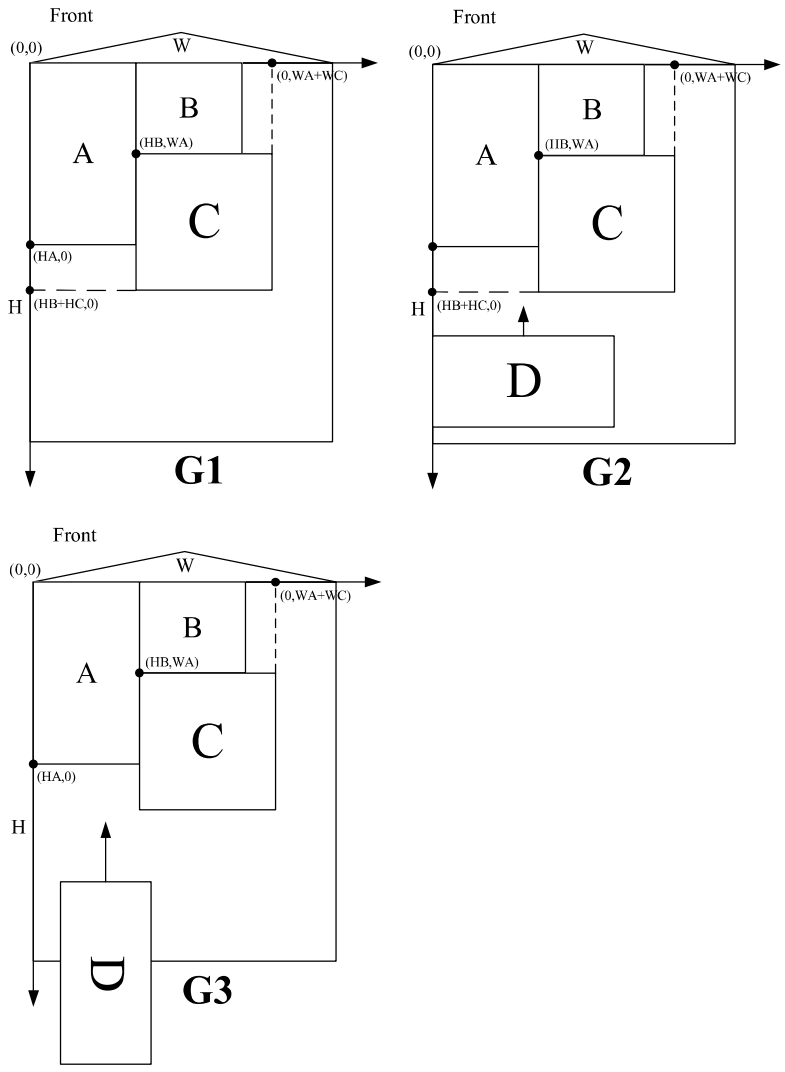
unrestricted barang konsumen pertama akan dimasukan terlebih dahulu. Rotasi dilakukan jika peletakan tidak mungkin dilakukan di semua poslist menggunakan orientasi awal. Jika suatu heuristik tidak menghasilkan solusi yang fisibel maka pada heuristik selanjutnya akan dimulai dari awal (*poslist*={0,0}).



Gambar 4.3 Ilustrasi pembentukan posisi baru pada *poslist*

Poslist akan diperbaharui ketika ada barang yang dimasukan ke dalam kontainer. Setiap barang yang dimasukan ke dalam kontainer akan menghasilkan 2 posisi baru pada *poslist*. Gambar 4.3 menunjukkan bagaimana 2 posisi baru terbentuk ketika ada barang masuk ke dalam kontainer. Posisi baru yang terbentuk pada Gambar 4.3 adalah (HB,WA+WC) dan (HB+HC,WA). Dimungkinkan kemunculan posisi baru dalam *poslist* akan menggantikan posisi yang sudah ada pada *poslist* seperti pada Gambar 4.4 (G1). Pada gambar tersebut posisi (HB,WA+WC) tidak terbentuk karena WA+WC lebih besar dibanding nilai terbesar W dalam *poslist* dan HB bukan merupakan nilai H terkecil dalam *poslist*. Sehingga posisi baru yang terbentuk adalah (0,WA+WC). Gambar 4.4 (G1)

menunjukkan bahwa posisi (HB+HC,WA) pada Gambar 4.3 tidak muncul. Hal ini dikarenakan HB+HC menjadi nilai H tertinggi dalam *poslist* dan WA bukan nilai W terkecil dalam *poslist*. Sehingga jika barang C memiliki dimensi seperti pada Gambar 4.4 maka posisi baru yang muncul adalah (0,WA+WC) dan (HB+HC,0). Gambar 4.4 (G2 dan G3) menjelaskan pemilihan posisi yang akan digunakan jika barang D dimasukan. Jika menggunakan *heur1* memilih posisi yang memiliki nilai W yang paling kecil, maka pada Gambar 4.4 (G2 dan G3) terdapat dua posisi yang memiliki nilai W terkecil yaitu (HB+HC,0) dan (HA,0). Posisi (HB+HC,0) akan terpilih ketika barang D tidak dilakukan rotasi seperti pada Gambar 4.4 (G2). Posisi (HA,0) akan dipilih ketika barang D dirotasi 90^{o} seperti pada Gambar 4.4 (G3). Jika terdapat 2 posisi yang memiliki nilai W yang sama dan keduanya sama-sama dimungkinkan untuk dipilih, maka yang dipilih adalah posisi yang memiliki nilai H paling kecil.



Gambar 4.4 Ilustrasi penyusunan barang ke kontainer

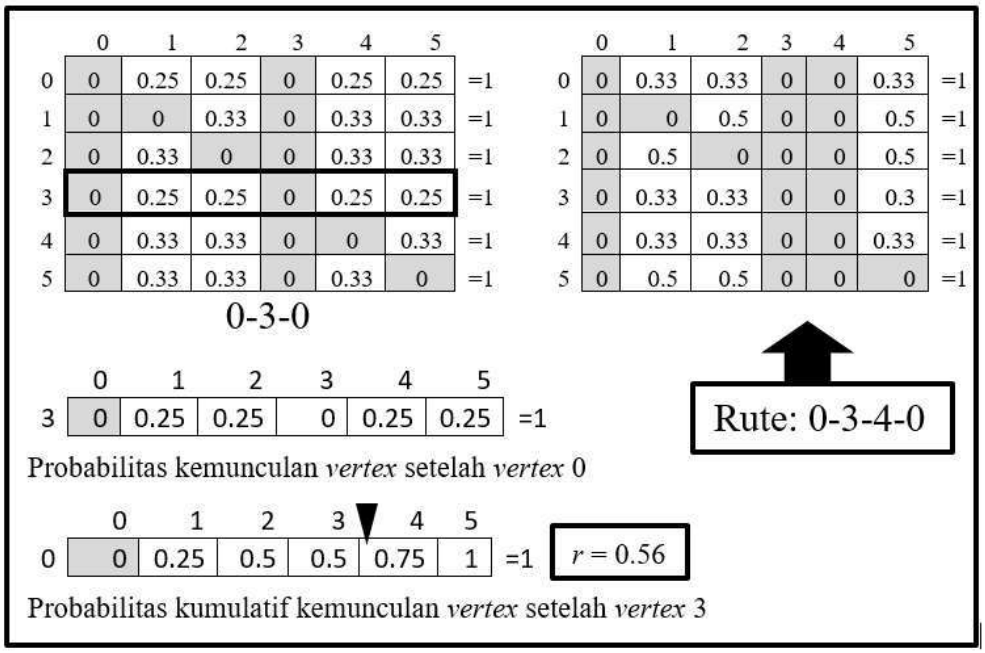
4.3. Pengembangan Algoritma CEGA

Pengembangan algoritma CEGA dilakukan agar algoritma dapat digunakan dengan baik untuk menyelesaikan suatu permasalahan yang spesifik dan menghasilkan hasil yang diharapkan. Pada sub bab ini akan menjelaskan pengembangan algoritma CEGA untuk menyelesaikan kasus 2L-HFVRP.

4.3.1. Inisial Solution

Inisialiasi solusi awal adalah hal pertama yang dilakukan di dalam algoritma CEGA. Solusi awal dapat dibangkitkan dengan menggunakan beberapa mekanisme. Mekanisme ini biasanya melekat dalam setiap Algoritma metaheuristik yang digunakan. Pada penelitian ini solusi awal dibangkitkan menggunakan matriks probabilitas transisi dan mekanisme *local improvement* yang diambil dari penelitian Ai & Kachitvichyanukul (2009).

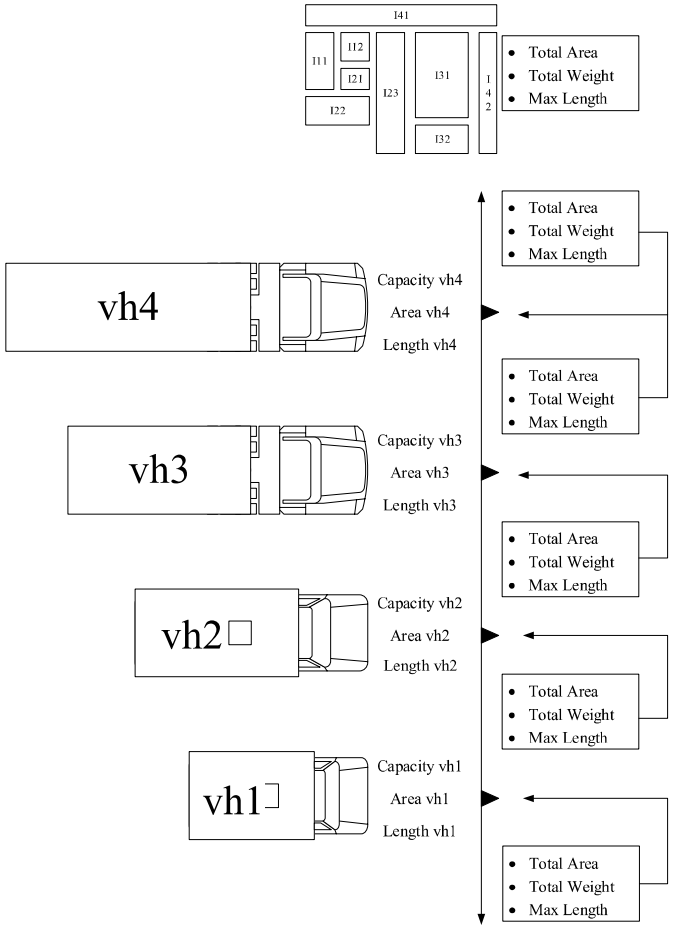
Matriks probabilitas transisi yang matriks dengan ukuran $n \times n$ yang berisi probabilitas munculnya *vertex* kolom setelah *vertex* baris. Gambar 4.5 menjelaskan bagaimana membangkitkan rute menggunakan matriks probabilitas transisi. Untuk memunculkan *vertex* tujuan, pilih baris *vertex* awal untuk dihitung probabilitas kumulatif nya. Pada Gambar 4.5 menunjukkan bagaimana membangkitkan memunculkan *vertex* baru yang akan diletakkan di urutan setelah *vertex* 3, sehingga baris yang dipilih adalah baris 3 untuk dihitung probabilitas kumulatif-nya. Bangkitkan bilangan random, misal $r = 0.56$. Bilangan random r tersebut berada di antara 0.5 (*vertex* 3) dan 0.75 (*vertex* 4) sehingga *vertex* yang terpilih adalah *vertex* 4. Cara seperti ini juga biasa disebut *roulette* wheels. Langkah selanjutnya adalah menggantikan nilai probabilitas pada kolom *vertex* yang terpilih (*vertex* 4) dengan 0 agar *vertex* tersebut tidak muncul lagi pada pembangkitan berikutnya. Jika sudah maka lakukan normalisasi matriks dengan mengubar nilai probabilitas agar setiap baris ketika dijumlahkan probabilitasnya akan berjumlah 1. Mekanisme seperti pada Gambar 4.5 dilakukan hingga semua *vertex* sudah masuk ke dalam rute. Setiap memasukan vertex baru terpilih untuk dimasukan ke dalam rute maka algoritma loading constraint dan pemeriksaan pemenuhan kapasitas kendaraan akan bekerja untuk memeriksa fisibilitas dari rute yang terbentuk. Jika rute yang terbentuk tidak fisibel maka akan dicoba dimasukan ke rute kendaraan lain atau kendaraan baru.



Gambar 4.5 Ilustrasi pembangkitan rute dengan matriks probabilitas transisi

Penambahan kendaraan baru dilakukan jika *vertex* baru yang muncul tidak fisibel untuk diletakkan di kendaraan yang sudah tersedia. Penentuan jenis kendaraan dilakukan dengan mempertimbangkan 3 hal yaitu: total beban, total luas dan panjang maksimum barang dari konsumen yang belum masuk ke dalam rute. Tabel 4.4 menjelaskan mekanisme pemilihan jenis kendaraan baru menggunakan *pseucode*. Gambar 4.6 menjelaskan mekanisme tersebut menggunakan ilustrasi gambar. Langkah pertama adalah hitung total berat, total luas dan maksimum panjang dari barang konsumen yang belum masuk ke dalam rute. Langkah selanjutnya adalah bandingkan dengan kapasitas berat, kapasitas luas, dan panjang dari masing-masing jenis kendaraan. Perbandingan dilakukan dengan kendaraan dengan kapasitas paling besar. Pada Gambar 4.6 digambarkan bahwa jika total berat, total luas dan maksimum panjang dari barang konsumen yang belum masuk ke dalam rute lebih dari kapasitas kendaraan 4 atau lebih dari kapasitas kendaraan 3 dan kurang dari kapasitas kendaraan 4 maka kendaraan 4 dipilih untuk digunakan. Jika total berat, total luas dan maksimum panjang dari barang konsumen yang

belum masuk ke dalam rute lebih besar dari kapasitas kendaraan 3 dan lebih kecil dari kapasitas kendaraan maka dipilih kendaraan 4, dan seperti itu seterusnya seperti ilustrasi pada Gambar 4.6.



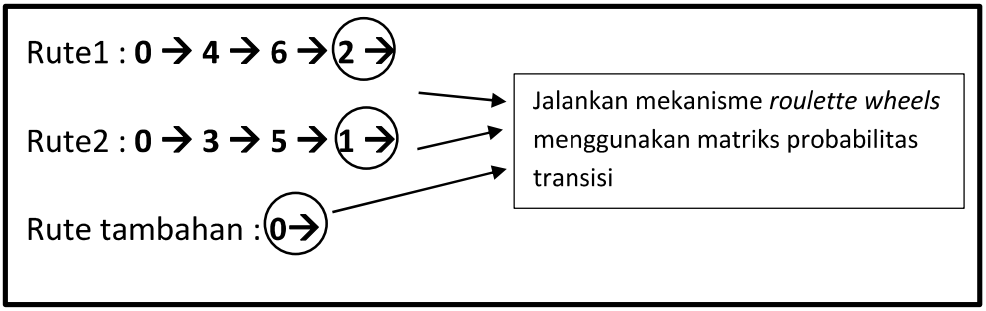
Gambar 4.6 Ilustrasi pemilihan kendaraan

Tabel 4.4 Pseucode penentuan jenis kendaraan

```
Allnode = {1,2,…,n}; Route ⊆ Allnode ; nonRoute ⊆ Allnode \ {Route}
Vehicle = {P,P-1,….1}; VehicleP>VehicleP-1>……>Vehicle1
TotVol = sum(Vol(nonRoute))
TotWeight= sum(Weight(nonRoute))
MaxDim=max(Dimension(nonRoute))

for i = P : 1 step -1
      if AND(TotVol>= VolVehicle P, TotWeight>= CapVehicle P, MaxDim>= LVehicle P)
          Select = Vehicle P
      elseif  AND(TotVol>= VolVehicle (i), TotWeight>= CapVehicle (i), MaxDim>= LVehicle (i))
          Select = Vehicle (i+1)
      elseif AND(TotVol < VolVehicle 1, TotWeight < CapVehicle 1, MaxDim < LVehicle 1)
          Select = Vehicle 1
       end
end
```

Jika sudah terdapat lebih dari satu rute terbentuk maka pembangkitan *vertex* dilakukan menggunakan probabilitas *vertex* yang belum muncul dari *vertex* terakhir pada masing-masing rute. Seperti pada Gambar 4.7 prioritas pertama adalah rute pertama, dimana *vertex* terakhir adalah *vertex* 2. Bangkitkan *vertex* baru menggunakan probabilitas *vertex 2* di baris ke-3 pada matriks probabilitas transisi menggunakan mekanisme *roulette wheels*. Jika *vertex* baru tidak fisibel untuk dimasukan ke dalam rute1 maka dibangkitkan *vertex* baru untuk rute 2. Probabilitas yang digunakan adalah probabilitas *vertex* 1 baris ke-2 karena pada rute2 *vertex* 1 adalah *vertex* yang terakhir di rute. Jalankan mekanisme *roulette wheels* untuk mendapatkan *vertex* baru. Jika tidak menghasilkan rute yang fisibel maka tambahkan kendaraan baru dengan mekanisme yang sudah dijelaskan sebelumnya. Pada kendaraan baru probabilitas yang digunakan adalah probabilitas *vertex* 0 di baris pertama pada matriks probabilitas transisi. Jalankan mekanisme *roulette wheels* untuk mendapatkan *vertex* baru pada kendaraan baru. Lakukan langkah-langkah ini hingga semua *vertex* sudah masuk ke dalam rute.



Gambar 4.7 Ilustrasi penggunaan matriks probabilitas transisi pada HFVRP

Setelah rute lengkap terbentuk langkah selanjutnya pada tahap inisialisasi solusi awal adalah memperbaiki rute yang terbentuk dengan prosedur 2-*opt*, 1-1 *exchange,* dan 1-0 *exchange*. Algoritma dari ketiga algoritma tersebut sudah dijelaskan pada Algoritma 1.5, 1.6 dan 1.7 pada bab 2. Ketiga prosedur tersebut adalah prosedur mutasi pada VRP, ketika mutasi terjadi algoritma *loading constraint* berjalan untuk memeriksa fisibitas dari rute yang termutasi.

4.3.2. Algoritma CEGA

Algoritma CEGA yang digunakan pada penelitian ini didasarkan pada penelitian Santosa dkk. (2011) dengan beberapa modifikasi. *Crossover* tidak dilakukan di penelitian ini. Hal ini dilakukan karena *crossover* akan sulit menghasilkan solusi yang fisibel untuk kasus 2L-HFVRP. Mekanisme *crossover* digantikan dengan pembangkitan sampel baru dengan mekanisme matriks probabilitas transisi. Penggantian dilakukan ketika suatu sampel dalam populasi tidak mengalami perubahan dalam tiga iterasi. Selama tiga iterasi nilai fungsi tujuan dari setiap iterasi akan dicatat untuk dibandingkan. Sampel dengan nilai fungsi tujuan yang sama selama 3 iterasi akan diganti dengan sampel baru dengan mekanisme matriks probabilitas transisi. Hal ini bertujuan untuk keluar dari lokal optimum. Penggantian ini menjadi masuk akal karena dua mekanisme tersebut sama-sama memunculkan individu baru yang didasarkan genetika sampel (orang tua) elit.

Populasi akan dimutasi dengan 3 jenis mutasi yang berbeda yaitu 2-*opt*, 1-1 *exchange*, dan 1-0 *exchange*. Setiap sampel dimutasi dengan jenis mutasi yang berbeda-beda. Penentuan jenis mutasi yang digunakan dilakukan dengan mekanisme *roulette wheels* dengan masing-masing jenis mutasi memiliki probabilitas yang sama.

Terdapat 2 kriteria pemberhentian iterasi yaitu jumlah maksimal iterasi dan nilai toleransi pemberhentian ε. Toleransi pemberhentian ε adalah selisih dari parameter permutasi pada iterasi saat ini $p_{m(it)}$ dengan pada iterasi sebelumnya $p_{m(it-1)}$. Nilai ε biasanya di tetapkan bernilai mendekati 0 yang berarti fungsi tujuan dari sampel elit bernilai sama antar iterasi karena terindikasi sudah mencapai optimum. Jika salah satu kriteria pemberhentian terpenuhi maka algoritma akan berhenti. Langkah-langkah diatas dijelaskan secara lebih rinci pada Algoritma 1.9.

Algoritma 1.9 Algoritma CEGA

1. Bangkitkan populasi sejumlah N
2. Hitung fungsi tujuan dari populasi yang terbentuk
3. Tetapkan sampel elit sejumlah ρ x N (N_{elite})
4. Jika terdapat sampel yang tidak berubah dalam tiga iterasi maka bangkitkan sampel baru sejumlah sampel tersebut dengan mekanisme matriks probabilitas transisi untuk mengganti sejumlah sampel di populasi. Matriks probabilitas transisi P_M akan dihitung setiap iterasi dengan rumus:
$$P_{M\,(it)} = \alpha\omega + (1-\alpha)P_{M\,(it-1)} \qquad (4.1)$$
5. Gantikan sampel yang tidak mengalami perubahan nilai fungsi tujuan dengan sampel baru dari langkah 4.
6. Lakukan mekanisme mutasi untuk masing-masing sampel
   a. Bangkitkan bilangan random r
   b. Jika $r \leq 0.33$ maka jalankan prosedur 2-*opt*
   c. Jika $0.33 < r \leq 0.66$ maka jalankan prosedur 1-1 *exchange*
   d. Jika $r > 0.66$ maka jalankan prosedur 1-0 *exchange*
   e. Ulangi langkah 6a sampai semua sampel selesai dimutasi

7. Update G_{best} jika sampel terbaik pada iterasi ini memiliki nilai fungsi tujuan yang lebih baik dibandingkan G_{best} sebelumnya.
8. Perbaharui parameter mutasi p_m menggunakan sampel elit dengaan persamaan (4.1)

   $$p_{m(it)} = \frac{A_{(it)}}{2} \tag{4.2}$$

   Dengan nilai $A_{(it)}$ didapatkan dari persamaan (4.3) dan persamaan (4.4)

   $$A_{(it)} = (1-\alpha)u + A_{(it-1)}\alpha \tag{4.3}$$

   $$u = \frac{\overline{z_e}}{2z_{best}} \tag{4.4}$$

9. Ulangi langkah 2 jika jumlah maksimum iterasi atau nilai toleransi pemberhentian ε belum terpenuhi

   $$\epsilon = \left|p_{m(it)} - p_{m(t-1)}\right| \tag{4.4}$$

   Jika maksimum iterasi atau nilai ε sudah terpenuhi maka iterasi akan berhenti dan solusi terbaik di iterasi terakhir akan dicatat sebagai output dari algoritma CEGA.

Keterangan:

$\overline{z_e}$: Rata-rata fungsi tujuan sampel elit

z_{best} : Solusi terbaik pada tiap iterasi

4.3.3. Contoh Numerik

Berikut adalah contoh numerik untuk memperjelas algoritma CEGA yang dijelaskan sebelumnya. Agar lebih mudah dimengerti maka akan digunakan contoh soal yang sederhana dan algoritma akan dijalankan sebanyak satu iterasi. Untuk menggambarkan penggantian oleh mekanisme matriks probabilitas transisi maka contoh numerik diasumsikan berada pada iterasi 4. Berikut adalah parameter yang digunakan dalam contoh numerik ini:

$N = 4; ; \alpha = 0.8; n = 6; A_0 = 2; \varepsilon = 0.01$

Iterasi 4

Populasi dari iterasi sebelumnya.

Sampel 1 : 0-5-3-6-0 $f = 170$
0-2-4-1-0

Sampel 2 : 0-6-5-0 $f = 150$
0-2-4-3-1-0

Sampel 3 : 0-3-2-6-1-0 $f = 120$
0-5-4-0

Sampel 4 : 0-5-1-0 $f = 190$
0-2-6-1-3-0

Nilai fungsi tujuan dalam 3 iterasi.

	Iterasi 1	Iterasi 2	Iterasi 3
Sampel 1	170	170	170
Sampel 2	175	164	150
Sampel 3	140	137	120
Sampel 4	190	190	190

$G_{best} = 120$

Langkah 3 : Tetapkan sampel elit sejumlah ρ x N $\rho = 20\%$

$size_{elite} = 20\% \times 4 \approx 1$

$POP_{elite} =$ Sampel 3 : 0-3-2-6-1-0 $f = 120$
0-5-4-0

Langkah 4 Karena sampel 1 dan sampel 4 selama tiga iterasi tidak mengalami perubahan nilai fungsi tujuan maka bangkitkan sampel baru sejumlah sampel sampel tersebut (2 sampel).

$POP_{elite} =$ 0-3-2-6-1-0

0-5-4-0

$$\omega = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Matriks ω hanya berisi $1/N_{elite}$ pada baris sebagai *vertex* asal dan kolom sebagai *vertex* tujuan pada POP_{elite}. Jadi jika pada POP_{elite} terdapat rute dari *vertex* 3 ke *vertex* 2 maka pada matriks ω baris *vertex* 2 (baris 3) dan kolom *vertex* 3 (baris 4) akan bernilai $1/size_{elite} = 1$.

Update nilai P_M :

$$P_{M\,(it)} = 0.8 \times \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} + (0.2) \times \begin{bmatrix} 0 & .17 & .17 & .17 & .17 & .17 & .17 \\ .17 & 0 & .17 & .17 & .17 & .17 & .17 \\ .17 & .17 & 0 & .17 & .17 & .17 & .17 \\ .17 & .17 & .17 & 0 & .17 & .17 & .17 \\ .17 & .17 & .17 & .17 & 0 & .17 & .17 \\ .17 & .17 & .17 & .17 & .17 & 0 & .17 \\ .17 & .17 & .17 & .17 & .17 & .17 & 0 \end{bmatrix}$$

$$P_{M\,(it)} = \begin{bmatrix} 0 & 0.03 & 0.03 & \mathbf{0.83} & 0.03 & \mathbf{0.83} & 0.03 \\ \mathbf{0.83} & 0 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0 & 0.03 & 0.03 & 0.03 & \mathbf{0.83} \\ 0.03 & 0.03 & \mathbf{0.83} & 0 & 0.03 & 0.03 & 0.03 \\ \mathbf{0.83} & 0.03 & 0.03 & 0.03 & 0 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0.03 & 0.03 & \mathbf{0.83} & 0 & 0.03 \\ 0.03 & \mathbf{0.83} & 0.03 & 0.03 & 0.03 & 0.03 & 0 \end{bmatrix}$$

Nilai $P_{M\,(it)}$ akan digunakan untuk membangkitkan individu baru untuk menggantikan sampel pada populasi. Dengan rumus diatas nilai $P_{M\,(it)}$ sudah berisi informasi dari sampel elit, sehingga probabilitas kemunculan pola rute seperti sampel elit akan tinggi.

Sampel baru 1 : 0-3-4-1-0 $f = 130$

0-2-5-6-0

Sampel baru 2 : 0-5-6-0 $f = 117$

0-3-2-4-1-0

Langkah 5 : Karena sampel 1 dan 4 adalah sampel yang tidak mengalami perubahan nilai fungsi tujuan dalam tiga iterasi maka sampel 1 dan sampel 4 akan diganti dengan sampel baru.

Sampel 1 : **0-3-4-1-0**

0-2-5-6-0

Sampel 2 : 0-6-5-0

0-2-4-3-1-0

Sampel 3 : 0-3-2-6-1-0

0-5-4-0

Sampel 4 : **0-5-6-0**

0-3-2-4-1-0

Langkah 6 : Mutasi

Untuk Sampel 1 bangkitkan bilangan random : $r = 0.34$

Sampel 1 dikenakan prosedur 1-1 *exchange* menjadi :

- 0-2-3-6-0 $f = 155$
- 0-5-4-1-0

Untuk Sampel 2 bangkitkan bilangan random : $r = 0.74$

Sampel 2 dikenakan prosedur 1-0 *exchange* menjadi :

- 0-3-6-5-0 $f = 147$
- 0-2-4-1-0

Untuk Sampel 3 bangkitkan bilangan random : $r = 0.81$

Sampel 3 dikenakan prosedur 2-*opt* menjadi :

- 0-3-1-6-2-0 $f = 116$
- 0-5-4-0

Untuk Sampel 4 bangkitkan bilangan random : $r = 0.12$

Sampel 4 dikenakan prosedur 1-0 *exchange* menjadi :

- 0-3-1-6-2-0 $f = 113$
- 0--4-5-0

Langkah 7 : Karena pada sampel 4 menghasilkan nilai fungsi tujuan yang lebih baik dibanding G_{best} maka lakukan update nilai G_{best} dengan nilai fungsi tujuan dari sampel 4. Sehingga $G_{best} = 113$.

Langkah 8 : Perbaharui parameter permutasi p_m

$$u = \frac{\overline{z_e}}{2z_{best}} = \frac{120}{2 \times 113} = 0.5309$$

$$A_{(it)} = (1-\alpha)u + A_{(it-1)}\alpha = (0.2 \times 0.5309) + (2 \times 0.8) = 1.70618$$

$$p_{m(it)} = \frac{A_{(it)}}{2} = \frac{1.7025}{2} = 0.85309$$

Langkah 9 : Hitung nilai ϵ

$$\epsilon = \left|p_{m(it)} - p_{m(t-1)}\right| = |0.85309 - 1| = 0.146$$

Karena nilai $\epsilon > 0.01$ maka ulangi langkah 2

Lakukan langkah-langkah diatas hingga kriteria pemberhentian terpenuhi dan catat nilai terbaik dari solusi terbaik di iterasi terakhir. Pada contoh numerik diatas dihasilkan 4 individu baru dengan beberapa individu menghasilkan nilai fungsi tujuan yang lebih baik. Tabel 4.5 menjelaskan perbandingan populasi awal dengan populasi baru hasil dari algoritma CEGA. Pada Tabel 4.5 mencatat bahwa terdapat 3 individu dalam populasi yang mengalami perbaikan.

Tabel 4.5 Perbandingan populasi iterasi 3 dan iterasi 4

	Iterasi 3		Iterasi 4	
	Rute	*f*	Rute	*f*
Individu 1	0-5-3-6-0 0-2-4-1-0	170	0-2-3-6-0 0-5-4-1-0	155
Individu 2	0-6-5-0 0-2-4-3-1-0	150	0-3-6-5-0 0-2-4-1-0	147
Individu 3	0-3-2-6-1-0 0-5-4-0	120	0-3-1-6-2-0 0-5-4-0	116
Individu 4	0-5-1-0 0-2-6-1-3-0	190	0-3-1-6-2-0 0-4-5-0	113

BAB 5
EKSPERIMEN DAN ANALISA

Bab ini akan menguraikan tentang mekanisme pelaksanaan eksperimen seperti proses validasi, penentuan parameter dan proses menjalankan algoritma usulan terhadap data set yang sudah dijelaskan sebelumnya. Bab ini juga akan menampilkan perbandingan hasil dengan beberapa penelitian sebelumnya dan analisa berdasarkan hasil perbandingan.

5.1. Verifikasi dan Validasi

Verifikasi adalah proses untuk memeriksa kesesuaian antara logika pada model (program computer) dan logika pada algoritma dan diagram alur (Hoover & Perry, 1989). Sederhananya adalah memeriksa kesalahan dalam program komputer yang dibuat berdasarkan algoritma yang digunakan. Validasi adalah proses untuk melihat apakah model yang merupakan konseptualisasi atau abstraksi, sudah merepresentasikan sistem nyata (Hoover & Perry, 1989). Dalam kasus ini sistem tersebut adalah 2L-HFVRP.

Pada penelitian ini verifikasi dan validasi dilakukan dengan mencocokkan output dari program dengan perhitungan manual. Output yang dihasilkan dari program adalah berupa rute, jumlah biaya perjalanan, jumlah dan jenis kendaraan yang digunakan dan koordinat peletakan muatan dalam kontainer. Program dikatakan valid jika rute yang dihasilkan menghasilkan total biaya yang sama dibanding dengan hasil dari hasil hitung secara manual dan tidak melanggar semua konstrain yang ada berupa kapasitas berat dan penataan dalam kontainer. Agar lebih mudah maka proses ini akan dilakukan menggunakan kasus yang paling sederhana pada data set yang digunakan yaitu pada data set *instance 1 class 2.* Pada *instance* tersebut terdiri dari 15 konsumen dengan 24 kotak barang yang harus diantar menggunakan 4 jenis kendaraan yang berbeda. Program dijalankan dengan nilai $\alpha = 0.2,\ \rho = 0.2,$ N=5 , $\epsilon = 1 \times 10^{-5}$ dan jumlah iterasi maksimum (*maxiter*) = 30 dengan versi *unrestricted*. Berikut adalah hasil output dari program yang dibuat.

Total biaya perjalanan (program) = 702.4458

Rute:

- 0 – 14 – 13 – 4 – 0 (*vehicle* 4)
- 0 – 12 – 15 – 10 – 9 – 0 (*vehicle* 4)
- 0 – 2 – 3 – 1 – 0 (*vehicle* 4)
- 0 – 5 – 11 – 6 – 0 (*vehicle* 4)
- 0 – 8 – 7 – 0 (*vehicle* 4)

Koordinat muatan dalam kontainer:

0	0	0	0	0
1	0	0	0	0
2	24	0	7	12
3	7	0	7	10
4	13	0	0	0
5	14	11	0	0
6	0	8	0	0
7	0	0	0	0
8	0	5	15	0
9	27	0	0	0
10	18	7	0	7
11	0	17	0	11
12	0	17	25	12
13	24	0	0	0
14	0	8	0	11
15	0	14	0	0

Tabel 5.1 Perhitungan manual total biaya perjalanan

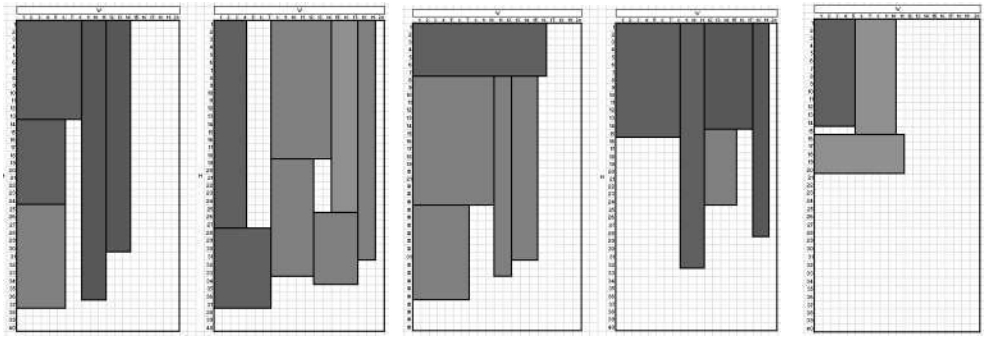
	Jarak	Vc	Biaya Variabel	Fc	Total Biaya
Vehicle 4	68.73349	1.5	103.1002	40	143.1002
Vehicle 4	75.76509	1.5	113.6476	40	153.6476
Vehicle 4	69.42267	1.5	104.1340	40	144.1340
Vehicle 4	58.56456	1.5	87.8468	40	127.8468
Vehicle 3	62.47807	1.5	93.71711	40	133.7171
			Total Biaya =		702.4458

Perbandingan dilakukan antara hasil perhitungan total biaya pada output program dengan hasil perhitungan manual. Berdasarkan hasil perhitungan manual pada Tabel 5.1 menunjukkan bahwa hasil perhitungan program sama dengan hasil perhitungan menggunakan program. Rute yang dihasilkan program tidak melanggar kapasitas berat dari jenis kendaraan yang digunakan seperti yang

diperlihatkan pada Tabel 5.2. Koordinat yang dihasilkan dari program akan digambarkan pada untuk melihat apakah ada pelanggaran terhadap kendala luas kontainer. Gambar 5.1 menggambarkan posisi dari muatan dalam kontainer. Dari Gambar 5.1 terlihat bahwa algoritma *packing heuristic* dapat berjalan dengan baik dengan menata muatan dalam kontainer. Koordinat penataan kontainer hasil output dari program setelah digambarkan tidak terjadi pelanggaran kendala, seperti saling bertumpuk atau melebihi kapasitas luas kontainer. Beberapa proses diatas dapat dikatakan bahwa program yang dibuat sudah lolos tahap verifikasi dan validasi.

Tabel 5.2 Perhitungan muatan dalam kendaraan.

Jenis Kendaraan	Kapasitas Berat	Berat Muatan
Vehicle 4	60	53
Vehicle 4	60	55
Vehicle 4	60	53
Vehicle 4	60	55
Vehicle 4	40	42



Gambar 5.1 Posisi peletakan muatan dalam kontainer

5.2. Data Set

Dalam melakukan eksperimen digunakan data set yang di dalamnya merupakan sekumpulan kasus VRP yang digunakan dalam menguji algoritma CEGA. Data set yang digunakan terdiri dari 15 *instance* dengan masing-masing

instance terdiri dari 5 kelas. Kelas pertama merupakan kasus CVRP karena dimensi barang yang menjadi *demand* konsumen bernilai 1. Kelas 2-5 merupakan kasus 2L-CVRP. Pada setiap *instance* memiliki jumlah dan koordinat konsumen yang sama, yang membedakan adalah jumlah dan luas barang yang harus diangkut. Semakin tinggi kelas maka akan semakin banyak jumlah barang yang harus diangkut, namun tidak pada luas dari barang tersebut. Beberapa kasus memiliki jumlah barang (it) yang lebih banyak dibandingkan kelas sebelumnya tapi dengan total luas barang ($r\%$) yang lebih kecil (lihat Gambar 4.1). Dari *instance* 1-15 memiliki jumlah konsumen yang semakin banyak.

5.3. Penentuan Parameter

Algoritma CEGA yang diusulkan menggunakan beberapa parameter yang harus ditentukan agar algoritma dapat dijalankan dengan baik. Parameter tersebut antara lain adalah:

N: Jumlah individu yang ada dalam populasi (ukuran populasi)
α: Koefisien penghalusan
ρ: Rasio sampel elit
ϵ: Kriteria pemberhentian
$maxiter$: Jumlah maksimum iterasi

Jumlah *N* akan mempengaruhi banyaknya individu dalam populasi yang bertugas dalam mencari solusi. Semakin banyak *N* semakin tinggi kemungkinan untuk mendapatkan solusi yang baru, karena semakin banyak yang bertugas dalam mencari solusi. Namun semakin banyak nilai *N* maka waktu komputasi akan menjadi lebih lama karena dalam MATLAB perhitungan akan dilakukan secara runtut dan satu per satu untuk setiap individu. Sehingga pada penelitian ini nilai N yang digunakan adalah 5. Rasio sampel elit (ρ) biasanya berkisar antara 1%-10% (Santosa et al., 2011), namun pada penelitian ini nilai ρ yang digunakan adalah 2%. Nilai parameter lainnya dicari dengan melakukan pengujian dengan melakukan beberapa percobaan dengan kombinasi tertentu. Konvergensi dan Divergensi menjadi indikator dalam menentukan nilai parameter yang akan digunakan. Konvergensi adalah suatu situasi yang menggambarkan kemampuan algoritma

dalam menghasilkan solusi yang mendekati solusi optimal. Divergensi adalah suatu situasi yang menggambarkan penyimpangan antara solusi yang dihasilkan algoritma dibanding dengan solusi optimal. Pengujian parameter dilakukan untuk dua jenis kasus yang berbeda yaitu 2L-CVRP dan 2L-HFVRP. Pada kasus CVRP murni akan menggunakan parameter yang sama dengan 2L-CVRP karena yang berbeda antar keduanya adalah pada CVRP dimensi barang bernilai 1. Tabel 5.3 dan Tabel 5.4 adalah hasil dari pengujian parameter untuk kasus 2L-HFVRP dan 2L-CVRP. Setiap kombinasi parameter dilakukan 5 kali replikasi pencarian solusi. Berdasarkan pengertian dari konvergensi maka dalam perhitungannya, konvergensi didapatkan dengan membagi selisih antara rata-rata solusi dan solusi optimal dengan solusi optimal. Sedangkan Divergensi dihitung dengan mengurangi solusi terburuk dalam 5 replikasi dengan solusi optimal.

Tabel 5.3 Pengujian parameter untuk kasus 2L-HFVRP

							Best Know Solution			603.15	
	α	ϵ	max iter	Rep 1	Rep 2	Rep 3	Rep 4	Rep 5	Rata2	Konvergensi	Divergensi
1	0.8	0.001	30	609.81	624.47	641.24	610.27	640.51	625.26	3.67%	38.09
2	0.8	0.001	60	625.46	644.42	633.56	622.72	616.29	628.49	4.20%	41.27
3	0.8	0.0001	30	602.88	644.60	622.46	610.19	604.26	616.88	2.28%	41.45
4	0.8	0.0001	60	611.99	625.46	609.81	617.10	596.31	612.13	1.49%	22.31
5	0.8	0.00001	30	609.81	602.88	604.26	604.26	604.26	605.10	0.32%	6.659
6	0.8	0.00001	60	604.26	604.26	602.88	609.81	609.81	606.69	0.59%	6.659
7	0.5	0.001	30	622.72	627.01	602.88	604.26	609.81	613.34	1.69%	23.86
8	0.5	0.001	60	633.16	626.85	604.26	637.10	617.10	623.69	3.41%	33.95
9	0.5	0.0001	30	609.81	610.19	604.26	610.19	604.26	607.74	0.76%	7.037
10	0.5	0.0001	60	610.19	611.99	604.26	614.85	604.26	609.11	0.99%	11.7
11	0.5	0.00001	30	604.26	609.81	604.26	604.26	604.26	605.37	0.37%	6.659
12	0.5	0.00001	60	609.81	604.26	604.26	604.26	604.26	605.37	0.37%	6.659
13	0.2	0.001	30	648.94	645.65	609.81	609.81	604.26	617.38	2.36%	45.79
14	0.2	0.001	60	657.54	638.13	629.35	655.76	625.46	641.25	6.32%	54.39
15	0.2	0.0001	30	638.04	610.19	626.88	631.80	649.59	631.30	4.67%	46.44
16	0.2	0.0001	60	638.07	610.19	604.26	622.38	626.88	620.36	2.85%	34.92
17	0.2	0.00001	30	610.19	615.13	622.72	637.38	611.99	619.48	2.71%	34.23
18	0.2	0.00001	60	604.26	604.26	626.88	604.26	604.26	608.79	0.93%	23.73

Berdasarkan hasil pengujian pada Tabel 5.3 dapat dilihat bahwa kombinasi parameter $\alpha = 0.8$, $\epsilon = 0.00001$ dan *maxiter* = 30 adalah yang terbaik dari 18 kombinasi yang dicoba. Menjadi kombinasi terbaik karena memiliki nilai konvergensi dan divergensi yang paling kecil. Begitu pula pada Tabel 5.4 kombinasi parameter $\alpha = 0.2$, $\epsilon = 0.00001$ dan *maxiter* = 60 yang dipilih menjadi kombinasi terbaik untuk digunakan dalam eksperimen algoritma. Kombinasi tersebut berhasil menghasilkan nilai konvergensi dan divergensi yang baik.

Tabel 5.4 Pengujian parameter untuk kasus 2L-CVRP

							Best Know Solution			298.34	
	α	ϵ	maxiter	Rep 1	Rep 2	Rep 3	Rep 4	Rep 5	Rata2	Konvergensi	Divergensi
1	0.8	0.001	30	305.77	392.01	304.13	343.67	308.56	330.83	12.3%	97.527
2	0.8	0.001	60	346.13	404.39	306.94	313.12	321.53	338.42	14.9%	109.91
3	0.8	0.0001	30	321.71	434.81	330.35	319.58	295.17	340.32	15.6%	140.33
4	0.8	0.0001	60	332.74	315.68	284.52	351.22	350.32	326.90	11.0%	56.74
5	0.8	0.00001	30	305.19	331.09	342.33	364.55	374.68	343.57	16.7%	80.19
6	0.8	0.00001	60	329.11	298.29	350.23	415.38	318.21	345.53	17.3%	120.90
7	0.5	0.001	30	399.47	302.66	394.41	366.12	309.57	354.45	20.4%	104.98
8	0.5	0.001	60	327.51	314.15	302.66	330.60	303.56	312.75	6.2%	36.12
9	0.5	0.0001	30	362.75	396.27	382.97	391.76	319.61	370.67	25.9%	101.79
10	0.5	0.0001	60	335.00	290.85	395.33	333.20	295.17	329.91	12.0%	100.84
11	0.5	0.00001	30	302.63	421.64	382.81	309.19	328.74	349.00	18.5%	127.16
12	0.5	0.00001	60	316.30	359.63	336.00	311.09	319.09	328.42	11.5%	65.14
13	0.2	0.001	30	392.72	303.69	309.03	341.99	305.98	315.17	7.0%	98.23
14	0.2	0.001	60	364.28	356.03	356.03	300.39	290.85	333.52	13.3%	69.79
15	0.2	0.0001	30	298.29	351.65	321.39	315.49	328.92	323.15	9.7%	57.17
16	0.2	0.0001	60	337.02	290.85	290.85	307.45	326.52	310.54	5.5%	42.53
17	0.2	0.00001	30	290.85	312.60	306.75	306.59	324.86	308.33	4.7%	30.37
18	0.2	0.00001	60	320.61	320.41	299.51	284.52	307.45	306.50	4.1%	26.12

Pada Tabel 5.3 dan 5.4 diperjelas dengan gradien warna hijau ke putih ke merah. Warna hijau melambangkan nilai paling kecil dan merah melambangkan nilai paling besar pada suatu kolom. Sehingga mudah untuk dilihat bahwa algoritma ini membutuhkan iterasi yang panjang untuk mendapatkan solusi yang baik. Hal ini terlihat bahwa nilai konvergensi dan divergensi semakin kecil dengan

meningkatkan nilai *maxiter* dan semakin kecil nilai ϵ. Namun memperpanjang iterasi menjadi tidak efektif ketika menggunakan nilai α yang besar karena akan menyebabkan algoritma memiliki kecenderungan terjebak dalam lokal optimum. Jadi jika menggunakan nilai α besar maka algoritma dimungkinkan mendapatkan solusi yang mendekati optimal namun juga memiliki kemungkinan terjebak dalam lokal optimum. Pencarian akan semakin menjadi luas dengan menggunakan nilai α yang kecil namun membutuhkan jumlah iterasi yang lebih panjang.

5.4. Hasil Eksperimen

Eksperimen dilakukan dengan 3 kasus yang berbeda yaitu CVRP, 2L-CVRP dan 2L-HFVRP. Masing-masing kasus 2L-CVRP dan 2L-HFVRP dibagi menjadi dua versi yaitu *sequential* dan *unrestricted*. Algoritma ditulis dan dijalankan dalam program MATLAB 7.10 dengan komputer prosesor Core i3-5015U 2.10 GHz dengan RAM 4GB pada sistem operasi Windows 10.

Tabel 5.5 Perbandingan hasil untuk kasus CVRP

						SA_HLS	GTS	ACO	G x E
Inst	CEGA	SA_HLS	GTS	ACO	G x E	*%gap*	*%gap*	*%gap*	*%gap*
1	278.73	278.73	278.73	278.73	278.73	0.0%	0.0%	0.0%	0.0%
2	334.96	334.96	334.96	334.96	334.96	0.0%	0.0%	0.0%	0.0%
3	360.09	358.4	358.4	358.4	358.4	-0.5%	-0.5%	-0.5%	-0.5%
4	430.88	430.89	430.89	430.89	430.89	0.0%	0.0%	0.0%	0.0%
5	375.28	375.28	375.28	375.28	375.28	0.0%	0.0%	0.0%	0.0%
6	495.85	495.85	495.85	495.85	495.85	0.0%	0.0%	0.0%	0.0%
7	570.15	568.56	568.56	568.56	568.56	-0.3%	-0.3%	-0.3%	-0.3%
8	574.95	568.56	568.56	568.56	568.56	-1.1%	-1.1%	-1.1%	-1.1%
9	608.89	607.65	607.65	607.65	607.65	-0.2%	-0.2%	-0.2%	-0.2%
10	536.86	535.8	535.8	535.8	535.8	-0.2%	-0.2%	-0.2%	-0.2%
11	516.24	505.01	505.01	505.01	505.01	-2.2%	-2.2%	-2.2%	-2.2%
12	618.71	610	610	610	610	-1.4%	-1.4%	-1.4%	-1.4%
13	2081.20	2006.34	2006.34	2006.34	2006.34	-3.7%	-3.7%	-3.7%	-3.7%
14	846.73	837.67	837.67	837.67	837.67	-1.1%	-1.1%	-1.1%	-1.1%
15	867.59	837.67	837.67	837.67	837.67	-3.6%	-3.6%	-3.6%	-3.6%
				Rata-rata %gap		-1.0%	-1.0%	-1.0%	-1.0%

5.4.1. Hasil untuk CVRP

Pada data set class 1 adalah untuk kasus CVRP murni. Hasil dari algoritma CEGA dibandingkan dengan hasil penelitian sebelumnya yaitu algoritma SA_HLS (Leung *et al.,* 2013), GTS (Zachariadis *et al.,* 2009), ACO (Fuellerer *et al.,* 2009), dan GRAPS x ELS (Duhamel *et al.*, 2011). Tabel 5.5 menampilkan perbandingan hasil dari setiap algoritma untuk kasus CVRP. Kolom %gap memperlihatkan persentase selisih dari algoritma CEGA terhadap 3 algoritma pembanding. Nilai %gap dapat dihitung dengan mengurangkan hasil algoritma pembanding dengan hasil algoritma CEGA lalu dibagi dengan hasil algoritma pembanding. Dilihat dari solusi yang dihasilkan pada algoritma CEGA mampu bersaing dengan 3 algoritma pembanding. Dengan kasus-kasus dengan jumlah konsumen yang sedikit algoritma CEGA mampu menghasilkan solusi yang optimal. Algoritma memiliki rata-rata $\%gap = -1$ untuk masing- masing algoritma pembanding. Hal itu berarti bahwa rata-rata hasil CEGA memiliki selisih lebih besar sebesar 1%.

5.4.2. Hasil untuk 2L-CVRP

Kasus 2L-CVRP menggunakan beberapa penelitian sebagai pembanding. Penelitian sebelumnya yang digunakan sebagai pembanding antara lain algoritma SA_HSL (Leung et al., 2013), GTS (Zachariadis et al., 2009), dan ACO (Fuellerer et al., 2009) untuk yang versi *sequential* sedangkan untuk versi *unrestricted* ditambahkan algoritma GRAPS x ELS (Duhamel et al., 2011).

Tabel 5.6 Rata-rata hasil komputasi kelas 2-5 untuk 2L-CVRP *sequential*

Inst	Rata-rata Class 2-5				% gap		
	CEGA	SA_HSL	GTS	ACO	SA_HSL	GTS	ACO
1	299.05	298.34	304.22	294.48	-0.24%	1.70%	-1.55%
2	347.71	345.23	346.71	345.24	-0.72%	-0.29%	-0.72%
3	392.23	390.78	393.35	381.4	-0.37%	0.28%	-2.84%
4	444.28	444.21	444.62	441.11	-0.02%	0.08%	-0.72%
5	389.73	387.59	396.36	382.39	-0.55%	1.67%	-1.92%
6	498.86	503.66	505.04	499.49	0.95%	1.22%	0.13%
7	712.71	719.95	723.83	702.27	1.01%	1.54%	-1.49%
8	723.21	716.62	715.72	711.65	-0.92%	-1.05%	-1.62%
9	618.26	621.23	622.2	614.54	0.48%	0.63%	-0.60%

Tabel 5.6 Rata-rata hasil komputasi kelas 2-5 untuk 2L-CVRP *sequential* (lanjutan)

10	718.42	727.73	727.86	697.2	1.28%	1.30%	-3.04%
11	765.38	761.25	768.15	728.61	-0.54%	0.36%	-5.05%
12	623.68	618.98	628.62	615.76	-0.76%	0.79%	-1.29%
13	2664.35	2641.36	2679.34	2591.77	-0.87%	0.56%	-2.80%
14	1093.85	1079.47	1092.78	1042.33	-1.33%	-0.10%	-4.94%
15	1239.35	1230.28	1234.21	1212.93	-0.74%	-0.42%	-2.18%
			Rata-rata%Gap		-0.22%	0.55%	-2.04%

Tabel 5.6 dan Tabel 5.6 menampilkan hasil rata-rata dari setiap algoritma. Algoritma CEGA dibandingkan dengan masing-masing algoritma pembanding. Pada Tabel 5.6 rata-rata %gap yang dihasilkan untuk masing-masing algoritma SA_HSL, GTS, dan ACO adalah -0,22% ; 0,55% ; dan -2,04%. Jika dilihat dari %gap dari masing-masing algoritma pembanding maka secara rata-rata keseluruhan solusi yang dihasilkan CEGA untuk kasus 2L-CVRP *sequential* lebih baik dibanding dengan algoritma GTS. Untuk *instance* no 6, 7, 9, dan 10, algoritma CEGA mampu menghasilkan rata-rata solusi yang lebih baik dibandingkan algoritma SA_HLS dan *instance* no 6 untuk algoritma ACO. Sedangkan untuk 2L-CVRP *unrestricted* algoritma CEGA menghasilkan solusi dengan %gap masin-masing -1,95%, -1,01%, -3,02%, dan -3,87% untuk algoritma SA_HSL, GTS, ACO, dan GRASP x ESP. Sehingga dapat dikatakan secara rata-rata solusi yang dihasilkan algoritma CEGA tidak lebih baik dibandingkan keempat algoritma pembanding untuk kasus 2L-CVRP *unrestricted*. Dibandingkan dengan algoritma CEGA beberapa *instance* (1, 3, 4, 8, dan 9) memiliki rata-rata yang lebih baik dibandingkan algoritma GTS dan *instance* no 8 untuk algoritma SA_HSL.

Tabel 5.7 Rata-rata hasil komputasi kelas 2-5 untuk 2L-CVRP *unrestricted*

Inst	Rata-rata Class 2-5					% gap			
	CEGA	SA_HSL	GTS	ACO	G x E	SA_HSL	GTS	ACO	G x E
1	291.80	290.37	295.74	285.77	282.65	-0.49%	1.33%	-2.11%	-3.24%
2	344.49	341.35	341.89	341.02	339.26	-0.92%	-0.76%	-1.02%	-1.54%
3	380.20	377.37	384.49	376.32	376.32	-0.75%	1.11%	-1.03%	-1.03%
4	439.31	435.01	441.45	438.65	435.01	-0.99%	0.49%	-0.15%	-0.99%
5	386.45	379.49	382.22	379.03	379.03	-1.83%	-1.11%	-1.96%	-1.96%
6	502.53	501.02	499.47	497.27	497.04	-0.30%	-0.61%	-1.06%	-1.11%
7	705.74	698.65	703.49	696.91	691.11	-1.02%	-0.32%	-1.27%	-2.12%
8	696.77	703.13	705.6	691.14	678.84	0.90%	1.25%	-0.82%	-2.64%
9	613.17	612.02	615.65	612.02	612.01	-0.19%	0.40%	-0.19%	-0.19%
10	716.68	701.61	713	682.53	675.79	-2.15%	-0.52%	-5.00%	-6.05%
11	765.83	736.30	740.04	721.82	705.95	-4.01%	-3.49%	-6.10%	-8.48%
12	636.95	613.94	616.83	611.26	611.26	-3.75%	-3.26%	-4.20%	-4.20%
13	2667.53	2561.40	2599.4	2520.73	2490.62	-4.14%	-2.62%	-5.82%	-7.10%
14	1081.90	1015.81	1036.77	991.26	984.42	-6.51%	-4.35%	-9.14%	-9.90%
15	1230.38	1193.97	1197.83	1167.28	1144.69	-3.05%	-2.72%	-5.41%	-7.49%
				Rata-rata %Gap		-1.95%	-1.01%	-3.02%	-3.87%

5.4.3. Hasil untuk 2L-HFVRP

Pada kasus 2L-HFVRP algoritma SA_HSL digunakan sebagai pembanding. Hal ini dikarenakan belum banyak penelitian yang membahas 2L-HFVRP. Pertama dilakukan oleh Leung *et al.* (2013) dengan algoritma SA_HSL. Perbandingan hasil kedua algoritma untuk versi *sequential* dan *unrestricted* dapat dilihat pada Tabel 5.8 dan Tabel 5.9.

Kasus 2L-HFVRP versi *sequential* menghasilkan %gap sebesar -4,78% yang berarti rata-rata hasil algoritma CEGA memiliki selisih lebih besar sebesar 4,78% dibanding algoritma SA_HSL. Pada kasus 2L-HFVRP versi *unrestricted* menghasilkan %gap sebesar -5,68% yang berarti rata-rata hasil algoritma CEGA memiliki selisih lebih besar sebesar 5,68% dibanding algoritma SA_HSL. *Instance* no 7, 8, 10, 11, dan 13untuk kasus 2L-HFVRP *sequential* dan *Instance* no 7, 8, 10, 11, 13, dan 15 memiliki %gap yang cukup besar. Namun untuk *instance* lainnya memiliki %gap yang relatif kecil. Analisis dari hasil akan dibahas pada sub-bab berikutnya. *Sec h* adalah waktu dimana solusi terbaik terakhir diperbaharui dan *Sec tot* adalah waktu keseluruhan dalam menjalankan algoritma.

Tabel 5.8 Rata-rata hasil komputasi kelas 2-5 untuk 2L-HFVRP *sequential*

Inst	Rata-rata Class 2-5		Sec h		Sec tot		% gap
	CEGA	SA_HSL	CEGA	SA_HSL	CEGA	SA_HSL	
1	607.786	603.150	75.80	5.73	121.05	31.04	-0.77%
2	719.525	705.030	52.27	6.09	104.53	31.28	-2.06%
3	777.527	771.810	189.33	10.35	232.03	36.61	-0.74%
4	710.593	704.870	198.91	8.71	339.10	35.19	-0.81%
5	799.536	802.560	410.03	9.35	546.85	27.62	0.38%
6	841.512	834.760	311.91	9.92	393.03	43.9	-0.81%
7	6539.450	5770.830	392.79	1.95	718.87	31.4	-13.32%
8	6976.225	5633.040	353.24	4.77	547.51	37.42	-23.84%
9	1055.125	1047.600	383.43	15.74	481.92	68.84	-0.72%
10	8477.200	7730.730	976.95	5.87	1363.35	51.78	-9.66%
11	9243.800	8491.940	917.17	10.07	1540.98	58.88	-8.85%
12	1668.625	1681.610	134.86	32.9	323.48	158.45	0.77%
13	29007.750	26761.400	1824.07	6.71	2070.17	65.34	-8.39%
14	11411.250	11120.300	1053.74	11.29	1530.83	55.62	-2.62%
15	11949.500	11916.300	1455.51	24.53	1686.65	148.5	-0.28%
					Rata-rata %Gap		-4.78%

Tabel 5.9 Rata-rata hasil komputasi kelas 2-5 untuk 2L-HFVRP *unrestricted*

Inst	Rata-rata Class 2-5		Sec h		Sec tot		% gap
	CEGA	SA_HSL	CEGA	SA_HSL	CEGA	SA_HSL	
1	607.49	600.77	79.68	4.52	118.33	29.89	-1.12%
2	711.71	699.21	78.09	5.2	113.60	32.88	-1.79%
3	768.79	770.12	159.69	10.37	274.36	33.84	0.17%
4	706.37	698.19	201.65	7.73	344.50	30.04	-1.17%
5	789.95	786.84	504.02	7.82	563.99	27.68	-0.39%
6	831.90	831.32	329.04	8.9	429.64	42.78	-0.07%
7	6488.88	5630.02	688.04	1.15	948.80	31.08	-15.25%
8	6713.35	5602.60	835.47	1.92	875.24	30.22	-19.83%
9	1032.85	1035.62	687.18	7.88	1051.53	58.75	0.27%
10	8620.80	7625.05	1618.99	6.15	1813.64	43.27	-13.06%
11	9235.10	8329.69	1325.60	4.91	1628.83	52.61	-10.87%
12	1678.90	1681.07	259.82	34.91	306.83	167.21	0.13%
13	29508.75	25978.70	1983.75	9.35	2339.05	69.95	-13.59%
14	11196.75	10869.10	1015.75	16.31	1613.35	78.1	-3.01%
15	12136.00	11490.10	1174.91	8.88	1647.61	91.53	-5.62%
					Rata2 %Gap		-5.68%

5.5. Analisis Hasil

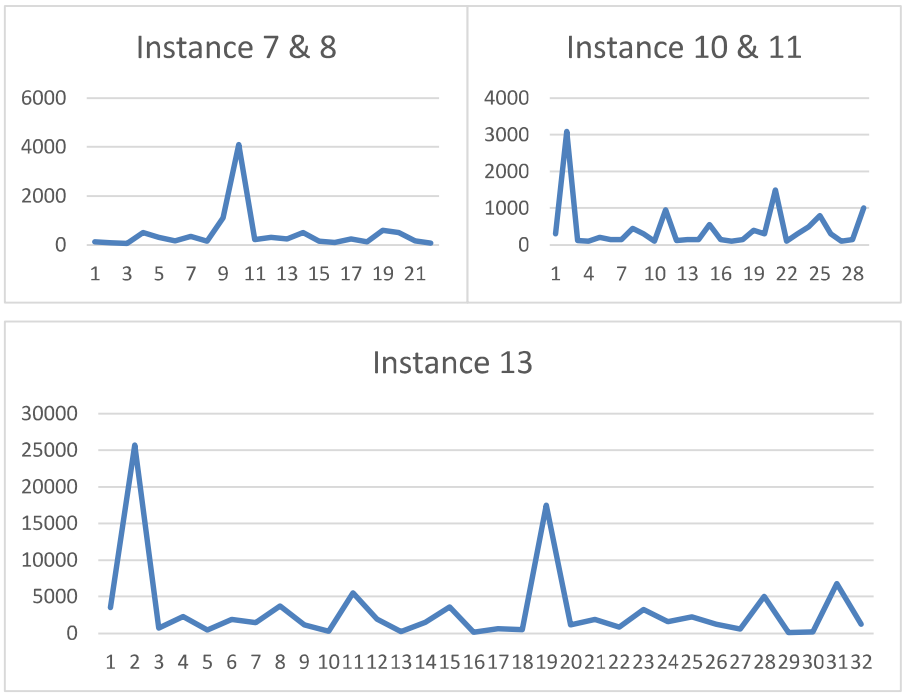
Algoritma CEGA dalam penelitian ini dalam menemukan solusi dipengaruhi oleh dua faktor yaitu nilai α dan jumlah iterasi. Jumlah iterasi dapat dibatasi dengan *maxiter* dan nilai ϵ. Nilai ϵ dalam CEGA mempengaruhi jumlah iterasi yang terjadi karena biasanya solusi belum mencapai nilai yang mendekati optimal namun nilai ϵ sudah melewati batas sehingga algoritma harus berhenti. Secara umum jika algoritma CEGA menggunakan nilai α yang kecil maka dibutuhkan jumlah iterasi yang lebih banyak untuk dapat menghasilkan solusi optimal. Menggunakan nilai α yang besar tidak selalu menjadi solusi yang baik, karena dengan nilai α yang besar beresiko untuk terjebak di lokal optimum. Semakin besar masalah yang ditangani membutuhkan jumlah iterasi yang besar dengan menggunakan nilai α yang besar juga semakin besar juga resiko untuk terjebak dalam lokal optimum.

Dilihat dari rata-rata %gap dari 3 jenis kasus yang diselesaikan dengan algoritma CEGA terlihat bahwa kasus 2L-HFVRP memiliki rata-rata %gap yang paling tinggi. Namun algoritma CEGA mampu menyelesaikan kasus CVRP dan 2L-CVRP dengan baik. Jika dilihat gambar 5.2 kasus 2L-HFVRP merupakan kasus 2L-CVRP dengan menambahkan variasi berupa jenis kendaraan yang berbeda. Di dalamnya terdapat mekanisme untuk menentukan jenis dan jumlah kendaraan yang digunakan. Sehingga nilai %gap yang tinggi pada kasus 2L-HFVRP dipengaruhi mekanisme tersebut, karena pada kasus 2L-CVRP nilai %gap relatif kecil.

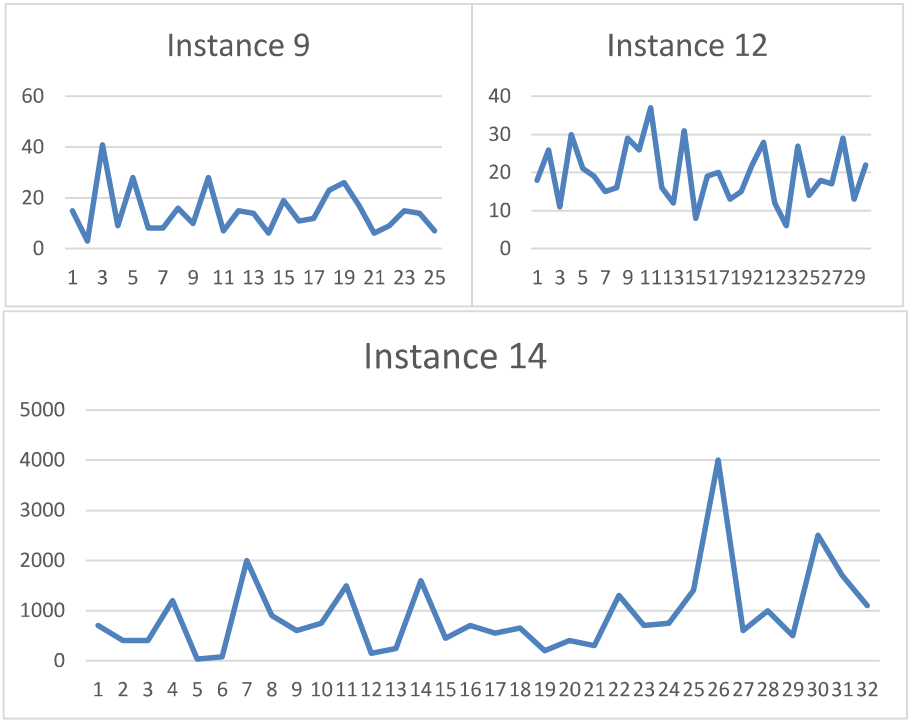
Gambar 5.2 Ilustrasi varian VRP

Namun perlu diketahui bahwa tidak semua contoh kasus dalam 2L-HFVRP menghasilkan solusi dengan %gap yang tinggi. Pada Tabel 5.8 dan 5.9 tercatat

bahwa *instance* no 7, 8, 10, 11, dan 13 menghasilkan nilai %gap yang tinggi. Pada *instance* tersebut terdapat beberapa konsumen dengan berat pesanan yang jauh lebih tinggi dibanding rata-rata konsumen lainnya dan berat pesanan dari beberapa konsumen tersebut hampir memenuhi 1 jenis kendaran dengan kapasitas terbesar. Jenis permasalahan tersebut memerlukan jenis kendaraan yang bervariasi jenisnya. Sedangkan mekanisme pemilihan jenis kendaraan yang digunakan dalam penelitian ini cenderung memilih jenis kendaraan dengan kapasitas yang tertinggi. Sehingga pada beberapa *instance* tersebut menghasilkan %gap yang tinggi. Ilustrasi berat pesanan setiap konsumen pada *instance* 7, 8, 10, 11, dan 13 dapat dilihat pada Gambar 5.3. Pada Gambar tersebut terlihat bahwa terdapat konsumen dengan berat pesanan sangat tinggi dibanding konsumen lainnya. Berbeda seperti *instance* lainnya yang tergambarkan pada Gambar 5.4. Pada Gambar 5.4 berat dari masing-masing konsumen tidak ada yang mendominasi.

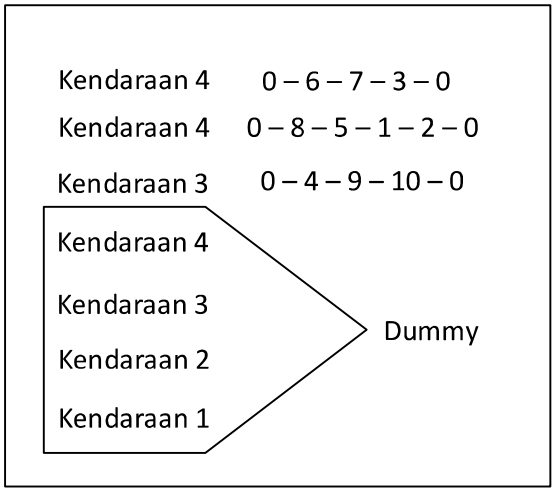


Gambar 5.3 Grafik berat pesanan setiap konsumen (1)

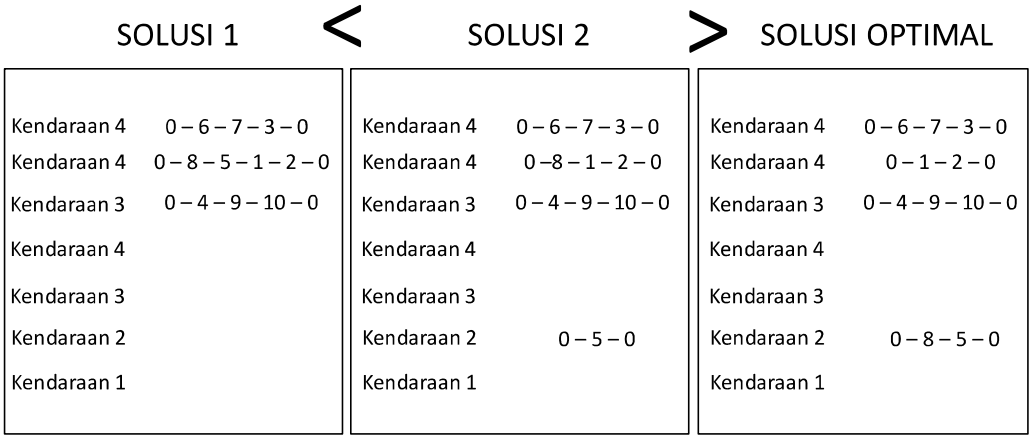


Gambar 5.4 Grafik berat pesanan setiap konsumen (2)

Untuk meningkatkan keterpilihan dari jenis kendaraan dengan kapasitas yang kecil dapat dilakukan dengan *dummy*. Jadi setiap solusi baru terbentuk dari mekanisme matriks probabilitas transisi ditambahkan masing-masing 1 kendaraan dari jenis kendaraan yang tersedia seperti yang digambarkan pada Gambar 5.5. Namun solusi yang dihasilkan tidak lebih baik. Hal ini dikarenakan mekanisme *local improvement* yang mensyaratkan solusi yang lebih baik agar mutasi dapat terjadi. Syarat tersebut justru membatasi ruang gerak dalam melakukan pencarian solusi.



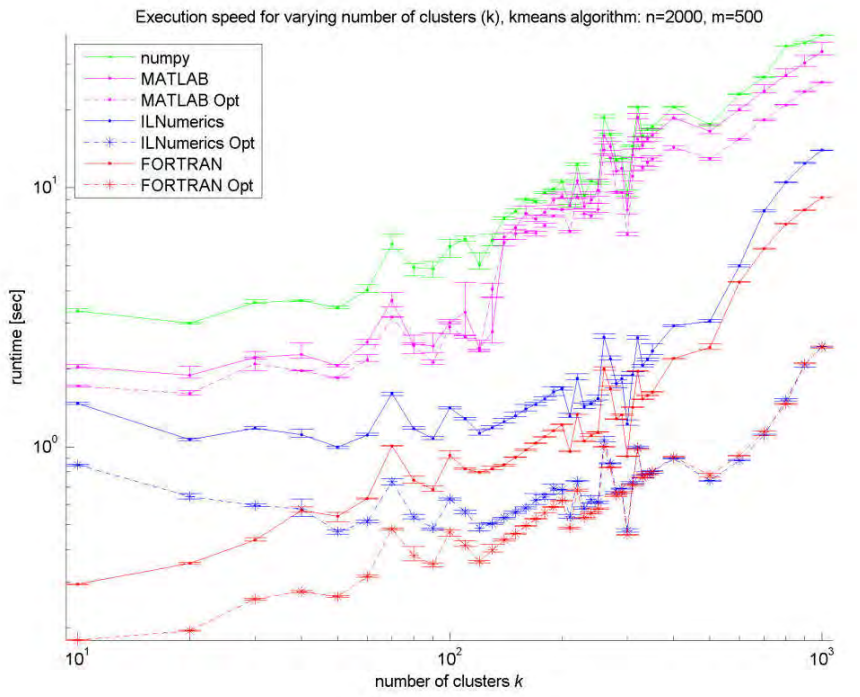
Gambar 5.5 Ilustrasi penambahan *dummy* kendaraan dalam rute



Gambar 5.6 Ilustrasi pencarian solusi

Algoritma *local improvement* adalah mutasi 1 langkah. Jadi setiap iterasi dalam algoritma *local improvement* hanya memindah 1 *node* atau menukar 2 *node* ke posisi baru. Dengan syarat mutasi yang dimilikinya maka solusi optimal yang digambarkan pada Gambar 5.6 tidak mungkin terjadi walaupun menggunakan *dummy*. Seperti pada Gambar 5.6 untuk mendapatkan solusi optimal maka *node* 5 dan 8 harus berpindah ke kendaraan 2. Namun *local improvement* hanya mampu

memindahkan node tersebut satu-persatu. Jika hanya memindahkan 1 *node*, solusi yang dihasilkan lebih besar dibanding sebelumnya maka 2 *node* tersebut tidak bisa berpindah ke kendaraan 2 untuk mendapatkan solusi yang optimal.



Gambar 5.6 Grafik waktu komputasi antar bahasa pemrograman (Haymo, 2016)

Waktu komputasi untuk semua kasus pada penelitian ini tidak dapat dibandingkan karena menggunakan *software*/bahasa pemrograman yang berbeda dalam membangun program. Pada penelitian ini menggunakan MATLAB yang menurut Haymo (2016) dalam laman ilnumerics.net menjelaskan bahwa perhitungan MATLAB lebih lama dibandingkan beberapa bahasa pemrograman lainnya. Pada Gambar 5.6 memperlihatkan bahwa program yang ditulis dengan MATLAB memiliki waktu komputasi yang lebih lama dibandingkan menggunakan bahasa pemrograman C# (ILNumerics). Pada penelitian sebelumnya algoritma

dituliskan dalam bahasa pemrograman C# sehingga menjadi masuk akal jika waktu komputasi penelitian ini jauh lebih lama dibandingkan penelitian sebelumnya.

BAB 6
KESIMPULAN DAN SARAN

6.1. Kesimpulan

Kesimpulan didapatkan berdasarkan analisis yang sudah dilakukan berdasarkan hasil eksperimen yang sudah dilakukan untuk menjawab rumusan masalah pada penelitian ini. Kesimpulan yang diperoleh adalah sebagai berikut:

1. Algoritma CEGA berhasil dikembangkan untuk menyelesaikan kasus *Two Dimensional Loading Constraint Heterogeneous Fleet Vehicle Routing Problem.*
2. Algoritma CEGA secara umum mampu menyelesaikan kasus CVRP dan 2L-CVRP dengan baik dan 2L-HFVRP dengan beberapa catatan. Sehingga dapat dikatakan algoritma mampu bersaing dengan algoritma pada penelitian sebelumnya.
3. Algoritma CEGA membutuhkan jumlah iterasi yang banyak jika menggunakan nilai α kecil dan akan beresiko terjebak di *local optimum* jika menggunakan nilai α yang besar.
4. Mekanisme penentuan jenis kendaraan cenderung untuk memilih kendaraan dengan kapasitas yang besar, pencarian kemungkinan solusi menjadi terbatas.
5. *Local improvement* terbukti memiliki kemampuan yang bagus dalam melakukan perbaikan solusi namun syarat terjadinya mutasi membuat mekanisme ini terbatas dalam melakukan pencarian solusi.

6.2. Saran

Berikut adalah beberapa saran yang dapat digunakan untuk penelitian lanjutan:

1. Perlu dikembangkan mekanisme pemilihan kendaraan dalam kasus *Heterogeneous Fleet* yang sesuai dengan mekanisme CEGA.

2. Menambahkan *dummy* dalam rute yang terbentuk dengan mencari mekanisme mutasi lain yang memiliki kemampuan pencarian solusi yang lebih baik.
3. Modifikasi Algoritma CEGA agar tidak mudah terjebak di *local optimum* dan mencari nilai parameter yang optimum untuk semua *instance*.
4. Penelitian dapat dikembangkan dengan kasus lain seperti 3 *dimensional loading constraint* VRP.

DAFTAR PUSTAKA

Ai, T. J. (2008). *Particle Swarm Optimization for Generalized Vehicle Routing Problem*. Asian Institute of Technology, Thailand.

Ai, T. J., & Kachitvichyanukul, V. (2009). Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem. *Computers & Industrial Engineering*, *56*(1), 380–387.

Aini, B. K. (2012). *Pengembangan Algoritma Hybrid Cross Entropy-Genetic Algorithm untuk Penyelesaian Generalized Orienteering Problem*. Skripsi. Institut Teknologi Sepuluh Nopember. Surabaya.

Alonso, M. T., Alvarez-valdes, R., Iori, M., Parreño, F., & Tamarit, J. M. (2016). Mathematical models for multicontainer loading problems. *Omega*, 1–12. http://doi.org/10.1016/j.omega.2016.02.002

Baldacci, R., Battarra, M., & Vigo, D. (2008). Routing a heterogeneous fleet of vehicles. In *The vehicle routing problem: latest advances and new challenges* (pp. 3–27). Springer.

Bortfeldt, A. (2013). Computers & Operations Research Packing first , routing second — a heuristic for the vehicle routing and loading problem, *40*, 873–885. http://doi.org/10.1016/j.cor.2012.09.005

Braekers, K., Ramaekers, K., & Nieuwenhuyse, I. Van. (2016). Computers & Industrial Engineering The vehicle routing problem : State of the art classification and review. *Computers & Industrial Engineering*, *99*, 300–313. http://doi.org/10.1016/j.cie.2015.12.007

Cabo, M., & Edgar, N. (2011). A Comparison of Cross Entropy Approaches for the Classical Vehicle Routing Problem, (1), 1–11.

Chazelle, B. (1983). The bottomn-left bin-packing heuristic: an efficient implementation. *Computers, IEEE Transactions on*, *100*(8), 697–707.

Duhamel, C., Lacomme, P., Quilliot, A., & Toussaint, H. (2011). A multi-start evolutionary local search for the two-dimensional loading capacitated vehicle

routing problem. *Computers & Operations Research*, *38*(3), 617–640. http://doi.org/http://dx.doi.org/10.1016/j.cor.2010.08.017

Fernández, A., Gil, C., Baños, R., & Montoya, M. G. (2013). A parallel multi-objective algorithm for two-dimensional bin packing with rotations and load balancing. *Expert Systems With Applications*, *40*(13), 5169–5180. http://doi.org/10.1016/j.eswa.2013.03.015

Fuellerer, G., Doerner, K. F., Hartl, R. F., & Iori, M. (2009). Ant colony optimization for the two-dimensional loading vehicle routing problem. *Computers and Operations Research*, *36*(3), 655–673. http://doi.org/10.1016/j.cor.2007.10.021

Fuellerer, G., Doerner, K. F., Hartl, R. F., & Iori, M. (2010). Metaheuristics for vehicle routing problems with three-dimensional loading constraints. *European Journal of Operational Research*, *201*(3), 751–759. http://doi.org/10.1016/j.ejor.2009.03.046

Hauman, C., & Bekker, J. (2014). Application of the multi-objective cross-entropy method to the vehicle routing problem with soft time windows. *Orion*, *30*(1), 19–40.

Haymo. (2016). Comparison: C# (ILNumerics), FORTRAN, MATLAB and numpy – Part II. Retrieved December 12, 2016, from http://ilnumerics.net/blog/fast-faster-performance-comparison-c-ilnumerics-fortran-matlab-and-numpy-part-ii/

Hokama, P., Miyazawa, F. K., & Xavier, E. C. (2016). A branch-and-cut approach for the vehicle routing problem with loading constraints, *47*, 1–13. http://doi.org/10.1016/j.eswa.2015.10.013

Hoover, S. V, & Perry, R. F. (1989). *Simulation: a problem-solving approach*. Addison-Wesley Longman Publishing Co., Inc.

Iori, M. (2005). Metaheuristic algorithms for combinatorial optimization problems. *4OR*, *3*(2), 163–166.

Iori, M., & Martello, S. (2010). Routing problems with loading constraints. *Top*,

18(1), 4–27.

Jabali, O., & Laporte, G. (2016). Thirty years of heterogeneous vehicle routing, *249*, 1–21. http://doi.org/10.1016/j.ejor.2015.07.020

Junqueira, L., & Morabito, R. (2015). Computers & Industrial Engineering Heuristic algorithms for a three-dimensional loading capacitated vehicle routing problem in a carrier q. *Computers & Industrial Engineering*, *88*, 110–130. http://doi.org/10.1016/j.cie.2015.06.005

Kumar, S., & Panneerselvam, R. (2012). A Survey on the Vehicle Routing Problem and Its Variants. *Intelligent Information Management*, 9. http://doi.org/10.4236/iim.2012.43010

Lahdji, F. M. (2016). *Implementasi Algoritma Hybrid Cross Entropy-Genetic Algorithm untuk Menyelesaikan Single Stage Capacitated Warehouse Location Problem (Studi Kasus: PT. Petrokimia Gresik)*. Skripsi. Institut Teknologi Sepuluh Nopember. Surabaya.

Leung, S. C. H., Zhang, Z., Zhang, D., Hua, X., & Lim, M. K. (2013). A meta-heuristic algorithm for heterogeneous fleet vehicle routing problems with two-dimensional loading constraints. *European Journal of Operational Research*, *225*(2), 199–210. http://doi.org/10.1016/j.ejor.2012.09.023

Li, X., & Zhang, K. (2015). A hybrid differential evolution algorithm for multiple container loading problem with heterogeneous containers. *Computers and Industrial Engineering*, *90*, 305–313. http://doi.org/10.1016/j.cie.2015.10.007

Lodi, A., Martello, S., & Vigo, D. (1999). Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *INFORMS Journal on Computing*, *11*(4), 345–357.

Lopez-garcia, P., Onieva, E., Osaba, E., Masegosa, A. D., & Perallos, A. (2016). Regular articles GACE : A meta-heuristic based in the hybridization of Genetic Algorithms and Cross Entropy methods for continuous optimization. *Expert Systems With Applications*, *55*, 508–519. http://doi.org/10.1016/j.eswa.2016.02.034

Mahvash, B., Awasthi, A., & Chauhan, S. (2015). ScienceDirect Based Heuristic Heuristic for the Capacitated Capacitated Vehicle Vehicle Routing Routing Based for the Problem with Three-dimensional Loading Constraints Generation Based Heuristic Capacitated Vehicle with Three-dimensional Loading Constrai. *IFAC-PapersOnLine*, *48*(3), 448–453. http://doi.org/10.1016/j.ifacol.2015.06.122

Männel, D., & Bortfeldt, A. (2016). A Hybrid Algorithm for the Vehicle Routing Problem with Pickup and Delivery and Three dimensional Loading Constraints. *European Journal of Operational Research Received*. http://doi.org/10.1016/j.ejor.2016.04.016

Miao, L., Ruan, Q., Woghiren, K., & Ruo, Q. (2012). A hybrid genetic algorithm for the vehicle routing problem with three-dimensional loading constraints. *RAIRO - Operations Research*, *46*(1), 63–82. http://doi.org/10.1051/ro/2012008

Ruan, Q., Zhang, Z., Miao, L., & Shen, H. (2013). Computers & Operations Research A hybrid approach for the vehicle routing problem with three-dimensional loading constraints. *Computers and Operation Research*, *40*(6), 1579–1589. http://doi.org/10.1016/j.cor.2011.11.013

Santosa, B., Budiman, M. A., & Wiratno, S. E. (2011). A Cross Entropy-Genetic Algorithm for m-Machines No-Wait Job-Shop Scheduling Problem, *2011*(August), 171–180. http://doi.org/10.4236/jilsa.2011.33018

Santosa, B., Damayanti, R., & Sarkar, B. (2016). Solving multi-product inventory ship routing with a heterogeneous fleet model using a hybrid cross entropy-genetic algorithm : a case study in Indonesia. *Production & Manufacturing Research*, *4*(1), 1–24. http://doi.org/10.1080/21693277.2016.1204961

Santosa, B., & Willy, P. (2011). *Metoda Metaheuristik konsep dan implementasi*. Surabaya: Guna Widya.

Toth, P., & Vigo, D. (2002a). *The vehicle routing problem*. (P. Toth & D. Vigo, Eds.), *Optimization* (Vol. 9). Philadelphia: Society for Industrial and Applied Mathematics. http://doi.org/10.1137/1.9780898718515

Toth, P., & Vigo, D. (2002b). *The Vehicle Routing Problem* (Vol. 9). SIAM.

Wang, C., & Qiu, Y. (2012). Vehicle Routing Problem with Stochastic Demands and Simultaneous Delivery and Pickup Chuansheng Wang, Yue Qiu. *Applied Mechanics and Materials*, *149*, 810–813. http://doi.org/10.4028/www.scientific.net/AMM.148-149.810

Wei, L., Zhang, Z., Zhang, D., & Lim, A. (2015). Discrete Optimization A variable neighborhood search for the capacitated vehicle routing problem with two-dimensional loading constraints. *European Journal of Operational Research*, *243*(3), 798–814. http://doi.org/10.1016/j.ejor.2014.12.048

Weise, T., Podlich, A., & Gorldt, C. (2010). Solving Real-World Vehicle Routing Problems with Evolutionary Algorithms. *Natural Intelligence for Scheduling, Planning and Packing Problems*, 1–27.

Yeun, L. C., Ismail, W. A. N. R., Omar, K., & Zirour, M. (2008). VEHICLE ROUTING PROBLEM : MODELS AND SOLUTIONS, *4*(1), 205–218.

Zachariadis, E. E., Tarantilis, C. D., & Kiranoudis, C. T. (2009). A Guided Tabu Search for the Vehicle Routing Problem with two-dimensional loading constraints. *European Journal of Operational Research*, *195*(3), 729–743. http://doi.org/10.1016/j.ejor.2007.05.058

(Halaman ini sengaja dikosongkan)

LAMPIRAN

A. Penulisan MATLAB Algoritma CEGA

```
function[Sech,SecTot,AllBG,BGjarut,BGrute]=CEGA2LHFVRP_V4(W,W2,N,s
quence,X,alpa,maxiter,e1,rho)
Time1=cputime;
xx=size(X,2)/5;
V1=[];
n=size(W,1)-1;
for nn=1:xx
    V1=[V1;X((nn-1)*5+1:(nn-1)*5+1+2)];
end

Fx=[];
for nn=1:xx
    Fx=[Fx;X((nn-1)*5+4)];
end

Vc=[];
for nn=1:xx
    Vc=[Vc;X((nn-1)*5+5)];
end


%INITIAL SOLUTION
[mjar]=hitungjarak(W);
[wb,~]=size(W);
Plite=ceil(rho*N); %jml sample elite
prob=1/(wb-1);
P=ones(wb,wb)*prob;
k=1;
%membuat matrik transisi PERTAMA
for i=1:wb
    P(i,i)=0;
end

AllBG=[];
BGjarut=999999999999999999;
durasi_periter=[];
allr=[];
[opt_rute,vh,VHuse,PLV,V,nnode,vhHF,used,cap]=solCE(P,wb,W,V1,W2,s
quence,N,xx,n);

%Menghitung jarak masing2 rute yang terbantuk
[opt_jarut]=HFjarakrute(mjar,opt_rute,vh,VHuse,Fx,Vc);

%Local improvement 2-Opt
[opt_rute,opt_jarut]=HFtwoOPT(vh,N,n,opt_rute,opt_jarut,mjar,W2,PL
V,squence,VHuse,Fx,Vc);

%Local improvement 1-1 Exchange
[opt_rute,opt_jarut,used]=HFIIexchange(opt_rute,opt_jarut,nnode,us
ed,N,vh,mjar,W,cap,W2,squence,PLV,VHuse,Fx,Vc);
```

```
%Local improvement 1-0 Exchange
[opt_rute,opt_jarut,used,nnode]=HFIOexchange(opt_rute,opt_jarut,nn
ode,used,N,vh,n,mjar,W,cap,W2,squence,PLV,VHuse,Fx,Vc);

durasi_init=cputime-Time1;
[mjar]=hitungjarak(W);
[wb,~]=size(W);


[~,EidxS]=sort(opt_jarut);
BGrute=opt_rute((EidxS(1)-1)*vh+1:EidxS(1)*vh,:);
BGjarut=opt_jarut(EidxS(1));
BGVHuse=VHuse(:,EidxS(1));
AllBG=BGjarut;

prob=1/(wb-1);
P_lama=ones(wb,wb)*prob;
k=1;
%membuat matrik transisi PERTAMA
for i=1:wb
    P_lama(i,i)=0;
end
durasi_periter=[];
allr=[];
P_new=1;
iter=1;
e=1;

Note=zeros(N,3);

%PERULANGAN ITERASI
while e>e1
    Note(:,1)=opt_jarut;
    Time2=cputime;
    vh=size(opt_rute,1)/N;
    [~,Eidx]=sort(opt_jarut);
    Slite=[];
    JarSlite=[];
    for i=1:Plite
        Slite=[Slite;opt_rute((Eidx(i)-1)*vh+1:Eidx(i)*vh,:)];
        JarSlite=[JarSlite;opt_jarut(Eidx(i))];
    end
    Slite=[zeros(size(Slite,1),1) Slite]; %sample elite

    nnodeL=zeros(size(Slite,1),1);
    %MENGHITUNG JUMLAH NODE PADA SAMPEL ELITE
    for i=1:size(Slite,1)
        for j=1:wb
            if Slite(i,j)~=0
                nnodeL(i)=nnodeL(i)+1;
            end
        end
    end

    g=(1/Plite);
```

```
    Mtran=zeros(wb,wb);
    %membuat matrik transisi dari rute
    for l=1:size(nnodeL,1)
        for j=1:nnodeL(l)

Mtran(Slite(l,j)+1,Slite(l,j+1)+1)=Mtran(Slite(l,j)+1,Slite(l,j+1)
+1)+g;
        end
    end

    %UPDATE PARAMETER
    P=alpa*Mtran+(1-alpa)*P_lama;
    rata2=zeros(wb,1);
    for i=1:wb
        rata2(i)=sum(P(i,:));
    end
    for i=1:wb
        for j=1:wb
            %NORMALISASI NILAI P BARU
            P(i,j)=P(i,j)/rata2(i);
        end
    end

    NoteR=zeros(N,1);
    NoteN=zeros(N,1);
    rN=zeros(N,1);
    N1=0;
    for i=1:N
        NoteR(i)=mean(Note(i,:));
        NoteN(i)=abs(NoteR(i)-Note(i,1));
        if NoteN(i)<0.0001
            rN(i)=(i);
            N1=N1+1;
        else
        end
    end
    r1=[];
    for i=1:N
        if rN(i)~=0
            r1=[r1;rN(i)];
        else
        end
    end

    if sum(rN)~=0

[opt_ruteN,vhN,VHuseN,PLVN,VN,nnodeN,vhHFN,usedN,capN]=solCE(P,wb,
W,V1,W2,squence,N1,xx,n);
        %Menghitung jarak masing2 rute yang terbantuk

[opt_jarut]=HFjarakrute(mjar,opt_ruteN,max(vhHFN),VHuseN,Fx,Vc);

        sel=abs(size(PLVN,1)-size(PLV,1));
        if size(PLV,1)<size(PLVN,1)
            newrute_sem=[];
            for yy=1:N
```

```
                newrute_sem=[newrute_sem;opt_rute((yy-
1)*vh+1:yy*vh,:)];

newrute_sem=[newrute_sem;zeros(sel,size(opt_rute,2))];
            end
            opt_rute=newrute_sem;
            V=[V;zeros(sel,size(V,2))];
            PLV=[PLV;zeros(sel,size(PLV,2))];
            for i=1:N
                PLV(end,(i-1)*2+1:i*2)=V1(1,2:3);
                V(end,i)=V1(1,1);
            end
            VHuse=[VHuse;ones(sel,size(VHuse,2))];
            for i=1:N1
                opt_rute((r1(i)-
1)*vhN+1:r1(i)*vhN,:)=opt_ruteN((i-1)*vhN+1:i*vhN,:);
                V(:,r1(i))=VN(:,i);
                vhHF(r1(i))=vhHFN(i);
                PLV(:,(r1(i)-1)*2+1:r1(i)*2)=PLVN(:,(i-
1)*2+1:i*2);
                VHuse(:,r1(i))=VHuseN(:,i);
            end


        elseif size(PLV,1)>size(PLVN,1)
            newrute_sem=[];
            for yy=1:N1
                newrute_sem=[newrute_sem;opt_ruteN((yy-
1)*vhN+1:yy*vhN,:)];

newrute_sem=[newrute_sem;zeros(sel,size(opt_rute,2))];
            end
            opt_ruteN=newrute_sem;

            VN=[VN;zeros(sel,N1)];
            PLVN=[PLVN;zeros(sel,N1*2)];
            for i=1:sel
                for j=1:N1
                    PLVN(vhN+i,(j-1)*2+1:j*2)=V1(1,2:3);
                end
            end
            VHuseN=[VHuseN;ones(sel,N1)];

            for i=1:N1
                opt_rute((r1(i)-1)*vh+1:r1(i)*vh,:)=opt_ruteN((i-
1)*vh+1:i*vh,:);
                V(:,r1(i))=VN(:,i);
                vhHF(r1(i))=vhHFN(i);
                PLV(:,(r1(i)-1)*2+1:r1(i)*2)=PLVN(:,(i-
1)*2+1:i*2);
                VHuse(:,r1(i))=VHuseN(:,i);
            end
        else
            for i=1:N1
                opt_rute((r1(i)-1)*vh+1:r1(i)*vh,:)=opt_ruteN((i-
1)*vh+1:i*vh,:);
                V(:,r1(i))=VN(:,i);
```

```
                vhHF(r1(i))=vhHFN(i);
                PLV(:,(r1(i)-1)*2+1:r1(i)*2)=PLVN(:,(i-
1)*2+1:i*2);
                VHuse(:,r1(i))=VHuseN(:,i);
            end
        end

        %Menghitung jarak masing2 rute yang terbantuk

[opt_jarut]=HFjarakrute(mjar,opt_rute,size(opt_rute,1)/N,VHuse,Fx,
Vc);
    end

    vh=size(opt_rute,1)/N;
    nnode=zeros(size(opt_rute,1),1);  %jumlah node
    for i=1:size(opt_rute,1)
        for j=1:n
            if opt_rute(i,j)~=0
                nnode(i)=nnode(i)+1;
            else
            end
        end
    end

    used=zeros(size(opt_rute,1),1);
    for ii=1:size(opt_rute,1)
        for ij=1:nnode(ii)
            used(ii)=used(ii)+W(opt_rute(ii,ij)+1,4);
        end
    end



    P_lama=P;

    cap=zeros(vh*N,1);
    k=1;
    for i=1:N
        for j=1:vh
            cap(k,1)=V1(VHuse(j,i),1);
            k=k+1;
        end
    end

    R=zeros(1,N);
    for i=1:N
        r=rand;
        opt_ruteN=opt_rute((i-1)*vh+1:i*vh,:);
        PLVN=PLV(:,(i-1)*2+1:i*2);
        VHuseN=VHuse(:,i);
        capN=cap((i-1)*vh+1:i*vh,:);
        nnodeN=nnode((i-1)*vh+1:i*vh,:);
        usedN=used((i-1)*vh+1:i*vh,:);
        if r<=1/3
            %Local improvement 2-Opt
```

```
[opt_ruteK,opt_jarutK]=HFtwoOPT(vh,1,n,opt_ruteN,opt_jarut(i),mjar
,W2,PLVN,squence,VHuseN,Fx,Vc);
            opt_rute((i-1)*vh+1:i*vh,:)=opt_ruteK;
            opt_jarut(i)=opt_jarutK;
            R(1,i)=1;
        elseif and(r>1/3,r<=2/3)
            %Local improvement 1-1 Exchange

[opt_ruteK,opt_jarutK,usedK]=HFIIexchange(opt_ruteN,opt_jarut(i),n
nodeN,usedN,1,vh,mjar,W,capN,W2,squence,PLVN,VHuseN,Fx,Vc);
            opt_rute((i-1)*vh+1:i*vh,:)=opt_ruteK;
            opt_jarut(i)=opt_jarutK;
            used((i-1)*vh+1:i*vh,:)=usedK;
            R(1,i)=2;
        else
            %Local improvement 1-0 Exchange
            [opt_ruteK,
opt_jarutK,usedK,nnodeK]=HFIOexchange(opt_ruteN,opt_jarut(i),nnode
N,usedN,1,vh,n,mjar,W,capN,W2,squence,PLVN,VHuseN,Fx,Vc);
            opt_rute((i-1)*vh+1:i*vh,:)=opt_ruteK;
            opt_jarut(i)=opt_jarutK;
            nnode((i-1)*vh+1:i*vh,:)=nnodeK;
            used((i-1)*vh+1:i*vh,:)=usedK;
            R(1,i)=3;
        end
    end

    %PENGGANTIAN JENIS VEHICLE
    kk=1;
    kl=1;
    for i=1:size(used,1)
        VHuse_sem=VHuse;PLV_sem=PLV;V_sem=V;
        for j=1:xx
            if used(i)<=V1(j,1)
                VHuse_sem(kl,kk)=j;
                PLV_sem(kl,(kk-1)*2+1:kk*2)=V1(j,2:3);
                V_sem(kl,kk)=V1(j,1);

[~,noserv,~]=TWOLHFVPReval(W2,squence,opt_rute(i,:),V1(j,2:3));
                if noserv==0
                    VHuse=VHuse_sem;
                    PLV=PLV_sem;
                    V_sem=V;
                else
                end
                break
            else
            end
        end
        if mod(i,vh)==0
            kk=kk+1;
            kl=1;
        else
            kl=kl+1;
        end
    end
```

```
    %Menghitung jarak masing2 rute yang terbantuk

[opt_jarut]=HFjarakrute(mjar,opt_rute,size(opt_rute,1)/N,VHuse,Fx,
Vc);

    for i=1:(3-1)
        Note(:,3-(i-1))=Note(:,3-i);
    end

    [~,Bidx]=sort(opt_jarut);
    BIrute=opt_rute((Bidx(1)-1)*vh+1:Bidx(1)*vh,:);
    BIjarut=opt_jarut(Bidx(1));
    BIVHuse=VHuse(:,Bidx(1));
    if BIjarut<BGjarut
        BGjarut=BIjarut;
        BGrute=BIrute;
        BGVHuse=BIVHuse;
        Sech=cputime-Time1;
    else
    end
    AllBG=[AllBG;BIjarut];
    durasi_t=cputime-Time2;
    durasi_periter=[durasi_periter;durasi_t];
    allr=[allr;R];
    u=mean(JarSlite)/(2*BGjarut);
    P_old=P_new;
    P_new=((1-alpa)*u+P_old*alpa)/2;
    e=abs(P_old-P_new);
    iter=iter+1;
    if iter>=maxiter
        break
    else
    end
end

nnode=zeros(size(BGrute,1),1);  %jumlah node
for i=1:size(BGrute,1)
    for j=1:n
        if BGrute(i,j)~=0
            nnode(i)=nnode(i)+1;
        else
        end
    end
end

used=zeros(size(BGrute,1),1);
for ii=1:size(BGrute,1)
    for ij=1:nnode(ii)
        used(ii)=used(ii)+W(BGrute(ii,ij)+1,4);
    end
end


cap=zeros(size(BGrute,1),1);
k=1;
```

```
for i=1:1
    for j=1:size(BGrute,1)
        cap(k,1)=V1(BGVHuse(j,i),1);
        k=k+1;
    end
end

[BGVHuse cap used BGrute]
sum(nnode)
BGjarut
Sech
SecTot=cputime-Time1
iter=1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
ccc=1;
cck=1;
for i=1:size(BGrute,1)*1

[~,noserv1,heur]=TWOLHFVPReval(W2,1,BGrute(i,:),V1(BGVHuse(i),2:3)
);
    noserv(i,1)=noserv1;
    if mod(i,size(BGrute,1))==0
        ccc=1;
        cck=cck+1;
    else
        ccc=ccc+1;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
Noserv
```

B. Penulisan MATLAB Algoritma Loading Constraint

```
function [pos,noserv,uter]=TWOLHFVPR(W2,N,vh,PLV,squence,rute)
[wp,wl]=size(W2);
n=wp-1;
pos=[zeros(wp*N,wl) zeros(wp*N,wl-2)];
W2S=zeros(wp*N,wl);

W2plus=zeros(size(W2,1),1);
for i=1:size(W2,1)
    for j=1:W2(i,2)
        W2plus(i)=W2plus(i)+W2(i,j*2+1)*W2(i,j*2+2);
    end
end

for i=1:N
    pos(i*wp-wp+1:i*wp,1:wl)=[W2(:,1:2) zeros(wp,wl-2)];
    W2S(i*wp-wp+1:i*wp,:)=W2;
end

nnode4=zeros(vh*N,1); %jumlah node dalam @rute
```

```
for i=1:vh*N
    for j=1:n
        if rute(i,j)~=0
            nnode4(i,1)=nnode4(i,1)+1;
        else
            nnode4(i,1)=nnode4(i,1)+0;
        end
    end
end

if squence==1
    uter=zeros(vh*N,max(nnode4));
    for i=1:N*vh
        for j=1:nnode4(i)
            uter(i,j)=rute(i,(nnode4(i)+1)-j);
        end
    end
elseif squence==2
    uter=zeros(vh*N,max(nnode4));
    ruteW=rute;
    for i=1:N*vh
        for j=1:nnode4(i)
            ruteW(i,j)=W2plus(rute(i,j)+1);
        end
    end
    [~,Widx]=sort(ruteW,2,'descend');
    for i=1:N*vh
        for j=1:nnode4(i)
            uter(i,j)=rute(i,Widx(i,j));
        end
    end
end


h=1;
g=1;
noserv=zeros(N*vh,1);
xx=1;
while h<=vh*N-vh+1
    pos_sem=[W2(:,1:2) zeros(wp,wl-2) zeros(wp,wl-2)];
    W2_sem=W2;
    cc=1;
    for i=h:h+(vh-1)
        for heur=1:5
            pos_sem1=pos_sem;
            W2_sem1=W2_sem;
            poslist=zeros(1,2);
            frbd=zeros(1,4);
            for j=1:nnode4(i)
                L=zeros(W2(uter(i,j)+1,2),1);
                for k=1:W2(uter(i,j)+1,2)

L(k,1)=W2(uter(i,j)+1,(k*2+1))*W2(uter(i,j)+1,(k*2+2));
                end
                [~,idx]=sort(L,'descend');
                for k=1:W2(uter(i,j)+1,2)
                    [~,kpl]=size(poslist);
```

```
                    nposlist=kpl/2;
                    kor=zeros(1,2);
                    ok=0;
                    postrans=zeros(nposlist,2);
                    if nposlist==1
                        if k==1
                            poslist_sem=poslist;
                            frbd_sem=frbd;
                            postrans_sem=postrans;
                            pos_sem2=pos_sem;
                            W2_sem2=W2_sem;
                        else
                        end


kor=W2(uter(i,j)+1,(idx(k,1)*2+1):(idx(k,1)*2+1)+1);
                        kor_sem=kor;
                        kor_sem(1,1)=kor(1,2);
                        kor_sem(1,2)=kor(1,1);
                        if sum(poslist+kor<PLV(cc,xx*2-1:xx*2))>1
                            %Menempatkan paket dari konsumen pada
suatu koordinat

pos_sem(uter(i,j)+1,(idx(k,1)*2+1):(idx(k,1)*2+1)+1)=[poslist];

pos_sem(uter(i,j)+1,2)=pos_sem(uter(i,j)+1,2)-1;
                            %Menghilangan paket dalam daftar paket

W2_sem(uter(i,j)+1,(idx(k,1)*2+1):(idx(k,1)*2+1)+1)=[0 0];
                            newposlist=[poslist zeros(1,2)];
                            newposlist(1,1:2)=[kor(1,1)
poslist(1,1)];
                            newposlist(1,3:4)=[poslist(1,2)
kor(1,2)];
                            frbd=[frbd; poslist poslist+kor];

pos_sem(uter(i,j)+1,(idx(k,1)*2+5):(idx(k,1)*2+5)+1)=poslist+kor;

                        elseif sum(poslist+kor_sem<PLV(cc,xx*2-
1:xx*2))>1
                            %Menempatkan paket dari konsumen pada
suatu koordinat

pos_sem(uter(i,j)+1,(idx(k,1)*2+1):(idx(k,1)*2+1)+1)=poslist;

pos_sem(uter(i,j)+1,2)=pos_sem(uter(i,j)+1,2)-1;
                            %Menghilangan paket dalam daftar paket

W2_sem(uter(i,j)+1,(idx(k,1)*2+1):(idx(k,1)*2+1)+1)=[0 0];
                            newposlist=[poslist zeros(1,2)];
                            newposlist(1,1:2)=[kor_sem(1,1)
poslist(1,1)];
                            newposlist(1,3:4)=[poslist(1,2)
kor_sem(1,2)];
                            frbd=[frbd; poslist poslist+kor_sem];
```

```
pos_sem(uter(i,j)+1,(idx(k,1)*2+5):(idx(k,1)*2+5)+1)=poslist+kor_s
em;
                        else
                            break
                        end
                    else
                        %bukan iterasi pertama

kor=W2(uter(i,j)+1,(idx(k,1)*2+1):(idx(k,1)*2+1)+1);
                        kor_sem=kor;
                        kor_sem(1,1)=kor(1,2);
                        kor_sem(1,2)=kor(1,1);
                        postrans=zeros(nposlist,2);
                        for a=1:nposlist
                            postrans(a,:)=poslist(1,2*a-1:2*a);
                        end
                        [nfrbd,~]=size(frbd);
                        kor_g=[kor;kor_sem];


                        b=1; %MENGHILANGKAN POSLIST YANG BERHIMPIT
BARANG (POSLIST)
                        while b<=nposlist
                            for c=2:nfrbd
                                if poslist(1,b*2)==frbd(c,2)
                                    if and(poslist(1,b*2-
1)>frbd(c,1),poslist(1,b*2-1)<frbd(c,3))
                                        poslist(b*2-1:b*2)=[];
                                        [~,klp]=size(poslist);
                                        nposlist=klp/2;
                                        break
                                    end
                                end
                            end
                            b=b+1;
                        end

                        b=1; %MENGHILANGKAN POSLIST YANG BERHIMPIT
BARANG (POSLIST)
                        while b<=nposlist
                            for c=2:nfrbd
                                if poslist(1,b*2-1)==frbd(c,1)
                                    if
and(poslist(1,b*2)>frbd(c,2),poslist(1,b*2)<frbd(c,4))
                                        poslist(b*2-1:b*2)=[];
                                        [~,klp]=size(poslist);
                                        nposlist=klp/2;
                                        break
                                    end
                                end
                            end
                            b=b+1;
                        end

                        %MENGHINDARI POSLIST YANG MENGGANTUNG
(POSLIST)
```

```
                        [~,idk]=sort(frbd(:,3),'descend');
                        for b=1:nposlist
                            for c=1:nfrbd
                                if
and(poslist(1,b*2)>=frbd(idk(c),2),poslist(1,b*2)<frbd(idk(c),4))
                                    if poslist(1,b*2-
1)>=frbd(idk(c),3)
                                        poslist(1,b*2-
1)=frbd(idk(c),3);
                                        break
                                    else
                                    end
                                else
                                end
                            end
                        end

                        %MENGHINDARI POSLIST YANG MENGGANTUNG
(POSLIST)
                        [~,mk]=sort(frbd(:,4),'descend');
                        for b=1:nposlist
                            if poslist(1,2*b)>=frbd(mk(1),4)
                                poslist(1,2*b-1)=0;
                            end
                        end

                        %MENGHINDARI POSLIST TERHALANG (POSLIST)
                        b=1;
                        while b<=nposlist
                            for c=1:nfrbd
                                if poslist(1,2*b-1)<frbd(c,1)
                                    if
and(poslist(1,2*b)<frbd(c,4),poslist(1,2*b)>=frbd(c,2))
                                        poslist(1,2*b)=frbd(c,4);
                                        b=b-1;
                                        break
                                    else
                                    end
                                else
                                end
                            end
                            b=b+1;
                        end

                        b=1;  %MENGHINDARI POSLIST GANDA (POSLIST)
                        while b<=nposlist-1
                            d=b+1;
                            while d<=nposlist
                                if poslist(1,b*2-
1:b*2)==poslist(1,d*2-1:d*2)
                                    poslist(d*2-1:d*2)=[];
                                    [~,klp]=size(poslist);
                                    nposlist=klp/2;
                                end
                                d=d+1;
                            end
                            b=b+1;
```

```
                        end

                        %MENGUBAH POSLIIST KE POSTRANS
                        postrans=zeros(nposlist,2);
                        for a=1:nposlist
                            postrans(a,:)=poslist(1,2*a-1:2*a);
                        end



                        if heur==1

[priopos,poslist,postrans,nposlist]=PosHeur(postrans,nposlist,frbd
,nfrbd);
                        elseif heur==2

[priopos,postrans,nposlist,kor_g]=PosHeur2(postrans,nposlist,kor_g
);
                        elseif heur==3

[priopos,postrans,nposlist,kor_g,prio1,TouchP]=PosHeur3(poslist,po
strans,nposlist,frbd,kor_g,PLV(cc,xx*2-1:xx*2),nfrbd);
                        elseif heur==4

[priopos,postrans,nposlist,kor_g,prio1,TouchP]=PosHeur4(poslist,po
strans,nposlist,frbd,kor_g,PLV(cc,xx*2-1:xx*2),nfrbd);
                        elseif heur==5

[priopos,nposlist,prio1,AreaP]=PosHeur5(poslist,nposlist,frbd,kor_
g,PLV(cc,xx*2-1:xx*2),nfrbd);
                        end



                        if k==1
                            poslist_sem=poslist;
                            frbd_sem=frbd;
                            postrans_sem=postrans;
                            pos_sem2=pos_sem;
                            W2_sem2=W2_sem;
                        else
                        end

                        %perulangan perhitungan pemilihan poslist
                        capos=[9999 9999;9999 9999];



                        for b=1:nposlist
                            %perulangan perhitungan untuk dua
orientasi
                            %(rotasi)
                            a=1;
                            while a<=2
                                if or(or(heur==3,heur==4),heur==5)
                                    if a==1
                                        priopos=prio1(1,:);
                                    elseif a==2
```

```
                                    priopos=prio1(2,:);
                                end
                            else
                            end

                            if sum(poslist(1,priopos(b)*2-
1:priopos(b)*2)+kor_g(a,:)<=PLV(cc,xx*2-1:xx*2))>1

capos(a,:)=poslist(1,priopos(b)*2-1:priopos(b)*2);
                                c=1;
                                while c<=nfrbd
                                    if
or(and(and(capos(a,1)<=frbd(c,1),capos(a,1)<=frbd(c,3)),and(capos(
a,1)+kor_g(a,1)<=frbd(c,1),capos(a,1)+kor_g(a,1)<=frbd(c,3))),and(
and(capos(a,1)>=frbd(c,1),capos(a,1)>=frbd(c,3)),and(capos(a,1)+ko
r_g(a,1)>=frbd(c,1),capos(a,1)+kor_g(a,1)>=frbd(c,3))))
                                        if
or(and(and(capos(a,2)<=frbd(c,2),capos(a,2)<=frbd(c,4)),and(capos(
a,2)+kor_g(a,2)<=frbd(c,2),capos(a,2)+kor_g(a,2)<=frbd(c,4))),and(
and(capos(a,2)>=frbd(c,2),capos(a,2)>=frbd(c,4)),and(capos(a,2)+ko
r_g(a,2)>=frbd(c,2),capos(a,2)+kor_g(a,2)>=frbd(c,4))))

                                        else

                                        end
                                    elseif
or(and(and(capos(a,2)<=frbd(c,2),capos(a,2)<=frbd(c,4)),and(capos(
a,2)+kor_g(a,2)<=frbd(c,2),capos(a,2)+kor_g(a,2)<=frbd(c,4))),and(
and(capos(a,2)>=frbd(c,2),capos(a,2)>=frbd(c,4)),and(capos(a,2)+ko
r_g(a,2)>=frbd(c,2),capos(a,2)+kor_g(a,2)>=frbd(c,4))))

                                    else
                                        if
frbd(c,3)+kor_g(a,1)<=PLV(cc,xx*2-1)

capos(a,:)=poslist(1,priopos(b)*2-1:priopos(b)*2);

capos(a,1)=frbd(c,3);
                                            c=1;
                                        else
                                            capos(a,:)=[9999
9999];
                                        end
                                    end
                                    c=c+1;
                                end
                            else
                            end
                            a=a+1;
                        end

                        [~,mx]=sort(frbd(:,4),'descend');
                        for a=1:2
                            if
and(capos(a,2)+kor_g(a,2)>frbd(mx(1),4)+ceil(0.1*PLV(cc,xx*2)),cap
os(a,2)<frbd(mx(1),4))
                                capos(a,:)=[9999 9999];
```

```
                                    break
                                end
                            end

                            if or(heur==1,heur==2)
                                if sum(sum(capos))==39996
                                    ok=ok+1;
                                elseif
sum(capos(1,:))<=sum(capos(2,:))
                                    %Menempatkan paket dari
konsumen pada suatu koordinat

pos_sem(uter(i,j)+1,(idx(k,1)*2+1):(idx(k,1)*2+1)+1)=capos(1,:);

pos_sem(uter(i,j)+1,2)=pos_sem(uter(i,j)+1,2)-1;
                                    %Menghilangan paket dalam
daftar paket

W2_sem(uter(i,j)+1,(idx(k,1)*2+1):(idx(k,1)*2+1)+1)=[0 0];
                                    newposlist=[poslist
zeros(1,2)];
                                    [~,knpl]=size(newposlist);
                                    plpilih=capos(1,:);
                                    newposlist(1,priopos(b)*2-
1:priopos(b)*2)=[plpilih(1,1) kor_g(1,2)+plpilih(1,2)];
                                    newposlist(1,knpl-
1:knpl)=[kor_g(1,1)+plpilih(1,1) plpilih(1,2)];
                                    frbd=[frbd;capos(1,:)
capos(1,:)+kor_g(1,:)];

pos_sem(uter(i,j)+1,(idx(k,1)*2+5):(idx(k,1)*2+5)+1)=capos(1,:)+ko
r_g(1,:);
                                    break
                                elseif
sum(capos(2,:))<sum(capos(1,:))
                                    %Menempatkan paket dari
konsumen pada suatu koordinat

pos_sem(uter(i,j)+1,(idx(k,1)*2+1):(idx(k,1)*2+1)+1)=capos(2,:);

pos_sem(uter(i,j)+1,2)=pos_sem(uter(i,j)+1,2)-1;
                                    %Menghilangan paket dalam
daftar paket

W2_sem(uter(i,j)+1,(idx(k,1)*2+1):(idx(k,1)*2+1)+1)=[0 0];
                                    newposlist=[poslist
zeros(1,2)];
                                    [~,knpl]=size(newposlist);
                                    plpilih=capos(2,:);
                                    newposlist(1,priopos(b)*2-
1:priopos(b)*2)=[plpilih(1,1) kor_g(2,2)+plpilih(1,2)];
                                    newposlist(1,knpl-
1:knpl)=[kor_g(2,1)+plpilih(1,1) plpilih(1,2)];
                                    frbd=[frbd;capos(2,:)
capos(2,:)+kor_g(2,:)];
```

```
pos_sem(uter(i,j)+1,(idx(k,1)*2+5):(idx(k,1)*2+5)+1)=capos(2,:)+ko
r_g(2,:);
                                    break
                                else
                                end
                            end

                            if or(heur==3,heur==4)
                                if sum(sum(capos))==39996
                                    ok=ok+1;
                                elseif
and(TouchP(1,prio1(1,b))>=TouchP(2,prio1(2,b)),sum(capos(1,:))~=19
998)
                                    %Menempatkan paket dari
konsumen pada suatu koordinat

pos_sem(uter(i,j)+1,(idx(k,1)*2+1):(idx(k,1)*2+1)+1)=capos(1,:);

pos_sem(uter(i,j)+1,2)=pos_sem(uter(i,j)+1,2)-1;
                                    %Menghilangan paket dalam
daftar paket

W2_sem(uter(i,j)+1,(idx(k,1)*2+1):(idx(k,1)*2+1)+1)=[0 0];
                                    newposlist=[poslist
zeros(1,2)];
                                    [~,knpl]=size(newposlist);
                                    plpilih=capos(1,:);
                                    newposlist(1,prio1(1,b)*2-
1:prio1(1,b)*2)=[plpilih(1,1) kor_g(1,2)+plpilih(1,2)];
                                    newposlist(1,knpl-
1:knpl)=[kor_g(1,1)+plpilih(1,1) plpilih(1,2)];
                                    frbd=[frbd;capos(1,:)
capos(1,:)+kor_g(1,:)];

pos_sem(uter(i,j)+1,(idx(k,1)*2+5):(idx(k,1)*2+5)+1)=capos(1,:)+ko
r_g(1,:);
                                    break
                                elseif
and(TouchP(2,prio1(2,b))>TouchP(1,b),sum(capos(2,:))~=19998)
                                    %Menempatkan paket dari
konsumen pada suatu koordinat

pos_sem(uter(i,j)+1,(idx(k,1)*2+1):(idx(k,1)*2+1)+1)=capos(2,:);

pos_sem(uter(i,j)+1,2)=pos_sem(uter(i,j)+1,2)-1;
                                    %Menghilangan paket dalam
daftar paket

W2_sem(uter(i,j)+1,(idx(k,1)*2+1):(idx(k,1)*2+1)+1)=[0 0];
                                    newposlist=[poslist
zeros(1,2)];
                                    [~,knpl]=size(newposlist);
                                    plpilih=capos(2,:);
                                    newposlist(1,prio1(2,b)*2-
1:prio1(2,b)*2)=[plpilih(1,1) kor_g(2,2)+plpilih(1,2)];
```

```
                                    newposlist(1,knpl-
1:knpl)=[kor_g(2,1)+plpilih(1,1) plpilih(1,2)];
                                    frbd=[frbd;capos(2,:)
capos(2,:)+kor_g(2,:)];

pos_sem(uter(i,j)+1,(idx(k,1)*2+5):(idx(k,1)*2+5)+1)=capos(2,:)+ko
r_g(2,:);
                                    break
                                end
                            end

                            if heur==5
                                if sum(sum(capos))==39996
                                    ok=ok+1;
                                elseif
sum(capos(1,:))<=sum(capos(2,:))
                                    %Menempatkan paket dari
konsumen pada suatu koordinat

pos_sem(uter(i,j)+1,(idx(k,1)*2+1):(idx(k,1)*2+1)+1)=capos(1,:);

pos_sem(uter(i,j)+1,2)=pos_sem(uter(i,j)+1,2)-1;
                                    %Menghilangan paket dalam
daftar paket

W2_sem(uter(i,j)+1,(idx(k,1)*2+1):(idx(k,1)*2+1)+1)=[0 0];
                                    newposlist=[poslist
zeros(1,2)];
                                    [~,knpl]=size(newposlist);
                                    plpilih=capos(1,:);
                                    newposlist(1,prio1(1,b)*2-
1:prio1(1,b)*2)=[plpilih(1,1) kor_g(1,2)+plpilih(1,2)];
                                    newposlist(1,knpl-
1:knpl)=[kor_g(1,1)+plpilih(1,1) plpilih(1,2)];
                                    frbd=[frbd;capos(1,:)
capos(1,:)+kor_g(1,:)];

pos_sem(uter(i,j)+1,(idx(k,1)*2+5):(idx(k,1)*2+5)+1)=capos(1,:)+ko
r_g(1,:);
                                    break
                                elseif
sum(capos(2,:))<sum(capos(1,:))
                                    %Menempatkan paket dari
konsumen pada suatu koordinat

pos_sem(uter(i,j)+1,(idx(k,1)*2+1):(idx(k,1)*2+1)+1)=capos(2,:);

pos_sem(uter(i,j)+1,2)=pos_sem(uter(i,j)+1,2)-1;
                                    %Menghilangan paket dalam
daftar paket

W2_sem(uter(i,j)+1,(idx(k,1)*2+1):(idx(k,1)*2+1)+1)=[0 0];
                                    newposlist=[poslist
zeros(1,2)];
                                    [~,knpl]=size(newposlist);
                                    plpilih=capos(2,:);
```

```
                                    newposlist(1,prio1(2,b)*2-
1:prio1(2,b)*2)=[plpilih(1,1) kor_g(2,2)+plpilih(1,2)];
                                    newposlist(1,knpl-
1:knpl)=[kor_g(2,1)+plpilih(1,1) plpilih(1,2)];
                                    frbd=[frbd;capos(2,:)
capos(2,:)+kor_g(2,:)];

pos_sem(uter(i,j)+1,(idx(k,1)*2+5):(idx(k,1)*2+5)+1)=capos(2,:)+ko
r_g(2,:);
                                    break
                                end
                            end

                        end

                    end
                    %end poslist==1

                    if ok==nposlist
                        poslist=poslist_sem;
                        frbd=frbd_sem;
                        postrans=postrans_sem;
                        pos_sem=pos_sem2;
                        W2_sem=W2_sem2;
                        break
                    else
                        poslist=newposlist;
                    end


                end
                %end k
            end
            %end j

            noserv1=0;
            for j=1:nnode4(i)
                if pos_sem(uter(i,j)+1,2)~=0
                    noserv1=noserv1+1;
                else
                end
            end
            if noserv1==0
                break
            else
                pos_sem=pos_sem1;
                W2_sem=W2_sem1;
            end
        end
        %end heur=1:5
        noserv(i,1)=noserv1;
        cc=cc+1;
    end
    %end i
    h=h+vh;
    xx=xx+1;
```

```
    poslist=[0 0];
    pos(g*wp-wp+1:g*wp,:)=pos_sem;
    W2S(g*wp-wp+1:g*wp,:)=W2_sem;
    g=g+1;
    %[nfrbd_all,~]=size(frbd_all);
    %frbd_all(nfrbd_all+1,:)=zeros(1,4);
    %frbd_all(nfrbd_all+2:nfrbd_all+1+nfrbd,:)=frbd;

end
%end h
```

C. Penulisan MATLAB Packing Heuristic

a. Heuristic 1

```
function
[priopos,poslist,postrans,nposlist]=PosHeur(postrans,nposlist,frbd
,nfrbd)

%HEURISTIC SATU
i=1;
while i<=nposlist
    for j=1:nfrbd
        if postrans(i,1)<frbd(j,1)
            if
and(postrans(i,2)<frbd(j,4),postrans(i,2)>=frbd(j,2))
                postrans(i,2)=frbd(j,4);
                [nposlist,~]=size(postrans);
                i=i-1;
                break
            else
            end
        else
        end
    end
    i=i+1;
end
poslist=zeros(1,2*nposlist);
for i=1:nposlist
    poslist(1,i*2-1:i*2)=postrans(i,:);
end

[~,priopos]=sort(postrans(:,1));
prio1=0;
```

b. Heuristic 2

```
function
[priopos,postrans,nposlist,kor_g]=PosHeur2(postrans,nposlist,kor_g
)

%HEURISTIC DUA

%MENGMBERIKAN BOBOT POSLIST %DENGAN NILAI W YANG SAMA
postrans1=postrans;
for i=1:nposlist-1
```

```
    for j=i+1:nposlist
        if postrans1(i,2)==postrans1(j,2)
            if postrans1(i,1)<postrans1(j,1)
                postrans1(j,2)=postrans1(j,2)+1;
            elseif postrans1(i,1)>postrans1(j,1)
                postrans1(i,2)=postrans1(i,2)+1;
            end
        end
    end
end

if kor_g(1,1)>kor_g(1,2)
    kor_g(2,:)=kor_g(1,:);
elseif kor_g(1,1)<kor_g(1,2)
    kor_g(1,:)=kor_g(2,:);
else
end

[~,priopos]=sort(postrans1(:,2));
```

c. Heuristic 3

```
function
[priopos,postrans,nposlist,kor_g,prio1,TouchP]=PosHeur3(poslist,po
strans,nposlist,frbd,kor_g,PLV,nfrbd)

%HEURISTIC TIGA
tc=zeros(nposlist,1);
poslist2=[poslist;poslist];
capos=[9999 9999;9999 9999];
for b=1:nposlist
    %perulangan perhitungan untuk dua orientasi (rotasi)
    for a=1:2
        if sum(poslist2(a,b*2-1:b*2)+kor_g(a,:)<=PLV)>1
            capos(a,:)=poslist2(a,b*2-1:b*2);
            c=1;
            while c<=nfrbd
                if
or(and(and(capos(a,1)<=frbd(c,1),capos(a,1)<=frbd(c,3)),and(capos(
a,1)+kor_g(a,1)<=frbd(c,1),capos(a,1)+kor_g(a,1)<=frbd(c,3))),and(
and(capos(a,1)>=frbd(c,1),capos(a,1)>=frbd(c,3)),and(capos(a,1)+ko
r_g(a,1)>=frbd(c,1),capos(a,1)+kor_g(a,1)>=frbd(c,3))))
                    if
or(and(and(capos(a,2)<=frbd(c,2),capos(a,2)<=frbd(c,4)),and(capos(
a,2)+kor_g(a,2)<=frbd(c,2),capos(a,2)+kor_g(a,2)<=frbd(c,4))),and(
and(capos(a,2)>=frbd(c,2),capos(a,2)>=frbd(c,4)),and(capos(a,2)+ko
r_g(a,2)>=frbd(c,2),capos(a,2)+kor_g(a,2)>=frbd(c,4))))

                    else

                    end
                elseif
or(and(and(capos(a,2)<=frbd(c,2),capos(a,2)<=frbd(c,4)),and(capos(
a,2)+kor_g(a,2)<=frbd(c,2),capos(a,2)+kor_g(a,2)<=frbd(c,4))),and(
and(capos(a,2)>=frbd(c,2),capos(a,2)>=frbd(c,4)),and(capos(a,2)+ko
r_g(a,2)>=frbd(c,2),capos(a,2)+kor_g(a,2)>=frbd(c,4))))
```

```
                else
                    if frbd(c,3)+kor_g(a,1)<=PLV(1,1)
                        capos(a,:)=poslist2(a,b*2-1:b*2);
                        capos(a,1)=frbd(c,3);
                        poslist2(a,b*2-1:b*2)=capos(a,:);
                        c=1;
                    else
                        capos(a,:)=[9999 9999];
                        poslist2(a,b*2-1:b*2)=PLV;
                    end
                end
                c=c+1;
            end
        else
            poslist2(a,b*2-1:b*2)=PLV;
        end
    end
end


TouchP=zeros(2,nposlist);
for b=1:nposlist
    %perulangan perhitungan untuk dua orientasi (rotasi)
    for a=1:2
        if sum(poslist2(a,b*2-1:b*2)+kor_g(a,:)<=PLV)>1
            capos(a,:)=poslist2(a,b*2-1:b*2);
            c=1;
            while c<=nfrbd
                if
or(and(and(capos(a,1)<=frbd(c,1),capos(a,1)<=frbd(c,3)),and(capos(
a,1)+kor_g(a,1)<=frbd(c,1),capos(a,1)+kor_g(a,1)<=frbd(c,3))),and(
and(capos(a,1)>=frbd(c,1),capos(a,1)>=frbd(c,3)),and(capos(a,1)+ko
r_g(a,1)>=frbd(c,1),capos(a,1)+kor_g(a,1)>=frbd(c,3))))
                    if
or(and(and(capos(a,2)<=frbd(c,2),capos(a,2)<=frbd(c,4)),and(capos(
a,2)+kor_g(a,2)<=frbd(c,2),capos(a,2)+kor_g(a,2)<=frbd(c,4))),and(
and(capos(a,2)>=frbd(c,2),capos(a,2)>=frbd(c,4)),and(capos(a,2)+ko
r_g(a,2)>=frbd(c,2),capos(a,2)+kor_g(a,2)>=frbd(c,4))))
                        %TIDAK ADA YG BERIRISAN
                    else
                        %BERIRISAN DI SUMBU W SAJA
                        if poslist2(a,b*2-1)==frbd(c,3)
                            aw=0;ak=0;
                            if poslist2(a,b*2)<=frbd(c,2)
                                aw=frbd(c,2);
                            else
                                aw=poslist2(a,b*2);
                            end

                            if
poslist2(a,b*2)+kor_g(a,2)<=frbd(c,4)
                                ak=poslist2(a,b*2)+kor_g(a,2);
                            else
                                ak=frbd(c,4);
                            end
                            TouchP(a,b)=TouchP(a,b)+(ak-aw);
```

```
                    end
                end
            elseif
or(and(and(capos(a,2)<=frbd(c,2),capos(a,2)<=frbd(c,4)),and(capos(
a,2)+kor_g(a,2)<=frbd(c,2),capos(a,2)+kor_g(a,2)<=frbd(c,4))),and(
and(capos(a,2)>=frbd(c,2),capos(a,2)>=frbd(c,4)),and(capos(a,2)+ko
r_g(a,2)>=frbd(c,2),capos(a,2)+kor_g(a,2)>=frbd(c,4))))
                %BERIRISAN DI SUMBU H SAJA
                if
or(poslist2(a,b*2)==frbd(c,4),poslist2(a,b*2)+kor_g(a,2)==frbd(c,2
))
                    aw=0;ak=0;
                    if poslist2(a,b*2-1)<=frbd(c,1)
                        aw=frbd(c,1);
                    else
                        aw=poslist2(a,b*2-1);
                    end

                    if poslist2(a,b*2-1)+kor_g(a,1)<=frbd(c,3)
                        ak=poslist2(a,b*2-1)+kor_g(a,1);
                    else
                        ak=frbd(c,3);
                    end
                    TouchP(a,b)=TouchP(a,b)+(ak-aw);
                end
            else
                %BERIRISAN DENGAN SUMBU H DAN W

            end
            c=c+1;
        end
    else

    end
end
end

[~,prio1]=sort(TouchP,2,'descend');
priopos=[];
```

d. Heuristic 4

```
function
[priopos,postrans,nposlist,kor_g,prio1,TouchP]=PosHeur4(poslist,po
strans,nposlist,frbd,kor_g,PLV,nfrbd)


%HEURISTIC EMPAT
tc=zeros(nposlist,1);
poslist2=[poslist;poslist];
capos=[9999 9999;9999 9999];
for b=1:nposlist
    %perulangan perhitungan untuk dua orientasi (rotasi)
    for a=1:2
        if sum(poslist2(a,b*2-1:b*2)+kor_g(a,:)<=PLV)>1
            capos(a,:)=poslist2(a,b*2-1:b*2);
```

```
            c=1;
            while c<=nfrbd
                if
or(and(and(capos(a,1)<=frbd(c,1),capos(a,1)<=frbd(c,3)),and(capos(
a,1)+kor_g(a,1)<=frbd(c,1),capos(a,1)+kor_g(a,1)<=frbd(c,3))),and(
and(capos(a,1)>=frbd(c,1),capos(a,1)>=frbd(c,3)),and(capos(a,1)+ko
r_g(a,1)>=frbd(c,1),capos(a,1)+kor_g(a,1)>=frbd(c,3))))
                    if
or(and(and(capos(a,2)<=frbd(c,2),capos(a,2)<=frbd(c,4)),and(capos(
a,2)+kor_g(a,2)<=frbd(c,2),capos(a,2)+kor_g(a,2)<=frbd(c,4))),and(
and(capos(a,2)>=frbd(c,2),capos(a,2)>=frbd(c,4)),and(capos(a,2)+ko
r_g(a,2)>=frbd(c,2),capos(a,2)+kor_g(a,2)>=frbd(c,4))))

                    else

                    end
                elseif
or(and(and(capos(a,2)<=frbd(c,2),capos(a,2)<=frbd(c,4)),and(capos(
a,2)+kor_g(a,2)<=frbd(c,2),capos(a,2)+kor_g(a,2)<=frbd(c,4))),and(
and(capos(a,2)>=frbd(c,2),capos(a,2)>=frbd(c,4)),and(capos(a,2)+ko
r_g(a,2)>=frbd(c,2),capos(a,2)+kor_g(a,2)>=frbd(c,4))))

                else
                    if frbd(c,3)+kor_g(a,1)<=PLV(1,1)
                        capos(a,:)=poslist2(a,b*2-1:b*2);
                        capos(a,1)=frbd(c,3);
                        poslist2(a,b*2-1:b*2)=capos(a,:);
                        c=1;
                    else
                        capos(a,:)=[9999 9999];
                        poslist2(a,b*2-1:b*2)=PLV;
                    end
                end
                c=c+1;
            end
        else
            poslist2(a,b*2-1:b*2)=PLV;
        end
    end
end


TouchP=zeros(2,nposlist);
for b=1:nposlist
    %perulangan perhitungan untuk dua orientasi (rotasi)
    for a=1:2
        if sum(poslist2(a,b*2-1:b*2)+kor_g(a,:)<=PLV)>1
            capos(a,:)=poslist2(a,b*2-1:b*2);
            if poslist2(a,b*2-1)==0
                TouchP(a,b)=TouchP(a,b)+kor_g(a,2);
            end
            if
or(poslist2(a,b*2)+kor_g(a,2)==PLV(1,2),poslist2(a,b*2)==0)
                TouchP(a,b)=TouchP(a,b)+kor_g(a,1);
            end
            c=1;
            while c<=nfrbd
```

```
                if
or(and(and(capos(a,1)<=frbd(c,1),capos(a,1)<=frbd(c,3)),and(capos(
a,1)+kor_g(a,1)<=frbd(c,1),capos(a,1)+kor_g(a,1)<=frbd(c,3))),and(
and(capos(a,1)>=frbd(c,1),capos(a,1)>=frbd(c,3)),and(capos(a,1)+ko
r_g(a,1)>=frbd(c,1),capos(a,1)+kor_g(a,1)>=frbd(c,3))))
                    if
or(and(and(capos(a,2)<=frbd(c,2),capos(a,2)<=frbd(c,4)),and(capos(
a,2)+kor_g(a,2)<=frbd(c,2),capos(a,2)+kor_g(a,2)<=frbd(c,4))),and(
and(capos(a,2)>=frbd(c,2),capos(a,2)>=frbd(c,4)),and(capos(a,2)+ko
r_g(a,2)>=frbd(c,2),capos(a,2)+kor_g(a,2)>=frbd(c,4))))
                        %TIDAK ADA YG BERIRISAN
                    else
                        %BERIRISAN DI SUMBU W SAJA
                        if poslist2(a,b*2-1)==frbd(c,3)
                            aw=0;ak=0;
                            if poslist2(a,b*2)<=frbd(c,2)
                                aw=frbd(c,2);
                            else
                                aw=poslist2(a,b*2);
                            end

                            if
poslist2(a,b*2)+kor_g(a,2)<=frbd(c,4)
                                ak=poslist2(a,b*2)+kor_g(a,2);
                            else
                                ak=frbd(c,4);
                            end
                            TouchP(a,b)=TouchP(a,b)+(ak-aw);
                        end
                    end
                elseif
or(and(and(capos(a,2)<=frbd(c,2),capos(a,2)<=frbd(c,4)),and(capos(
a,2)+kor_g(a,2)<=frbd(c,2),capos(a,2)+kor_g(a,2)<=frbd(c,4))),and(
and(capos(a,2)>=frbd(c,2),capos(a,2)>=frbd(c,4)),and(capos(a,2)+ko
r_g(a,2)>=frbd(c,2),capos(a,2)+kor_g(a,2)>=frbd(c,4))))
                    %BERIRISAN DI SUMBU H SAJA
                    if
or(poslist2(a,b*2)==frbd(c,4),poslist2(a,b*2)+kor_g(a,2)==frbd(c,2
))
                        aw=0;ak=0;
                        if poslist2(a,b*2-1)<=frbd(c,1)
                            aw=frbd(c,1);
                        else
                            aw=poslist2(a,b*2-1);
                        end

                        if poslist2(a,b*2-1)+kor_g(a,1)<=frbd(c,3)
                            ak=poslist2(a,b*2-1)+kor_g(a,1);
                        else
                            ak=frbd(c,3);
                        end
                        TouchP(a,b)=TouchP(a,b)+(ak-aw);
                    end
                else
                    %BERIRISAN DENGAN SUMBU H DAN W

                end
```

```
                c=c+1;
            end
        else

        end
    end
end

[~,prio1]=sort(TouchP,2,'descend');
priopos=[];
```

e. Heuristic 5

```
function
[priopos,nposlist,prio1,AreaP]=PosHeur5(poslist,nposlist,frbd,kor_
g,PLV,nfrbd)

%HEURISTIC LIMA
poslist2=[poslist;poslist];
capos=[9999 9999;9999 9999];
for b=1:nposlist
    %perulangan perhitungan untuk dua orientasi (rotasi)
    for a=1:2
        if sum(poslist2(a,b*2-1:b*2)+kor_g(a,:)<=PLV)>1
            capos(a,:)=poslist2(a,b*2-1:b*2);
            c=1;
            while c<=nfrbd
                if
or(and(and(capos(a,1)<=frbd(c,1),capos(a,1)<=frbd(c,3)),and(capos(
a,1)+kor_g(a,1)<=frbd(c,1),capos(a,1)+kor_g(a,1)<=frbd(c,3))),and(
and(capos(a,1)>=frbd(c,1),capos(a,1)>=frbd(c,3)),and(capos(a,1)+ko
r_g(a,1)>=frbd(c,1),capos(a,1)+kor_g(a,1)>=frbd(c,3))))
                    if
or(and(and(capos(a,2)<=frbd(c,2),capos(a,2)<=frbd(c,4)),and(capos(
a,2)+kor_g(a,2)<=frbd(c,2),capos(a,2)+kor_g(a,2)<=frbd(c,4))),and(
and(capos(a,2)>=frbd(c,2),capos(a,2)>=frbd(c,4)),and(capos(a,2)+ko
r_g(a,2)>=frbd(c,2),capos(a,2)+kor_g(a,2)>=frbd(c,4))))

                    else

                    end
                elseif
or(and(and(capos(a,2)<=frbd(c,2),capos(a,2)<=frbd(c,4)),and(capos(
a,2)+kor_g(a,2)<=frbd(c,2),capos(a,2)+kor_g(a,2)<=frbd(c,4))),and(
and(capos(a,2)>=frbd(c,2),capos(a,2)>=frbd(c,4)),and(capos(a,2)+ko
r_g(a,2)>=frbd(c,2),capos(a,2)+kor_g(a,2)>=frbd(c,4))))

                else
                    if frbd(c,3)+kor_g(a,1)<=PLV(1,1)
                        capos(a,:)=poslist2(a,b*2-1:b*2);
                        capos(a,1)=frbd(c,3);
                        poslist2(a,b*2-1:b*2)=capos(a,:);
                        c=1;
                    else
                        capos(a,:)=[9999 9999];
                        poslist2(a,b*2-1:b*2)=PLV;
```

```
                    end
                end
                c=c+1;
            end
        else
            poslist2(a,b*2-1:b*2)=PLV;
        end
    end
end

AreaP=zeros(2,nposlist);
[~,idx]=sort(frbd(:,2));
for b=1:nposlist
    pj=0;lb=0;
    %perulangan perhitungan untuk dua orientasi (rotasi)
    for a=1:2
        for c=2:nfrbd
            if and(poslist2(a,2*b-
1)>=frbd(idx(c),1),poslist2(a,2*b-1)<frbd(idx(c),3))
                if poslist2(a,2*b)<=frbd(idx(c),2)
                    lb=(frbd(idx(c),2)-poslist2(a,2*b));
                    pj=(PLV(1,1)-poslist2(a,2*b-1));
                    AreaP(a,b)=pj*lb;
                    break
                else
                    lb=(PLV(1,2)-poslist2(a,2*b));
                    pj=(PLV(1,1)-poslist2(a,2*b-1));
                    AreaP(a,b)=pj*lb;
                    break
                end
            else
                lb=(PLV(1,2)-poslist2(a,2*b));
                pj=(PLV(1,1)-poslist2(a,2*b-1));
                AreaP(a,b)=pj*lb;
                break
            end
        end
    end
end

for i=1:nposlist
    for j=1:2
        if AreaP(j,i)==0
            AreaP(j,i)=9999;
        else
        end
    end
end

[~,prio1]=sort(AreaP,2,'ascend');
priopos=[];
end
```

D. Penulisan MATLAB 2-opt

```
%2-opt
function
[opt_rute,opt_jarut,nnode]=HFtwoOPT(vh,N,n,rute,jarut,mjar,W2,PLV,
squence,VHuse,Fx,Vc)
nnode=zeros(vh*N,1); %jumlah node dalam @rute
for i=1:vh*N
    for j=1:n
        if rute(i,j)~=0
            nnode(i,1)=nnode(i,1)+1;
        else
        end
    end
end

opt_jarut=jarut;
opt_rute=rute;
djar=ones(N,1);
q=1;
while sum(djar(:,1))>=0.01
    k=1;
    jarut_lama=opt_jarut;
    for kk=1:N
        v=k;
        c=1;
        while k<=v+vh-1
            for i=1:nnode(k,1)-1
                for j=i+1:nnode(k,1)
                    rute_sem = opt_rute;
                    rute_sem(k,i)=opt_rute(k,j);
                    rute_sem(k,j)=opt_rute(k,i);

[jarut_sem]=HFjarakrute(mjar,rute_sem,vh,VHuse,Fx,Vc);

[~,noserv,heur]=TWOLHFVPReval(W2,squence,rute_sem(k,:),PLV(c,kk*2-
1:kk*2));
                    if
and(jarut_sem(kk,1)<opt_jarut(kk,1),noserv==0)
                        opt_jarut(kk,1)=jarut_sem(kk,1);
                        opt_rute(k,j)=rute_sem(k,j);
                        opt_rute(k,i)=rute_sem(k,i);
                    else
                    end
                end
            end
            k=k+1;
            if mod(k,vh)==0
                c=1;
            else
                c=c+1;
            end
        end
    end
    djar=abs(jarut_lama-opt_jarut);
    q=q+1;
end
```

E. Penulisan MATLAB 1-1 *exchange*

```
%1-1 Exchange
function
[opt_rute2,opt_jarut2,used2]=HFIIexchange(opt_rute,opt_jarut,nnode
,used,N,vh,mjar,W,cap,W2,squence,PLV,VHuse,Fx,Vc)
kk=1;
aa=[];
bbb=[];
k=1;
ii=1;
bb=1;
q=1;
opt_rute2=opt_rute;
opt_jarut2=opt_jarut;
djar=ones(N,1);
used2=used;
while sum(djar(:,1))>=0.01
    jarut_lama=opt_jarut2;
    while k<=vh*N
        c=1;
        for i=k:k+vh-1
            for j=i+1:k+vh-1
                for l=1:nnode(i)
                    for m=1:nnode(j)
                        if
mjar(opt_rute2(i,l)+1,opt_rute2(j,m)+1)<=mean(mean(mjar))
                            used_sem1=0;
                            used_sem2=0;
                            rute_sem = opt_rute2;
                            rute_sem(i,l)=opt_rute2(j,m);
                            rute_sem(j,m)=opt_rute2(i,l);

[jarut_sem]=HFjarakrute(mjar,rute_sem,vh,VHuse,Fx,Vc);

[~,noserv1,heur1]=TWOLHFVPReval(W2,squence,rute_sem(i,:),PLV(c,bb*
2-1:bb*2));

[~,noserv2,heur2]=TWOLHFVPReval(W2,squence,rute_sem(j,:),PLV(c+(j-
i),bb*2-1:bb*2));
                            if
and(jarut_sem(bb)<opt_jarut2(bb),(noserv1+noserv2)==0)
                                for a=1:nnode(i,1)

used_sem1=used_sem1+W(rute_sem(i,a)+1,4);
                                end
                                for b=1:nnode(j,1)

used_sem2=used_sem2+W(rute_sem(j,b)+1,4);
                                end
                                if used_sem1<=cap(i) &&
used_sem2<=cap(j);
                                    opt_rute2(i,l)=rute_sem(i,l);
                                    opt_rute2(j,m)=rute_sem(j,m);
                                    opt_jarut2=jarut_sem;
                                    used2(i,1)=used_sem1;
                                    used2(j,1)=used_sem2;
```

```
                                else
                                end
                            else
                            end
                        else
                        end
                    end
                end

                aa(kk,:)=[kk k i j];
                kk=kk+1;
            end
            bbb(ii,:)=[ii bb];
            ii=ii+1;
            c=c+1;
        end
        k=k+vh;
        bb=bb+1;
    end
    djar=abs(jarut_lama-opt_jarut2);
    q=q+1;
end
```

F. Penulisan MATLAB 1-0 *exchange*

```
function [opt_rute3,
opt_jarut3,used3,nnode1]=HFIOexchange(opt_rute2,opt_jarut2,nnode,u
sed2,N,vh,n,mjar,W,cap,W2,squence,PLV,VHuse,Fx,Vc)
kk=1;
aa=[];
bbb=[];
k=1;
ii=1;
bb=1;
q=1;
opt_rute3=opt_rute2;
opt_jarut3=opt_jarut2;
nnode1=nnode;
used3=used2;
djar=ones(N,1);
while sum(djar(:,1))>=0.01
    jarut_lama=opt_jarut3;
    while k<=vh*N
        c=1;
        ck=1;
        for i=k:k+vh-1
            for j=i+1:k+vh-1
                l=1;
                while l<=nnode1(i)+1
                    m=1;
                    while m<=nnode1(j)
                        if
mjar(opt_rute3(i,l)+1,opt_rute3(j,m)+1)<=mean(mean(mjar))
                            %MOVE2
                            nnode2=nnode1;
                            opt_rute3_sem=opt_rute3;
                            rute_sem = [opt_rute3 zeros(N*vh,2)];
```

```
                            rute_sem1=[zeros(N*vh,1) opt_rute3];
                            rute_sem(i,:)=[rute_sem1(i,1:l)
rute_sem1(j,m+1) rute_sem1(i,l+1:end)];
                            rute_sem(j,:)=[rute_sem1(j,1:m)
rute_sem1(j,m+2:end) 0 0];
                            rute_sem3=zeros(2,n+2);
                            rute_sem3(1,:)=rute_sem(i,:);
                            rute_sem3(2,:)=rute_sem(j,:);
                            rute_sem3(:,end)=[];
                            rute_sem3(:,1)=[];
                            rute_sem4=opt_rute3;
                            rute_sem4(i,:)=rute_sem3(1,:);
                            rute_sem4(j,:)=rute_sem3(2,:);

[jarut_sem1]=HFjarakrute(mjar,rute_sem4,vh,VHuse,Fx,Vc);

[~,noserv1,heur1]=TWOLHFVPReval(W2,squence,rute_sem4(i,:),PLV(c,ck
*2-1:ck*2));

[~,noserv2,heur2]=TWOLHFVPReval(W2,squence,rute_sem4(j,:),PLV(c+(j
-i),ck*2-1:ck*2));

                            %MOVE2
                            Nrute_sem = [opt_rute3 zeros(N*vh,2)];
                            Nrute_sem1=[zeros(N*vh,1) opt_rute3];
                            Nrute_sem(j,:)=[Nrute_sem1(j,1:l)
Nrute_sem1(i,m+1) Nrute_sem1(j,l+1:end)];
                            Nrute_sem(i,:)=[Nrute_sem1(i,1:m)
Nrute_sem1(i,m+2:end) 0 0];
                            Nrute_sem3=zeros(2,n+2);
                            Nrute_sem3(1,:)=Nrute_sem(i,:);
                            Nrute_sem3(2,:)=Nrute_sem(j,:);
                            Nrute_sem3(:,end)=[];
                            Nrute_sem3(:,1)=[];
                            Nrute_sem4=opt_rute3;
                            Nrute_sem4(i,:)=Nrute_sem3(1,:);
                            Nrute_sem4(j,:)=Nrute_sem3(2,:);

[jarut_sem2]=HFjarakrute(mjar,Nrute_sem4,vh,VHuse,Fx,Vc);

[~,noserv3,heur3]=TWOLHFVPReval(W2,squence,Nrute_sem4(i,:),PLV(c,c
k*2-1:ck*2));

[~,noserv4,heur4]=TWOLHFVPReval(W2,squence,Nrute_sem4(j,:),PLV(c+(
j-i),ck*2-1:ck*2));

                            if
and(jarut_sem1(bb)<opt_jarut3(bb),(noserv1+noserv2)==0)
                                if jarut_sem2(bb)<jarut_sem1(bb)
                                    %PILIH MOVE2
                                    nnode_sem=nnode1;
                                    nnode_sem(i)=nnode1(i)-1;
                                    nnode_sem(j)=nnode1(j)+1;
                                    used_sem1=0;
                                    used_sem2=0;
                                    for a=1:nnode_sem(i,1)
                                        if Nrute_sem3(1,a)~=0
```

```
used_sem1=used_sem1+W(Nrute_sem3(1,a)+1,4);
                                        else
                                        end
                                    end
                                    for b=1:nnode_sem(j,1)
                                        if Nrute_sem3(2,b)~=0

used_sem2=used_sem2+W(Nrute_sem3(2,b)+1,4);
                                        else
                                        end
                                    end
                                    if used_sem1<=cap(i) &&
used_sem2<=cap(j);

opt_rute3(i,:)=Nrute_sem3(1,:);

opt_rute3(j,:)=Nrute_sem3(2,:);
                                        opt_jarut3=jarut_sem2;
                                        nnode1(i)=nnode1(i)-1;
                                        nnode1(j)=nnode1(j)+1;
                                        used3(i,1)=used_sem1;
                                        used3(j,1)=used_sem2;
                                    else
                                    end
                                    break
                                else
                                    %PILIH MOVE1

opt_rute3_sem(i,:)=rute_sem3(1,:);

opt_rute3_sem(j,:)=rute_sem3(2,:);
                                    opt_jarut3_sem=jarut_sem1;
                                    nnode2(i)=nnode1(i)+1;
                                    nnode2(j)=nnode1(j)-1;

                                    nnode_sem=nnode1;
                                    nnode_sem(i)=nnode1(i)+1;
                                    nnode_sem(j)=nnode1(j)-1;
                                    used_sem1=0;
                                    used_sem2=0;
                                    for a=1:nnode_sem(i,1)
                                        if rute_sem3(1,a)~=0

used_sem1=used_sem1+W(rute_sem3(1,a)+1,4);
                                        else
                                        end
                                    end
                                    for b=1:nnode_sem(j,1)
                                        if rute_sem3(2,b)~=0

used_sem2=used_sem2+W(rute_sem3(2,b)+1,4);
                                        else
                                        end
                                    end
                                    if used_sem1<=cap(i) &&
used_sem2<=cap(j);
```

```
opt_rute3(i,:)=opt_rute3_sem(i,:);

opt_rute3(j,:)=opt_rute3_sem(j,:);
                                        opt_jarut3=opt_jarut3_sem;
                                        nnode1(i)=nnode2(i);
                                        nnode1(j)=nnode2(j);
                                        used3(i,1)=used_sem1;
                                        used3(j,1)=used_sem2;
                                    else
                                    end
                                    break
                                end
                            elseif
and(jarut_sem2(bb)<opt_jarut3(bb),(noserv3+noserv4)==0)
                                %PILIH MOVE2
                                nnode_sem=nnode1;
                                nnode_sem(i)=nnode1(i)-1;
                                nnode_sem(j)=nnode1(j)+1;
                                used_sem1=0;
                                used_sem2=0;
                                for a=1:nnode_sem(i,1)
                                    if Nrute_sem3(1,a)~=0

used_sem1=used_sem1+W(Nrute_sem3(1,a)+1,4);
                                    else
                                    end
                                end
                                for b=1:nnode_sem(j,1)
                                    if Nrute_sem3(2,b)~=0

used_sem2=used_sem2+W(Nrute_sem3(2,b)+1,4);
                                    else
                                    end
                                end
                                if used_sem1<=cap(i) &&
used_sem2<=cap(j);

opt_rute3(i,:)=Nrute_sem3(1,:);

opt_rute3(j,:)=Nrute_sem3(2,:);
                                    opt_jarut3=jarut_sem2;
                                    nnode1(i)=nnode1(i)-1;
                                    nnode1(j)=nnode1(j)+1;
                                    used3(i,1)=used_sem1;
                                    used3(j,1)=used_sem2;
                                else
                                end
                                break
                            else
                            end
                        else
                        end
                        m=m+1;
                    end
                    l=l+1;
                end
                aa(kk,:)=[kk k i j];
```

```
                kk=kk+1;
            end
            bbb(ii,:)=[ii bb];
            ii=ii+1;  c=c+1;
        end
        k=k+vh;
        ck=ck+1;
        bb=bb+1;
    end
    djar=abs(jarut_lama-opt_jarut3);
    q=q+1;
end
```

BIOGRAFI PENULIS

Penulis bernama Dominico Laksma Paramestha yang akrab dipanggil Nico. Penulis adalah anak kedua dari dua bersaudara lahir 19 Desember 1991 di kota Pemalang, Jawa Tengah. Penulis menyelesaikan TK-SMP di perguruan Pius Pemalang dan pendidikan SMA di SMAN 1 Pemalang.

Penulis menempuh pendidikan S1 di program studi Teknik Industri Universitas Atma Jaya Yogyakarta (UAJY) selama 4 tahun (2010-1014). Penulis aktif selama 2 tahun di Himpunan Mahasiswa Teknik Industri UAJY dan menjadi panitia beberapa kegiatan seperti National Marketing Research Competition (NMRC). Penulis juga aktif sebagai asisten lab di beberapa laboratorium yang ada di UAJY seperti Laboratorium Pemodelan dan Optimasi, Laboratorium Sistem Produksi, dan Laboratorium Sistem Kendali Industri. Penulis pernah melakukan kerja praktik di PT Djarum Tbk. pada bagian produksi. Penulis lulus jenjang pendidikan S1 dengan melakukan penelitian dengan judul "*Algoritma Particle Swarm Optimization untuk penyelesaian Capacitated Vehicle Routing Problem with Load Balancing*". Selama menempuh pendidikan S2 penulis melakukan magang selama 6 bulan di PT Mitra Pinasthika Mulia di bagian distribusi.

Penulis memiliki ketertarikan pada metaheuristik untuk kasus-kasus distribusi sehingga kali ini penulis melakukan penelitian tentang pengembangan algoritma *Cross Entropy Genetic Algorithm* untuk menyelesaikan salah satu varian *Vehicle Routing Problem*. Kritik, saran dan pertanyaan, penulis dapat dihubungi melalui email dlaksma@gmail.com.