TESIS – IS185401

PENYELESAIAN PERMASALAHAN *VEHICLE ROUTING* OBJEKTIF DENGAN JAMAK YANG MEMPERTIMBANGKAN KESEIMBANGAN JARAK RUTE KENDARAAN MENGGUNAKAN METODE HIPERHEURISTIK

SASMI HIDAYATUL YULIANING TYAS
NRP. 05211650010021

DOSEN PEMBIMBING I
Prof. Ir. Arif Djunaidy, M.Sc, Ph.D
NIP. 195810051986031003

DOSEN PEMBIMBING II
Ahmad Muklason, S.Kom, M.Sc, Ph.D
NIP. 198203022009121009

PROGRAM MAGISTER
DEPARTEMEN SISTEM INFORMASI
FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2019

(Halaman ini sengaja dikosongkan)

THESIS – IS185401

SOLVING MULTI-OBJECTIVE VEHICLE ROUTING PROBLEM CONSIDERING THE BALANCE OF ROUTE DISTANCE USING HYPER-HEURISTIC METHOD

SASMI HIDAYATUL YULIANING TYAS
NRP. 05211650010021

SUPERVISOR I
Prof. Ir. Arif Djunaidy, M.Sc, Ph.D
NIP. 195810051986031003

SUPERVISOR II
Ahmad Muklason, S.Kom, M.Sc, Ph.D
NIP. 198203022009121009

POSTGRADUATE PROGRAM
DEPARTEMENT OF INFORMATION SYSTEM
FACULTY OF INFORMATION TECHNOLOGY AND COMMUNICATION
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2019

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN TESIS

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M.Kom)
di
Institut Teknologi Sepuluh Nopember
Oleh:
SASMI HIDAYATUL YULIANING TYAS
NRP: 05211650010021

Tanggal Ujian: 17 Juli 2019
Periode Wisuda: September 2019

Disetujui oleh:
Pembimbing:

1. Prof. Ir. Arif Djunaidy, M.Sc,Ph.D. .....
   NIP: 195810051986031003

2. Ahmad Muklason, S.Kom, M.Sc, Ph.D. .....
   NIP: 198203022009121009

Penguji:

1. Erma Suryani, S.T, M.T, Ph.D. .....
   NIP: 197004272005012001

2. Faizal Mahananto, S.Kom, M.Eng, Ph.D. .....
   NIPH: 5200201301010

Kepala Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi

Mahendrawathi E.R., S.T, M.Sc., Ph.D.
NIP: 197610112006042001

(Halaman ini sengaja dikosongkan)

PENYELESAIAN PERMASALAHAN *VEHICLE ROUTING* DENGAN OBJEKTIF JAMAK YANG MEMPERTIMBANGKAN KESEIMBANGAN JARAK RUTE KENDARAAN MENGGUNAKAN METODE HIPERHEURISTIK

Nama mahasiswa	: SASMI HIDAYATUL YULIANING TYAS
NRP	: 05211650010021
Pembimbing 1	: Prof. Ir. Arif Djunaidy, M.Sc, Ph.D
Pembimbing 2	: Ahmad Muklason, S.Kom, M.Sc, Ph.D

ABSTRAK

Vehicle Routing Problem (VRP) adalah salah satu masalah kombinatorik yang sulit dipecahkan, sehingga dimasukkan ke dalam masalah NP-hard. VRP bertujuan untuk menghasilkan serangkaian rute terpendek dari beberapa kendaraan berkapasitas sama untuk mengunjungi beberapa pelanggan pada batas waktu tertentu. Depot adalah titik awal dan akhir rute. Karena kompleksitas kebutuhan industri, masalah VRP perlu ditingkatkan menjadi multi-tujuan. Sebagian besar penelitian VRP sebelumnya hanya meminimalkan jarak total sebagai tujuan tunggal. Oleh karena itu, dalam penelitian ini ditambahkan tujuan yang berkaitan dengan keseimbangan jarak antar rute. Dalam penelitian sebelumnya, VRP multi-tujuan diselesaikan dengan menggunakan metaheuristik. Dibutuhkan penentuan parameter dan desain algoritma khusus untuk menyelesaikan setiap domain masalah. Untuk mengatasi kekurangan ini, penelitian ini menggunakan metode hiper-heuristik untuk menyelesaikan multi-objektif VRP. Mengingat bahwa penggunaan hyper-heuristik dalam penelitian sebelumnya adalah untuk VRP objektif tunggal, maka penelitian ini juga mengusulkan hyper-heuristik untuk VRP multi-objektif. Desain algoritma yang diusulkan diimplementasikan dengan menggunakan Bahasa permrograman Java dan kerangka kerja HyFlex. Pada penelitian ini digunakan dua dataset pada tahapan uji coba yaitu dataset Solomon dan dataset Gehring dan Homberger. Masing-masing jenis dataset dipilih tiga dataset sehingga jumlah dataset yang diuji adalah enam dataset. Uji coba dilakukan untuk menentukan parameter dan menguji performa algoritma. Berdasarkan penelitian, formulasi keseimbangan jarak rute yang sesuai digunakan untuk objektif jamak VRP adalah diukur dengan simpangan baku. Penyelesaian objektif jamak VRP digunakan metode *pareto sorting* untuk menghasilkan solusi secara efektif yang diukur berdasarkan nilai fungsi objektif, indikator jumlah solusi pareto, nilai coverage dan nilai hypervolume. Hasil uji coba berdasarkan indikator jumlah solusi pareto menunjukkan bahwa algoritma Great Deluge (GD) dapat menghasilkan rata-rata solusi pareto yang lebih banyak (24) dibandingkan dengan algoritma Hill

Climbing (HC) (12). Sedangkan berdasarkan nilai coverage dan hypervolume, HC menghasilkan nilai coverage yang lebih kecil (0,06) dibandingkan dengan GD (0,8). Selain itu, nilai hypervolume HC lebih besar (0,56) dibandingkan dengan GD (0,1). Berdasarkan hasil uji coba, HC secara umum memberikan hasil yang lebih baik daripada GD untuk menyelesaikan VRP multi-objektif. Sehingga kombinasi algoritma objektif jamak Hiperheuristik dengan Hill Climbing lebih tepat untuk diimplementasikan. Berdasarkan hasil verifikasi, output kode program memberikan hasil yang sesuai dan tepat dibandingkan dengan perhitungan manual. Selain itu, hasil validasi menunjukkan bahwa solusi yang dihasilkan oleh algoritma objektif jamak hiperheuristik yang diusulkan mempunyai rute yang lebih seimbang dibandingkan dengan solusi hasil dari algoritma single objektif hiperheuristik. Pada penelitian ini dihasilkan metode dan solusi permasalahan objektif jamak dengan jarak rute yang seimbang sebagai acuan perusahaan logistik dalam menentukan rute pengiriman barang yang adil untuk setiap petugas pengiriman atau sopir

Kata Kunci: *Vehicle Routing Problem, Time Windows, keseimbangan, keadilan, Objektif jamak, Hiperheuristik, pareto sorting*

SOLVING MULTI-OBJECTIVE VEHICLE ROUTING PROBLEM CONSIDERING BALANCE OF ROUTE DISTANCES USING HYPER-HEURISTIC METHOD

By	: Sasmi Hidayatul Yulianing Tyas
Student Identity Number	: 05211650010021
Supervisor 1	: Prof. Ir. Arif Djunaidy, M.Sc, Ph.D
Supervisor 2	: Ahmad Muklason, S.Kom, M.Sc, Ph.D

ABSTRACT

Vehicle Routing Problem (VRP) is one of the combinatoric problems that is difficult to solve, so it is incorporated into an NP-hard problem. VRP aims to produce a set of shortest routes from several of the same capacity vehicles to visit several customers at a certain time limit. Depot is the starting and ending point of the route. Due to the complexity of industry needs, the VRP problem needs to be improved into a multi-objective. Most of VRP prior researches only minimize total distance as a single objective. Therefore, in this study added an objective related to the balance of distances between routes. In prior researches, multi-objective VRP was solved using metaheuristic. It requires the determination of parameters and specific algorithm design to solve each problem domain. To overcome these shortcomings, this study uses a hyper-heuristic method to complete multi-objective VRP. Given that the use of hyper-heuristics in previous studies is for single objective VRP, so this study also proposes hyper-heuristic for multi-objective VRP. The design of the proposed algorithm is implemented using the Java programming language and the HyFlex framework. In this study two datasets were used in the experiment phase, namely the Solomon dataset and Gehring and Homberger's dataset. Each dataset is selected by three datasets so that the number of datasets tested are six datasets. Experiments are conducted to determine parameters and test the performance of the algorithm. Based on the research, the distance balance formulation that is suitable for multi-objective VRP is measured by standard deviation. The multi-objective VRP solution is used the pareto sorting method to effectively produce solutions that are measured based on objective function values, indicators of the number of Pareto solutions, coverage values and hypervolume values. The results of the experiment based on the number of pareto solution indicators indicate that the Great Deluge (GD) algorithm can produce more average pareto solutions (24) compared to the Hill Climbing (HC) algorithm (12). While based on coverage and hypervolume values, HC produces a smaller coverage value (0.06) compared to GD (0.8). In addition, the hypervolume HC value is greater (0.56) compared to GD (0.1). Based on the results of the experiment, the Hill Climbing algorithm generally provides better results than the Great Deluge

algorithm for completing multi-objective VRP. So the combination of Hyperheuristic multi-objective algorithms with Hill Climbing is more appropriate to be implemented. Based on the results of verification, the program code output provides the appropriate and true results compared to manual calculations. In addition, the results of the validation indicate that the solution produced by the proposed hyperheuristic multi-objective algorithm has a more balanced route compared to the solution resulting from the hyperheuristic single objective algorithm. In this study a method and solution with a balanced route distance to multi-objective problems were obtained as a reference for logistic companies in determining fair shipping routes for each shipping officer or driver.

Kata Kunci: *Vehicle Routing Problem, Time Windows, balance, fairness, Multiobjektive, Hyperheuristic, pareto sorting.*

KATA PENGANTAR

Puji syukur kepada Allah SWT, karena dengan rahmat dan ridho-Nya penulis dapat menyelesaikan laporan Tesis yang berjudul "Penyelesaian Permasalahan Vehicle Routing dengan Objektif Jamak yang Mempertimbangkan Keseimbangan Jarak Rute Kendaraan Menggunakan Metode Hiperheuristik" sebagai salah satu syarat kelulusan Program Magister Departemen Sistem Informasi. Pada proses pengerjaan sampai terselesaikannya Tesis ini, penulis telah mendapatkan bantuan dari berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Ibu Mahendrawathi ER, S.T., M.Sc., Ph.D. selaku Kepala Departemen Sistem Informasi Institut Teknologi Sepuluh Nopember.
2. Bapak Dr. Apol Pribadi Subriadi, S.T., M.T. selaku Ketua Prodi S2 Departemen Sistem Informasi Institut Teknologi Sepuluh Nopember.
3. Bapak Prof. Ir. Arif Djunaidy, M.Sc., Ph.D. dan Bapak Ahmad Muklason S.Kom., M.Sc., Ph.D. selaku Pembimbing Tesis yang telah meluangkan waktu dan dengan sabar membimbing penulis sampai menyelesaikan Tesis ini.
4. Ibu Erma Suryani, S.T., M.T., Ph.D. dan Bapak Faizal Mahananto, S.Kom, M.Eng, Ph.D. selaku dosen penguji sidang Tesis.
5. Pihak-pihak terkait yang telah membantu hingga terselesaikannya laporan Tesis ini.

Penulis memohon maaf apabila masih terdapat banyak kekurangan pada laporan Tesis ini dan apabila terdapat kritik serta saran yang membangun dari pembaca dapat dikirimkan melalui email sasmihy.tyas@gmail.com. Semoga laporan ini dapat bermanfaat.

Surabaya, 14 Juni 2019

Penulis

(Halaman ini sengaja dikosongkan)

DAFTAR ISI

LEMBAR PENGESAHAN v
ABSTRAK vii
ABSTRACT ix
KATA PENGANTAR xi
DAFTAR ISI xiii
DAFTAR TABEL xvii
DAFTAR GAMBAR xix
DAFTAR KODE PROGRAM xxi
BAB I PENDAHULUAN 1
1.1 Latar Belakang Masalah 1
1.2 Perumusan Masalah 6
1.3 Tujuan dan Manfaat Penelitian 7
1.4 Kontribusi Penelitian 7
1.5 Batasan Penelitian 8
1.6 Sistematika Penulisan 9
BAB II DASAR TEORI DAN KAJIAN PUSTAKA 11
2.1 Dasar Teori 11
2.1.1 VRP (*Vehicle Routing Problem*) 11
2.1.2 Objektif Jamak VRP 14
2.1.3 Metode Optimasi Objektif Jamak 18
2.1.4 Hiperheuristik 21
2.1.5 HyFlex *(Hyper*heuristic *Flexible Framework)* 30
2.1.6 Indikator Uji Coba Optimasi Permasalahan Objektif Jamak 31
2.2 Kajian Penelitian Terdahulu 33
2.2.1 Penelitian Terdahulu Objektif Jamak VRP 33
2.2.2 Penelitian Terdahulu Metode Heuristik 36
BAB III METODOLOGI PENELITIAN 39
3.1. Studi Literatur 40
3.2. Pemilihan Dataset dan Formulasi Objektif Jamak VRP 40
3.3. Pengembangan algoritma Hiperheuristik Objektif Jamak VRPTW 41
3.3.1. Desain Algoritma 41
3.3.2. Metode Objektif Jamak Hiperheuristik 44
3.3.3. Implementasi pada HyFlex 47
3.4. Uji Coba 47

3.5. Analisis Hasil 49
BAB IV DESAIN DAN IMPLEMENTASI 51
4.1 Dataset Penelitian dan Formulasi Objektif Jamak VRP 51
4.1.1 Dataset Penelitian 51
4.1.2 Pembuatan Model dan Formulasi Objektif Jamak VRPTW 54
4.2 Desain Algoritma Hiperheuristik 57
4.2.1. Desain Algoritma Inisialisasi Solusi 60
4.2.2. Desain Algoritma Seleksi LLH 61
4.2.3 Desain Algoritma LLH 61
4.2.4 Desain Algoritma *Move Acceptance* 73
4.2.5 Desain Algoritma Seleksi Solusi Pareto (*Pareto Sorting*) 73
4.3 Implementasi Algoritma 73
4.3.1 Implementasi Algoritma pada Kerangka Kerja HyFlex 74
4.3.2 Fungsi -Fungsi pada HyFlex yang Dibutuhkan 75
4.3.3 Implementasi Fungsi Objektif 77
4.3.4 Implementasi Metode Hiperheuristik Objektif Jamak 79
4.3.5 Implementasi Kode Program Grafik dan Indikator Uji Coba pada Python 85
BAB V UJI COBA DAN ANALISIS HASIL 87
5.1. Lingkungan dan Tahapan Uji Coba 87
5.2. Hasil Uji Coba 88
5.2.1. Hasil Uji Coba Awal 88
5.2.2. Hasil Uji Coba Inti 90
5.3. Hasil Validasi dan Verifikasi 92
5.4. Analisis Hasil 93
5.4.1. Perbandingan Algoritma Hiperheuristik HC dan GD 93
5.4.2 Perbandingan Solusi Single Objektif dan Objektif Jamak 100
BAB VI KESIMPULAN DAN SARAN 103
6.1 Kesimpulan 103
6.2 Saran 104
6.2.1 Saran untuk Penelitian Selanjutnya 105
6.2.2 Saran untuk Manajemen Calon Pengguna 105
DAFTAR PUSTAKA 107
LAMPIRAN A KODE PROGRAM PYTHON 111
LAMPIRAN B FLOWCHART ALGORITMA 113
LAMPIRAN C REKAPITULASI NILAI COVERAGE 117
LAMPIRAN D REKAPITULASI JUMLAH PARETO FRONT 123

LAMPIRAN E REKAPITULASI NILAI HYPERVOLUME 129
LAMPIRAN F HASIL RUNNING VALIDASI 135
LAMPIRAN G HASIL VERIFIKASI 147
BIODATA PENULIS 155

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

Tabel 2. 1 Penelitian objektif jamak VRP 14
Tabel 2. 2 Penelitian objektif jamak VRP (Lanjutan) 15
Tabel 4. 1 Keterangan notasi pada formula 57
Tabel 4. 2 Fungsi-fungsi HyFlex yang dibutuhkan 75
Tabel 4. 3 Fungsi-fungsi yang ditambahkan pada HyFlex 76
Tabel 5. 1 Hasil Uji Coba Awal 90
Tabel 5. 2 Rekapitulasi Hasil Uji Coba 92

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

Gambar 2. 1 Grafik Pareto front. 20
Gambar 2. 2 Klasifikasi Hiperheuristik 22
Gambar 2. 3 Kerangka Kerja Hiperheuristik 23
Gambar 2. 4 Contoh operator two-opt pada mutation heuristik (Walker, 2015a). 25
Gambar 2. 5 Contoh operator or-opt pada mutation heuristik (Walker, 2015a)... 25
Gambar 2. 6 Contoh operator shift pada mutation heuristik (Walker, 2015a)..... 26
Gambar 2. 7 Contoh operator interchange pada mutation heuristik (Walker, 2015a)..... 26
Gambar 2. 8 Diagram kelas HyFlex..... 31
Gambar 2. 9 Grafik penjelasan hypervolume 33
Gambar 3. 1 Metodologi Penelitian 39
Gambar 3. 3 Alur pembuatan desain algoritma 42
Gambar 3. 4 Metode Hiperheuristik yang diusulkan 45
Gambar 4. 1 Contoh instance dataset..... 52
Gambar 5. 1 Tahapan uji coba 88
Gambar 5. 2 Scatterplot perbandingan HC dan GD..... 94
Gambar 5. 3 Boxplot perbandingan HC dan GD 94
Gambar 5. 4 Histogram perbandingan HC dan GD 95
Gambar 5. 5 Swarmplot perbadingan HC dan GD 95
Gambar 5. 6 Grafik Perbandingan Jumlah Front 96
Gambar 5. 7 Grafik Perbandingan Nilai Coverage 97
Gambar 5. 8 Grafik Perbandingan Nilai Hypervolume 98
Gambar 5. 9 Grafik Perbandingan Single Objektif dan Objektif jamak berdasarkan Jarak Total menggunakan algoritma HC 100
Gambar 5. 10 Grafik Perbandingan Single Objektif dan Objektif jamak berdasarkan Jarak Total menggunakan algoritma GD 101
Gambar 5. 11 Grafik Perbandingan Single Objektif dan Objektif jamak berdasarkan Simpangan baku menggunakan algoritma HC..... 101
Gambar 5. 12 Grafik Perbandingan Single Objektif dan Objektif jamak berdasarkan Simpangan baku menggunakan algoritma GD…….102

(Halaman ini sengaja dikosongkan)

DAFTAR KODE PROGRAM

Kode Program 4. 1 Pseudocode Algortima Hiperheuristik untuk permasalahan Objektif jamak (Usulan) 58
Kode Program 4. 2 Pseudocode Detail Algoritma Objektif jamak Hiperheuristik (usulan) 59
Kode Program 4. 3 Pseudocode Algoritma Metaheuristic BioSS (prior research)60
Kode Program 4. 4 Pseudocode Generate solusi awal 61
Kode Program 4. 5 Pseudocode Two-opt Mutasi 62
Kode Program 4. 6 Pseudocode Or-opt Mutasi 63
Kode Program 4. 7 Pseudocode Shift Mutasi 64
Kode Program 4. 8 Pseudocode Penyisipan Customer 64
Kode Program 4. 9 Pseudocode Interchange Mutasi 65
Kode Program 4. 10 Pseudocode Location-based Radial Ruin 66
Kode Program 4. 11 Pseudocode Shift Local Search 68
Kode Program 4. 12 Pseudocode Interchange Local Search 68
Kode Program 4. 13 Pseudocode Two-opt Local Search 69
Kode Program 4. 14 Pseudocode GENI Local Search..... 70
Kode Program 4. 15 Pseudocode Combine Crossover 71
Kode Program 4. 16 Pseudocode Combine Long 72
Kode Program 4. 17 Pseudocode Algoritma selesi solusi pareto..... 73
Kode Program 4. 18 Fungsi getFunctionValues 77
Kode Program 4. 19 Fungsi calcOF1 78
Kode Program 4. 20 Fungsi calcOF2 79
Kode Program 4. 21 Fungsi pemanggilan dari kelas lain 80
Kode Program 4. 22 variabel-variabel yang dibutuhkan 80
Kode Program 4. 23 Seleksi LLH dan move acceptance Hill Climbing 82
Kode Program 4. 24 Seleksi solusi pareto 83
Kode Program 4. 25 Menampilkan output solusi..... 83
Kode Program 4. 26 Fungsi-fungsi pendukung 84
Kode Program 4. 27 Move Acceptance Great Deluge..... 85
Kode Program 4. 28 Perhitungan nilai coverage 86
Kode Program 4. 29 Perhitungan jumlah solusi pareto 86

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

Dalam bab ini dibahas mengenai pendahuluan penelitian. Bab ini terdiri dari latar belakang permasalahan, perumusan masalah, tujuan dan manfaat penelitian, kontribusi penelitian, keterbaruan *(novelty)* penelitian, batasan dan sistematika penelitian.

1.1 Latar Belakang Masalah

Transportasi merupakan salah satu aspek penting bagi kehidupan masyarakat, terutama untuk aktivitas industri. Transportasi dalam industri berkaitan dengan aktivitas logistik, distribusi dan manajemen rantai pasok. Salah satu domain permasalahan yang berkaitan dengan hal tersebut adalah *Vehicle Routing Problem* (VRP). VRP adalah proses penentuan rute kendaraan ekonomis untuk mengirim produk yang dibutuhkan oleh masing-masing pelanggan (Eksioglu et al., 2009). VRP memiliki banyak tipe yaitu *Capacitated Vehicle Routing Problem* (CVRP), *Heterogeneous Fleet VRP* atau *Mix Flee VRP*, *VRP with Time Windows* (VRPTW)*, VRP with Pickup and Delivery, VRP with Backhauls, Multi Depot VRP, Periodic VRP*. Namun jenis yang banyak digunakan dalam penelitian adalah tipe VRPTW dan CVRP karena kedua tipe ini banyak dibutuhkan dan diimplementasikan dalam industri, terutama industri manufaktur (Braekers et al., 2016). Pada VRPTW ditentukan sekumpulan rute yang optimal pada armada kendaraan yang identik dengan kapasitas terbatas untuk mengirimkan barang permintaan dari pelanggan yang telah diketahui. Pengiriman pada masing-masing pelanggan harus dilakukan pada rentang waktu tertentu *(time windows)*(El-Sherbeny, 2010).

Sebagian besar penelitian VRPTW terdahulu berfokus pada tujuan meminimalkan jarak rute, meminimalkan jumlah kendaraan dan meminimalkan biaya operasional pengiriman barang. Beberapa diantaranya hanya menggunakan salah satu *(single objective*) dari tujuan-tujuan tersebut, padahal faktanya implementasi di industri terutama logistik dibutuhkan kombinasi beberapa tujuan (*objektif jamak*) untuk menyelesaikan permasalahan penentuan rute yang paling optimal (Dabia et al., 2013; Eksioglu et al., 2009; Jozefowiez et al., 2008). Selain itu, dalam dunia industri juga tidak hanya perlu mempertimbangkan dari segi biaya

namun juga aspek keseimbangan pada masing-masing tujuan tersebut, misalnya keseimbangan rute antar kendaraan, keseimbangan beban kerja antar pagawai atau sopir, keseimbangan kapasitas yang diangkut oleh masing-masing kendaraan dan lain-lain (Jozefowiez et al., 2008; Lee and Ueng, 1999). Aspek keseimbangan sangat perlu dipertimbangkan karena hal tersebut seringkali menimbulkan masalah pada internal perusahaan, terutama antar pegawai. Hal ini sangat berkaitan erat dengan kesejahteraan pegawai. Contoh kasus suatu perusahaan di Taiwan, setiap harinya para sopir akan menerima tugas pengiriman barang yang harus diselesaikan pada hari tersebut. Di sisi lain, pembagian tugas ini masih dilakukan secara manual dan didasarkan pada pengalaman masing-masing sopir sehingga sering terjadi terjadi ketidakadilan dalam pembagian tugas. Setiap sopir akan mendapatkan tugas satu rute, namun setiap rute tidak sama jarak tempuhnya. Hal ini menyebabkan komplain dan menurunkan kualitas pekerja karena beban kerja (jarak tempuh pengiriman) berpengaruh pada insentif atau gaji yang diterima oleh masing-masing sopir. Sehingga keseimbangan beban kerja menjadi hal yang penting untuk dijadikan salah satu tujuan pada optimasi VRP (Lee and Ueng, 1999). Beberapa penelitian VRP yang menggunakan *multi-objective* telah dilakukan oleh para peneliti. Penelitian *multi-objective* VRP dengan meminimalkan total jarak yang ditempuh dan ketidakseimbangan jarak dan beban pada masing-masing kendaraan (Baños et al., 2013c) (Baños et al., 2013a). Melián-Batista et al., (2014) menggunakan dua tujuan yaitu meminimalkan jarak tempuh dan meminimalkan selisih antara panjang rute maksimal dan minimal berdasarkan waktu. Sedangkan penelitian yang dilakukan oleh Nahum et al., (2014) meminimalkan total jarak yang ditempuh dan jumlah kendaraan. Suatu permasalahan dikatakan *objektif jamak* apabila antar tujuannya terjadi konflik, namun sebagian besar penelitian masih termasuk pada *single objective* karena tujuan yang digunakan tidak berkonflik dan diselesaikan dengan teknik agregasi. Pada penelitian ini selain menyelesaikan VRP dengan tujuan meminimalkan jarak tempuh kendaraan, tujuan lain yang juga dipertimbangkan adalah keseimbangan total jarak tempuh masing-masing kendaraan. Hal ini karena jarak tempuh merupakan aspek yang paling utama dalam VRP. Jarak tempuh dapat berpengaruh langsung terhadap penggunaan bahan bakar dan biaya pengiriman yang harus ditanggung oleh perusahaan. Formula

keseimbangan pada sebagian besar literatur (Baños et al., 2013c; Jozefowiez et al., 2007, 2005, 2002; Melián-Batista et al., 2014; Pasia et al., 2007) digunakan selisih antara nilai maksimal dan nilai minimal *(range)*. Namun sebenarnya formula untuk mengukur keseimbangan bermacam-macam dan salah satu yang terbaik adalah simpangan baku (Matl et al., 2017). Berdasarkan hasil literatur (Matl et al., 2017) diketahui bahwa belum ada penelitian VRP yang menggunakan simpangan baku sebagai pengukuran tingkat keseimbangan beban kerja masing-masing kendaraan. Sehingga pada penelitian ini akan menggunakan simpangan baku sebagai alternatif pengukuran keseimbangan selain menggunakan *range*.

Penambahan aspek keseimbangan jarak rute kendaraan akan menambah kompleksitas permasalahan VRP yang akan diselesaikan. Di samping itu, VRP termasuk pada tipe combinatorial optimization problem yang bersifat NP-hard (Laporte, 1992), permasalahan ini sulit untuk diselesaikan terutama jika data yang digunakan (pelanggan) sangat besar (Lee et al., 2010; Lenstra and Kan, 1981). Akibat kompleksitas yang tinggi dan NP-hard, maka penyelesainnya akan lebih tepat jika digunakan metode heuristik dibandingkan metode eksak (Glover and Kochenberger, 2005). Metode yang digunakan untuk menyelesaikan permasalahan multi objektif VRP pada penelitian sebelumnya adalah metaheuristik (Baños et al., 2013d, 2013a; Melián-Batista et al., 2014; Nahum et al., 2014). Metode metaheuristik menghasilkan solusi terbaik hanya pada permasalahan yang spesifik karena sangat bergantung pada pengaturan parameter. (Braekers et al., 2016). Padahal dengan semakin berkembangnya industri dan transportasi maka dibutuhkan sebuah metode generik yang dapat memberikan solusi optimum pada berbagai macam permasalahan, tidak terkhusus pada permasalahan tertentu. Selain itu, metode heuristik memiliki kekurangan yaitu besanya kemungkinan untuk terjebak pada *local optima*. Hal ini menjadi peluang dalam topik penelitian VRP untuk menciptakan algoritma generik yang dapat menyelesaikan berbagai tipe permasalahan VRP (Braekers et al., 2016). Hiperheuristik adalah algoritma yang bersifat general yang dapat menyelesaikan banyak permasalahan karena metode ini bekerja dalam menentukan *low level* heuristik yang tepat untuk suatu permasalahan, tidak secara langsung berhubungan dengan solusi. Selain itu algoritma ini juga

disebut dengan generic algorithm karena parameter tuningnya dilakukan secara otomatis, tidak seperti metode heuristik dan metaheuristik yang membutuhkan waktu lama untuk dapat menentukan parameter yang tepat.

Hiperheuristik tidak hanya dapat digunakan untuk menyelesaikan *single objective* problem tetapi juga dapat menghasilkan solusi yang baik untuk *multi-objective* problem (Burke et al., 2013). Permasalahan yang memiliki tujuan yang lebih dari satu, dalam hal ini adalah Objektif jamak VRP hanya dapat diselesaikan dengan metode Hiperheuristik yang khusus untuk multi objective. Hal ini karena solusi yang dihasilkan oleh permasalahan dengan tujuan tunggal akan berbeda dengan solusi yang dihasilkan oleh permasalahan multi objektif. Permasalahan dengan tujuan tunggal akan menghasilkan satu solusi yang optimal sedangkan permasalahan yang multi-objective akan menghasilkan sekumpulan solusi pareto optimal. Apabila objektif ditambah maka yang menjadi fokusan utama tidak hanya optimumnya solusi, tetapi bagimana menemukan kompromi yang baik (*trade off*) antar objektif tersebut (Melián-Batista et al., 2014). Oleh karena itu, perlu digunakan pendekatan pareto optimal. Pendekatan ini digunakan apabila antar fungsi tujuan tidak bisa selaras atau terjadi konflik. Contoh konflik antar objektif pada permasalahan minimasi adalah ketika salah satu objektif diminimalkan maka objektif yang lain menjadi sebaliknya, sehingga perlu dicari solusi yang dapat memenuhi semua objektifnya. Pada penelitian sebelumnya telah digunakan *Multi Objective Evolutionary Algorithm* (MOEA) untuk menyelesaikan permasalahan objektif jamak dengan hasil yang unggul (Li et al., 2018, 2017). MOEA yang sering digunakan adalah NSGA II, PSEA. MOEA juga digunakan sebagai *low level* heuristik pada kerangka kerja multi-objective Hiperheuristik, namun sebagian besar permasalahan yang dapat diselesaikan dengan menggunakan MOEA standar adalah permasalahan yang bersifat kontinu dan memiliki batasan (*constraint*) yang tidak rumit. Sedangkan permasalahan yang akan diselesaikan pada penelitian ini adalah VRP yang merupakan permasalahan kombinatorik dengan *constraint* yang ketat dan banyak. Sehingga tidak bisa digunakan MOEA standar. Oleh karena itu perlu dilakukan modifikasi terhadap metode Hiperheuristik standar agar dapat digunakan untuk menyelesaikan permasalahan yang memiliki banyak objektif. Pada penelitian

ini diusulkan metode pembobotan secara random terhadap objective function untuk menghasilkan solusi yang pareto optimal (Jensen, 2003; Knowles et al., 2001). Selain itu juga dilakukan pengurutan solusi untuk menjaring solusi-solusi yang tidak terdominasi oleh solusi yang lain.

Hiperheuristik *Flexible Framework* (HyFlex) adalah salah satu kerangka kerja yang berupa *soft*ware untuk mengembangkan Hiperheuristik dan algoritma lain yang memiliki tujuan umum. HyFlex dibangun dengan menggunakan Bahasa pemrograman Java. HyFlex telah digunakan untuk menerapkan Hiperheuristik dengan berbagai problem domain yaitu *Permutation Flow Shop, One dimensional bin packing*, dan penjadwalan personnel (Burke et al., 2010). Kemudian pada tahun 2015, Walker membuat desain problem domain VRP untuk diterapkan pada HyFlex sebagai penelitian thesisnya (Walker, 2015a). Sehingga sampai sekarang modul domain permasalahan yang telah diimplementasikan berjumlah 6 jenis yaitu *Boolean Satisfiability (SAT), One-dimensional Bin Packing (BP), Permutation Flow Shop (FS), Personnel Scheduling (PS), Travelling Salesman Problem (TSP), Vehicle Routing Problem (VRP*). Selain itu pada framework HyFlex yang terkini telah diimplementasikan untuk permasalahan *single objective* (Melián-Batista et al., 2014).

Berdasarkan hasil studi literatur yang telah dijelaskan sebelumnya dan permasalahan yang ada pada dunia nyata industri logistik, maka pada penelitian ini akan dilakukan pengembangan problem domain dari single objective menjadi objektif jamak VRP dengan menambahkan aspek keseimbangan jarak rute kendaraan. Objektif yang digunakan pada penelitian ini adalah meminimalkan jarak tempuh masing-masing kendaraan dan meminimalkan ketidakseimbangan jarak rute antar kendaraan. Tingkat keseimbangan beban kerja dihitung dengan rumus simpangan baku. Selanjutnya dilakukan modifikasi algoritma Hiperheuristik agar dapat menyelesaikan permasalahan VRP yang objektif jamak dan diimplementasikan pada HyFlex untuk dilakukan eksperimen serta pengujian terhadap beberapa macam dataset.

1.2 Perumusan Masalah

Berdasarkan latar belakang di atas, permasalahan yang menjadi dasar pengerjaan penelitian ini adalah perlunya menambahkan aspek keseimbangan beban kerja kendaraan sebagai objektif VRP. Pada sebagian besar literatur yang menyelesaikan VRP, keseimbangan diukur dengan mencari selisih antara nilai maksimal dan nilai minimalnya *(range)*. Sebenarnya pengukuran yang lebih baik adalah menggunakan simpangan baku karena nilainya langsung dipengaruhi oleh setiap hasil distribusi dan tidak hanya mempertimbangkan nilai tertentu yang ekstrim (Matl et al., 2017). Solusi yang dihasilkan akan lebih seimbang. Formula fungsi objektif menggunakan simpangan baku belum tersedia pada literatur sehingga perlu dirancang formula fungsi objektif baru agar dapat merepresentasikannya. Permasalahan yang kedua adalah belum adanya metode multi objektif Hiperheuristik yang dapat menyelesaikan permasalahan yang kombinatorik dengan *constraint* yang banyak seperti VRP. Pada penelitian sebelumnya digunakan MOEA standar sebagai strategi untuk menghasilkan solusi pareto, namun metode MOEA standar hanya mampu diterapkan untuk metode multi objektif Hiperheuristik yang menyelesaikan permasalahan kontinu. Oleh karena itu perlu disusun strategi untuk menghasilkan sekumpulan solusi pareto pada metode *single objective* Hiperheuristik yang telah ada agar dapat menyelesaikan permasalahan yang objektif jamak VRP. Berdasarkan permasalahan-permasalahan yang telah diuraikan, maka secara umum pertanyaan pengerjaan penelitian yang akan diselesaikan pada penelitian ini adalah sebagai berikut.

a. Bagaimana formulasi keseimbangan (*balance*) jarak rute kendaraan yang sesuai pada objektif jamak VRP?
b. Bagaimana Algoritma Hiperheuristik yang efektif untuk menyelesaikan objektif jamak VRP yang mempertimbangkan keseimbangan beban kerja kendaraan?

Secara khusus permasalahan yang akan dselesaikan pada metode Hiperheuristik adalah bagaimana kombinasi metode pemilihan *low level* heuristik dan penerimaan solusi yang tepat untuk menghasikan solusi yang optimal. Pemilihan kombinasi terbaik didasarkan pada efektifitas metode dalam menghasilkan solusi optimal.

Indikator efektifitas diukur dengan indikator jumlah solusi pareto, indikator coverage dan indikator hypervolume.

1.3 Tujuan dan Manfaat Penelitian

Tujuan dari penelitian ini adalah untuk menyelesaikan *Vehicle Routing Problem* (VRP) dengan metode Objektif jamak Hiperheuristik sehingga dihasilkan jarak tempuh kendaraan yang minimal dan seimbang antar kendaraan. Solusi yang dihasilkan berupa sekumpulan solusi pareto optimal. Problem domain dan metode yang telah dikembangkan kemudian diimplementasikan pada salah satu framework Hiperheuristik yaitu *HyFlex*.

Apabila tujuan tersebut tercapai, maka manfaat yang diperoleh adalah dapat memberikan referensi terhadap praktisi atas solusi pareto yang dihasilkan. Solusi-solusi dengan mempertimbangkan aspek keseimbangan dapat membantu pengambil keputusan dalam merumuskan keputusan yang lebih adil untuk masing-masing pegawai yang bekerja sebagai pengendara. Selain itu, keseimbangan beban kerja yang dihasilkan akan mampu menyelesaikan permasalahan kesenjangan beban kerja antar pengendara kendaraan dan meminimalkan biaya operasional kendaraan yang dibutuhkan. Manfaat secara teoritis yang akan dihasilkan adalah dapat memberikan alternatif strategi dalam menyelesaikan permasalahan objektif jamak VRP dengan menggunakan metode Hiperheuristik.

1.4 Kontribusi Penelitian

Kontribusi yang diberikan oleh penelitian ini yang pertama adalah penyelesaian VRP dengan menggunakan pendekatan multi objektif dan mempertimbangkan aspek keseimbangan jarak rute kendaraan sehingga dihasilkan sekumpulan solusi pareto optimal. Berdasarkan penelitian terdahulu, sebagian besar penelitian belum menyelesaikan permasalahan VRP yang benar-benar objektif jamak. Hal ini karena objektif-objektif yang digunakan pada permasalahan tersebut bisa diselaraskan dan diselesaikan dengan teknik agregasi. Padahal objektif jamak yang sesungguhnya adalah permasalahan yang objektifnya saling berkonflik, artinya apabila salah satu dari objektif tersebut menghasilkan solusi yang lebih baik

maka objektif yang lainnya akan menghasilkan solusi yang lebih buruk. Sehingga ciri khas dari permasalahan objektif jamak adalah dihasilkannya solusi pareto. Pada penelitian ini digunakan dua objektif yaitu meminimalkan jarak tempuh kendaraan dan menyeimbangkan jarak tempuh antar kendaraan. Pengukuran keseimbangan pada penelitian sebelumnya digunakan formula *range*. Sedangkan pada penelitian ini juga akan digunakan formula simpangan baku sebagai alternatif strategi untuk menghasilkan solusi yang lebih seimbang. Pada akhir penelitian diharapkan dapat diketahui teknik pengukuran keseimbangan beban kerja yang lebih optimal.

Penelitian ini mengusulkan metode multi objektif Hyperheuristik yang dapat digunakan untuk menyelesaikan objektif jamak VRP. Metode ini lebih bersifat general karena dapat menyelesaikan permasalahan VRP dengan dataset yang berbeda. Pada penelitian sebelumnya, multi objektif VRP diselesaikan dengan metode metaheuristik yang terbukti memiliki kelemahan sangat bergantung pada pengaturan parameter dan hanya dapat digunakan untuk permasalahan atau dataset yang spesifik. Selain itu, *state of the art* multi objektif Hiperheuristik hanya dapat digunakan untuk permasalahan yang kontinu dan tidak memiliki banyak *constraint*. Sedangkan pada kehidupan nyata juga terdapat permasalahan optimasi yang kombinatorik dan miliki banyak *constraint* seperti permasalahan penjadwalan dan VRP. Kontribusi yang diberikan melalui metode yang diusulkan adalah membangun dan mengembangkan solusi pareto dengan strategi pembobotan fungsi objektif yang selalu berubah-ubah pada setiap iterasi yang dilakukan. Kemudian dilakukan modifikasi terhadap pengurutan solusi-solusi untuk menghasilkan solusi yang tidak terdominasi oleh solusi lain.

1.5 Batasan Penelitian

Pada penelitian ini terdapat ruang lingkup yang menjadi batasan dalam penelitian yang dilakukan. Batasan penelitian antara lain :

1) Jenis VRP yang digunakan pada penelitian ini adalah VRP *with time windows*, yaitu mempertimbangkan batasan jangka waktu dalam memberikan layanan (mengantar barang) pada masing-masing pelanggan.

2) Dataset yang digunakan pada penelitian ini terdiri dari dua macam dataset yaitu *Solomon* dataset dan *Gehring and Homberger* dataset.
3) Penelitian ini menggunakan dua objektif yaitu meminimalkan jarak tempuh masing-masing kendaraan dan meminimalkan ketidakseimbangan jarak rute kendaraan.
4) Fungsi objektif terkait keseimbangan (*balance*) beban kerja kendaraan ditinjau dari total jarak atau rute yang harus ditempuh oleh masing-masing kendaraan, sedangkan kapasitas berat maksimal yang harus dibawa oleh masing-masing kendaraan menjadi salah satu batasan (*hard constraint*) yang harus dipenuhi.
5) Perhitungan waktu tempuh masing-masing kendaraan tidak memperhatikan kemacetan, diasumsikan lalu lintas yang dilalui kendaraan lancar.
6) Framework yang digunakan dalam proses implementasi adalah *HyFlex*.

1.6 Sistematika Penulisan

Sistematika penulisan proposal penelitian ini adalah sebagai berikut:

1) Bab 1 Pendahuluan

Bab ini berisi penjelasan tentang pendahuluan penelitian yang terdiri dari dari latar belakang permasalahan, perumusan masalah, tujuan dan manfaat penelitian, kontribusi penelitian, batasan dan sistematika penelitian.

2) Bab 2 Kajian Pustaka

Bab ini berisi tentang hasil kajian terhadap teori maupun penelitian-penelitian terdahulu. Dasar teori dan kajian penelitian terdahulu digunakan sebagai dasar dan landasan dalam melakukan penelitian ini.

3) Bab 3 Metodologi Penelitian

Bab ini berisi uraian dan penjelasan terkait tahapan-tahapan sistematis dalam mengerjakan penelitian ini. Pada bagian ini juga dijelaskan mengenai input dan output dari masing-masing tahapan penelitian.

4) Bab 4 Daftar Pustaka

Bab ini berisi daftar referensi yang digunakan pada penelitian ini. Referensi dapat berupa jurnal, buku, maupun artikel ilmiah.

(Halaman ini sengaja dikosongkan)

BAB II
DASAR TEORI DAN KAJIAN PUSTAKA

Pada bab ini akan dibahas mengenai kajian pustaka dan dasar teori yang mendukung dalam pengerjaan penelitian. Bab ini terdiri dari hasil kajian teori tentang *Vehicle Routing Problem* (VRP), *objektif jamak Vehicle Routing Problem with Time Windows* (MOVRPTW), Metode Optimasi Objektif jamak, konsep Hiperheuristik dan HyFlex. Sedangkan dari hasil kajian penelitian terdahulu akan diuraikan tentang state of the art objektif jamak VRP dan Hiperheuristik.

2.1 Dasar Teori

Pada penelitian ini dibutuhkan konsep-konsep teori yang dijadikan dasar dalam pengerjaan penelitian. Dasar teori yang menjadi landasan penelitian adalah terkait dengan konsep dasar VRP, Objektif jamak VRP, Metode Optimasi Objektif jamak, Hiperheuristik dan kerangka kerja Hiperheuristik yang fleksibel (HyFlex).

2.1.1 VRP (*Vehicle Routing Problem*)

VRP (*Vehicle Routing Problem*) adalah salah satu domain permasalahan yang berkaitan dengan penentuan rute kendaraan. VRP juga dapat dideskripsikan sebagai permasalahan yang menciptakan sekumpulan rute yang optimal dari satu atau lebih depot menuju kota atau pelanggan yang tersebar secara geografis untuk memenuhi batasan *(constraint)*(Laporte, 1992). VRP dikembangkan dari konsep permasalahan klasik *Travelling Salesman Problem* (TSP) (Walker, 2015a). Namun yang membedakan antara VRP dan TSP adalah terutama pada jumlah kendaraan yang digunakan pada masing-masing permasalahan tersebut. TSP menggunakan kendaraan tunggal (satu rute), sedangkan VRP menggunakan beberapa kendaraan sehingga juga dihasilkan rute yang lebih dari satu sesuai dengan jumlah kendaraannya. Karakteristik umum VRP yang lainnya adalah sejumlah kendaraan tersebut harus memberikan servis kepada sekumpulan pelanggan yang jumlahnya juga telah ditentukan. Setiap kendaraan akan memulai perjalanannya pada suatu depot dan pada akhirnya harus kembali ke depot setelah mengunjungi pelanggan. Namun variasi VRP juga ada yang menggunakan depot yang lebih dari satu (Braekers et al., 2016).

Tujuan yang ingin dicapai pada VRP sangat bervariasi pada penelitian-penelitian sebelumnya, namun yang paling sering digunakan adalah meminimalkan jarak yang harus ditempuh setiap kendaraan dan meminimalkan jumlah kendaraam yang digunakan (Braekers et al., 2016). Namun akhir-akhir ini para peneliti juga mulai tertarik untuk meneliti VRP dengan mempertimbangkan aspek keseimbangan pada tujuannya, sehingga tidak hanya meminimalkan jarak atau jumlah kendaraan namun juga menambah tujuannya untuk menyeimbangkan jarak rute untuk setiap kendaraan dan jumlah beban yang diangkut masing-masing kendaraan. Hal ini berawal dari permasalahan kesenjangan antar pegawai yang dihadapi oleh beberapa perusahaan logistik, salah satunya di Cina (Lee and Ueng, 1999). Selain tujuan yang bervariasi, tipe VRP juga sangat bervariasi seiring dengan berkembangnya permasalahan di kehidupan nyata. Beberapa tipe VRP yang telah diteliti oleh penelitian terdahulu adalah *Capacitated Vehicle Routing Problem* (CVRP), *Heterogeneous Fleet VRP* atau *Mix Flee VRP*, *VRP with Time Windows, VRP with Pickup and Delivery, VRP with Backhauls, Multi Depot VRP, Periodic VRP*. Namun tipe yang banyak digunakan dalam penelitian dan mendapat perhatian lebih dari komunitas akademisi adalah tipe *VRP with Time Windows* (VRPTW) dan *Capacitated Vehicle Routing Problem* (CVRP) karena tipe ini banyak dibutuhkan dan diimplementasikan dalam industri, terutama industri manufaktur (Braekers et al., 2016; Walker, 2015a). VRPTW juga menjadi fokusan dalam penelitian ini.

Perbedaan antar tipe VRP berdasarkan pada tambahan batasan *(constraint)* yang digunakan pada masing-masing tipe VRP. Pada VRPTW, selain mempertimbangkan batasan yang ada pada VRP pada umumnya, juga memperhatikan aspek waktu (Braekers et al., 2016). Setiap pelanggan memiliki rentang waktu khusus *(time windows)* yang terdiri dari waktu mulai tersedianya pelanggan menerima service *(start time)* dan batas akhir waktu suatu pelanggan dapat menerima service *(end time)*. Hal ini bukan berarti kendaraan harus sampai pada pelanggan dalam rentang waktu tersebut, namun rentang waktu *(time windows)* ini hanya digunakan sebagai patokan kendaraan untuk dapat memulai memberikan service kepada pelanggan tertentu sesuai dengan *time windows* yang dimiliki pelanggan tersebut. Apabila kendaraan datang terlebih dahulu sebelum

start time tetap diperbolehkan namun akan ada konsekuensi waktu tunggu *(waiting time)* yang akan mempengaruhi besarnya total waktu operasi suatu kendaraan. Selain itu setiap pelanggan juga memiliki waktu layanan (*service time*), sehingga aturannya *service* tidak harus selesai sebelum *end time*, namun harus dimulai antara *start time* dan *end time*. Hal ini menjadi *hard constraint* pada VRPTW, solusi yang *feasible* pada permasalahan ini harus memenuhi *constraint* tersebut. (Walker, 2015a).

Tipe kedua yang digunakan pada penelitian ini adalah *Capacitated Vehicle Routing Problem* (CVRP). VRP pada tipe ini lebih berfokus pada kapasitas yang dibawa oleh masing-masing kendaraan. Tambahan batasan atau *constraint* yang digunakan pada tipe ini adalah adanya maksimal kapasitas untuk masing-masing kendaraan, besarnya kapasitas ini dapat sama atau berbeda pada masing-masing kendaraan (Borgulya, 2008; Braekers et al., 2016). Namun pada penelitian ini digunakan batasan setiap kendaraan memiliki kapasitas maksimal yang sama besarnya. Setiap pelanggan memiliki nilai demand yang juga dapat direpresentasikan ke dalam kapasitas atau beban yang harus diangkut oleh kendaraan. Batasan ini hampir sama dengan yang ada pada *Bin Packing Problem* (Walker, 2015a).

VRPTW dan CVRP merupakan permasalahan optimasi kombinatorik yang NP-hard (Non deterministic polynomial) (Braekers et al., 2016) karena ruang pencariannya sangat besar dan tidak dapat ditentukan waktu pemrosesan yang pasti untuk dapat menghasilkan solusi yang paling optimal. Sehingga hanya dapat diselesaikan dengan pendekatan heuristik. Berangkat dari kedua tipe VRP tersebut maka pada penelitian ini digabungkan keduanya untuk menyelesaikan objektif jamak VRP yang mempertimbangkan keseimbangan beban kerja kendaraan. Sehingga batasan *(constraint)* yang digunakan pada penelitian ini juga sama seperti yang digunakan pada kedua tipe tersebut. Dasar teori pada kedua tipe ini menjadi acuan dalam membuat formulasi dan representasi permasalahan yang akan diselesaikan pada penelitian ini.

2.1.2 Objektif Jamak VRP

Penelitian VRP telah berkembang dari VRP objektif tunggal menjadi objektif jamak VRP. Hal ini terjadi karena tiga tujuan yaitu untuk mengembangkan permasalahan yang digunakan dalam penelitian agar meningkatkan penggunaannya dalam praktik, mengembangkan permasalahan klasik dan untuk mempelajari kasus nyata dalam kehidupan yang fungsi tujuannya telah diidentifikasi dan disesuaikan oleh pengambil keputusan. Perkembangan fungsi tujuan pada VRP berjalan seiring dengan berkembangnya permasalahan nyata dan munculnya berbagai tipe VRP. Berbagai macam variasi penggunaan tujuan telah dilakukan pada penelitian-penelitian sebelumnya. Khususnya pada permasalahan VRP telah dirangkum perkembangan variasi tujuan yang digunakan oleh para peneliti dan ditunjukkan pada Tabel 2.1. Tabel ini bersumber dari (Jozefowiez et al., 2008) dan berdasarkan dari hasil literatur yang dilakukan pada penelitian ini.

Tabel 2. 1 Penelitian objektif jamak VRP

No	Peneliti	Jenis Permasalahan	Metode	Tujuan
1	(Ribeiro and Lourenc¸o, 2001)	Multiperiod VRP	Aggregation, Iterated Local Search	Meminimalkan jarak tempuh, mengoptimalkan keseimbangan jumlah pelanggan yang dikunjungi
2	(Jozefowiez et al., 2007, 2005, 2002)	VRP dengan keseimbangan rute	MOEA, tabu search	Miminimalkan jarak tempuh, mengoptimalkan keseimbangan jarak
3	(Geiger, 2001)	VRPTW	Pendekatan pareto, GA	Meminimalkan total jarak, meminimalkan penyimpangan dr aturan time windows, meminimalkan jumlah pelanggaran, meminimalkan jumlah kendaraan
4	(Tan et al., 2006)	VRPTW	Pendekatan pareto	Meminimalkan jarak tempuh, meminimalkan jumlah *constraint* yang dilanggar, meminimalkan jumlah kendaraan
5	(Baran and Schaerer, 2003)	VRPTW	Ant Colony	Meminimalkan total waktu, meminimalkan total waktu pengiriman, meminimalkan jumlah kendraaan
6	(Tan et al., 2006)	VRPTW	Pendekatan pareto, Hybrid GA	Meminimalkan total jarak, meminimalkan jumlah kendaraan
7	(Ombuki et al., 2006)	VRPTW	Pendekatan pareto (wighted sum), GA	Meminimalkan total jarak, meminimalkan jumlah kendaraan

Tabel 2. 2 Penelitian objektif jamak VRP (Lanjutan)

No	Peneliti	Jenis Permasalahan	Metode	Tujuan
8	(El-Sherbeny, 2001)	VRPTW, diadaptasi dari studi kasus perusahaan transportasi Belgia	Pendekatan pareto, Simulated Annealing	Meminimalkan total waktu, mengoptimalkan keseimbangan jarak, memaksimalkan fleksibilitas, meminimalkan waktu tunggu, meminimalkan julah truk, meminimalkan jumlah truk terbuka, meminimalkan jumlah truk terbuka, meminimalkan waktu kerja yang tidak terpakai
9	(Zografos and Androutsopoulos, 2004)	VRPTW untuk pengiriman produk berbahaya	Agregasi, heuristik	Meminimalkan jarak, meminimalkan risiko
10	(Mourgaya Virapatrin, 2004)	Multiperiod VRP	Heuristik	Mengoptimalkan klasterisasi pelanggan, mengoptimalkan keseimbangan beban
11	(Qi et al., 2015)	VRPTW	MOEA (metaheuristik)	Meminimalkan jarak dan meminmalkan jumlah kendaraan
12	(Wang et al., 2016)	VRPTW and Simultaneously Delivery and Pickup	MO Local search, MO memetic algorithm (metaheuristik)	Meminimalkan: jumlah kendaraan, total jarak, makespan, waktu tunggu, waktu tunda.
13	(Chiang and Hsu, 2014)	VRPTW	Evolutionary Algorithm, pendekatan pareto (metaheuristik)	Meminimalkan jumlah kendaraan dan total jarak
14	(Pasia et al., 2007)	VRP	Pendekatan pareto (metaheuristik)	Meminimalkan jarak dan keseimbangan rute
15	(Melián-Batista et al., 2014)	VRPTW untuk studi kasus di Terenife	Pendekatan metaheuristik dan pareto	Meminimalkan total jarak dan menyeimbangkan beban kerja sopir.
16	(Baños et al., 2013d, 2013a)	VRPTW	Pendekatan metaheuristik	Meminimalkan jarak untuk mengantarkan barang dan meminimalkan ketidakseimbangan beban kerja (jarak dan beban kerja)

Berdasarkan hasil rekapitulasi pada Tabel 2.1, maka dapat diketahui bahwa sebagian besar penelitian menggunakan tujuan meminimalkan jarak tempuh masing-masing kendaraan karena jarak merupakan poin penting yang sangat mempengaruhi proses pengiriman barang pada VRP. Namun dalam dekade terakhir para peneliti mulai mempertimbangkan keseimbangan beban kerja dalam penelitian

VRP (Borgulya, 2008) dengan ditandai dengan warna merah muda pada baris Tabel 2.1. Hal ini karena pentingnya kesejahteraan pegawai dalam mempengaruhi tercapainya target perusahaan khususnya dalam hal logistik (Lee and Ueng, 1999). Beberapa penelitian yang mengangkat tentang aspek keseimbangan dengan mempertimbangkan beberapa hal yaitu jarak atau rute, jumlah pelanggan dan kapasitas yang dikirimkan oleh masing-masing kendaraan. Pada penelitian ini digunakan tujuan meminimalkan keseimbangan beban kerja dengan mempertimbangkan jarak tempuh kendaraan. Selain itu pada penelitian ini juga menggunakan tujuan meminimalkan jarak tempuh masing-masing kendaraan sebagai tujuan yang pertama. Mengingat penelitian ini merupakan penelitian terkait objektif jamak VRP.

2.1.2.1 Konsep keseimbangan (balance)

Pada dua dekade ini, perkembangan penelitian tentang model dan metode untuk VRP lebih mempertimbangkan aspek keseimbangan. Aspek ini lebih berfokus pada keadilan dalam mengalokasikan beban kerja maupun sumberdaya yang dibutuhkan. Terdapat beberapa cara pengukuran keseimbangan yang digunakan pada beberapa penelitian sebelumnya, seperti simpangan baku, minmax, range dan lain-lain.

Pada literatur (Matl et al., 2017) membahas berbagai perbedaan cara pengukuran yang digunakan pada penelitian-penelitian terkait keseimbangan beban kerja untuk VRP. Pengukuran yang baik adalah yang memenuhi 8 kriteria, yaitu inequality relevance, transitivity, scale invariance, translation invariance, population independence, anonymity, monotonicity, Pigou-Dalton (PD). Penjelasan lebih lengkap terdapat pada Matl et al., 2017. Cara pengukuran yang banyak digunakan adalah dengan menghitung range, namun yang paling memenuhi semua kriteria yang telah disebutkan sebelumnya adalah dengan menggunakan simpangan baku. Kedua cara pengukuran memiliki kelebihan dan kekurangan masing-masing.

Perhitungan dengan menggunakan range adalah mengukur keseimbangan dengan menghitung selisih antara hasil yang terbesar dan hasil yang terkecil. Formula range ditunjukkan pada Rumus 2.1.

$$\max x_i - \min x_i \tag{2.1}$$

Cara tersebut merupakan cara yang paling sederhana, namun memiliki kekurangan yaitu kurang bisa merepresentasikan perbedaan pada setiap nilai yang dihasilkan. Range hanya berfokus pada nilai yang terbesar dan terkecil. contohnya apabila ada nilai [10,9,8,1] dan [5,10,2,2] maka yang dianggap lebih seimbang adalah yang kedua padahal solusi yang pertama tidak seimbangan hanya pada satu nilai.

Cara kedua yang dianggap lebih baik adalah menggunakan simpangan baku dalam menghitung tingkat keseimbangan beban kerja. Pada Rumus 2.2 ditunjukkan rumus perhitungan simpangan baku.

$$\sqrt{(\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2} \tag{2.2}$$

Cara ini memiliki keuntungan yang dapat menutupi kekurangan perhitungan range, namun kelemahan yang paling terlihat pada cara perhitungan ini adalah tingginya kompleksitas komputasinya jika dibandingkan dengan range. Namun hasil yang didapatkan lebih merepresentasikan tingkat keseimbangan antar nilai yang dibandingkan, tidak hanya berdasarkan pada nilai tertentu.

Diantara penelitian pada Tabel 2.1, terdapat beberapa yang menggunakan aspek keseimbangan sebagai tujuannya. Penelitian yang dilakukan oleh Baños et al., 2013c; Jozefowiez et al., 2007, 2005, 2002; Melián-Batista et al., 2014; Pasia et al., 2007, menggunakan cara perhitungan range dalam menentukan tingkat keseimbangan. Sedangkan Mourgaya menggunakan metode min max untuk mengukur keseimbangan. Min max mengukurnya dengan hanya melihat hasil yang paling buruk dari solusi yang dihasilkan. Min diperhatikan dan dioptimasi apabila digunakan pada permasalahan maksimasi dan sebaliknya apabila pada permasalahan minimasi yang dipertimbangkan adalah nilai yang maksimal. Penelitian yang dilakukan oleh Ribeiro dan Lorenco tidak menggunakan metode

range maupun min max. Berdasarkan hasil literatur (Matl et al., 2017) maka didapatkan bahwa belum ada penelitian VRP yang menggunakan simpangan baku sebagai pengukuran tingkat keseimbangan beban kerja masing-masing kendaraan. Sehingga pada penelitian ini akan digunakan cara perhitungan standard deviasi untuk menyelesaikan permasalahan VRP dengan keseimbangan jarak rute.

2.1.3 Metode Optimasi Objektif Jamak

Metode yang digunakan untuk menyelesaikan permasalahan optimasi objektif jamak akan berbeda dengan metode untuk optimasi tujuan tunggal. Hal ini karena pada permasalahan objektif jamak harus mempertimbangkan lebih dari satu tujuan untuk menghasilkan solusi yang optimal. Beberapa penelitian telah mengusulkan berbagai metode untuk menyelesaikan permasalahan objektif jamak. Metode yang paling banyak digunakan oleh penelitian terdahulu adalah metode skalar (agregation) dan metode pareto. (Jozefowiez et al., 2008).

2.1.3.1 Metode Skalar (Aggregation)

Salah satu metode skalar yang paling banyak digunakan dalam penelitian adalah metode agregasi atau pengumpulan. Pada metode ini dilakukan operasi penambahan pada semua objektif yang digunakan. Setiap objektif memiliki nilai bobot yang harus ditentukan di awal. Contoh model yang menggunakan metode aggregrasi adalah ditunjukkan pada Rumus 2.3.

$$\sum_{i=1}^{n} w_i OF_i = w_1 OF_1 + w_2 OF_2 + \ldots + w_n OF_n \tag{2.3}$$

Penentuan bobot dapat ditentukan dengan mengacu pada kebutuhan perusahaan. Namun penentuannya tidak dapat dilakukan dengan mudah karena harus meminta pendapat kepada praktisi atau pengambil keputusan dan setiap perusahaan dapat berbeda-beda. Hal ini menjadi kelemahan dari metode ini, meskipun di sisi lain komputasinya tidak kompleks. Selain itu, metode ini juga tidak bisa menghasilkan semua solusi Pareto Optimal, misalnya hanya menemukan solusi pada *convex hull* Pareto set yang optimal (Jozefowiez et al., 2008). Metode ini tidak hanya digunakan pada permasalahan objektif jamak namun juga digunakan untuk permasalahan dengan tujuan tunggal. Caranya dengan memecah tujuan tunggal tersebut menjadi

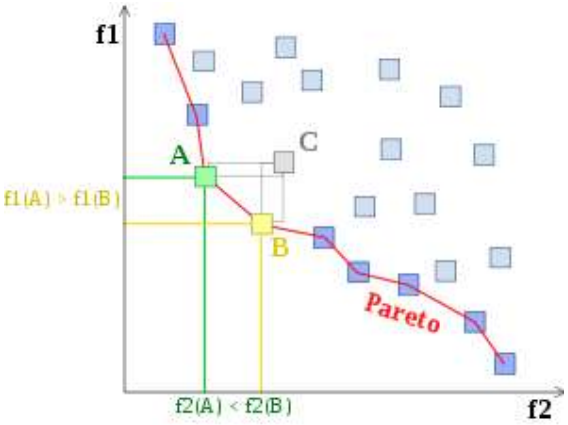
beberapa tujuan sehingga menjadi seperti objektif jamak. Metode ini dikembangkan oleh (Jensen, 2003; Knowles et al., 2001). Jensen dan Kowles melakukan multiobjetivization dengan menciptakan helper objective untuk menjadikan seolah-olah permasalahannya menjadi objektif jamak. Metode ini dapat mengatasi terjadinya *local optima* karena akan menghasilkan variasi solusi yang lebih banyak jika dibandingkan dengan hanya menggunakan tujuan tunggal yang sebenarnya. Selanjutnya metode agregasi dengan pembobotan ini akan menjadi dasar pengembangan algoritma Hiperheuristik yang akan diusulkan pada penelitian ini. Hal ini bertujuan untuk menghasilkan solusi pareto yang lebih optimal. Penjelasan terkait usulan metode lebih lengkap terdapat pada Bab Metodologi. Pada bagian selanjutnya akan dijelaskan mengenai konsep pareto optimal sebagai dasar pemilihan solusi yang akan digabungkan dengan metode pembobotan agregasi pada penelitian ini untuk menghasilkan sekumpulan solusi pareto yang optimal.

2.1.3.2 Konsep Pareto Optimal

Permasalahan Optimasi yang lebih mencerminkan kondisi nyata seringkali digambarkan dengan permasalahan yang memiliki banyak tujuan (objektif jamak). Hal yang menjadi fokusan pada banyak penelitian ini adalah bagaimana menyelesaikan permasalah multi-tujuan tersebut. Berdasarkan beberapa penelitian sebelumnya (Baños et al., 2013a, 2013d; Chiang and Hsu, 2014; Jozefowiez et al., 2008), diketahui bahwa terdapat dua alternatif yang dapat dilakukan untuk menyelesaikan beberapa objektif jamak dalam waktu yang bersamaan.

Alternatif yang pertama adalah menggunakan pendekatan agregasi atau juga disebut dengan teknik skalar (Jozefowiez et al., 2008). Pada pendekatan ini semua tujuan digabungkan menjadi satu objective dengan kombinasi formula matematis. Namun pendekatan ini memiliki kelemahan yang sulit untuk diatasi yaitu mempertimbangkan salah satu tujuan yang lebih penting dibandingkan dengan tujuan yang lainnya. Hal ini berhubungan dengan pembobotan masing-masing tujuan yang sulit untuk ditentukan secara tepat, terutama apabila informasi atau pengetahuan pada permasalahan nyata vehicle routing tidak cukup sebagai bahan pertimbangan (Tan et al., 2006).

Alternatif yang kedua adalah pendekatan pareto optimal. Pendekatan ini dapat digunakan untuk menyelidiki *trade-off* antar tujuan yang digunakan pada permasalahan optimasi. *Pareto optimality* mempunyai arti pembagian alokasi tujuan dikatakan optimal ketika tidak mungkin membuat suatu tujuan menjadi lebih baik tanpa membuat tujuan yang lain menjadi lebih buruk. Sehingga dalam satu solusi dapat optimal pada suatu tujuan (lebih baik) namun pada tujuan yang lain tidak lebih optimal (lebih buruk) dari tujuan yang pertama. Pada permasalahan objektif jamak, solusi yang dihasilkan tidak hanya satu solusi namun sekumpulan solusi (Pareto set) yang memenuhi konsep *pareto optimality*. Metode optimasi pareto menghasilkan hubungan antar solusi berdasarkan hubungan dominansinya dengan ketentuan yaitu suatu solusi dikatakan mendominasi solusi yang lainnya jika dan hanya jika solusi S_1 lebih baik daripada solusi S_2 minimal pada satu tujuan dan tidak lebih jelek pada tujuan yang lainnya. Pada Gambar 2.1 ditunjukkan titik-titik yang menjadi solusi pareto optimal. Titik-titik yang menghubungkan garis warna merah disebut dengan pareto frontier, yaitu solusi yang tidak didominasi oleh solusi yang lainnya. Sebagai contoh pada titik C bukan merupakan pareto frontier karena solusi titik C didominasi oleh titik A dan B. Sehingga dapat dikatakan bahwa titik A dan B merupakan contoh dari pareto frontier atau solusi yang tidak saling mendominasi.



Gambar 2. 1 Grafik Pareto front.

Oleh karena itu jika semua tujuan pada objektif jamak VRP dianggap sama pentingnya, maka dapat digunakan konsep pareto untuk mencari sekumpulan solusi

yang tidak didominasi yang sulit ditemukan pada permasalahan yang kompleks. Pendekatan pareto tidak digunakan untuk mencari solusi yang paling optimal pada semua objektif namun dengan pendekatan tersebut akan dihasilkan sekumpulan solusi yang optimal sehingga memberikan pilihan atau alternatif solusi bagi pengambil keputusan (*decision maker*). Oleh karena itu algoritma objektif jamak yang menggunakan konsep pareto memiliki keuntungan pada penggunaan secara praktis, yaitu dapat menghasilkan sekumpulkan solusi dengan *trade-off* yang digunakan pengambil keputusan pada perusahaan untuk memilih satu solusi berdasarkan kriteria yang spesifik.

2.1.4 Hiperheuristik

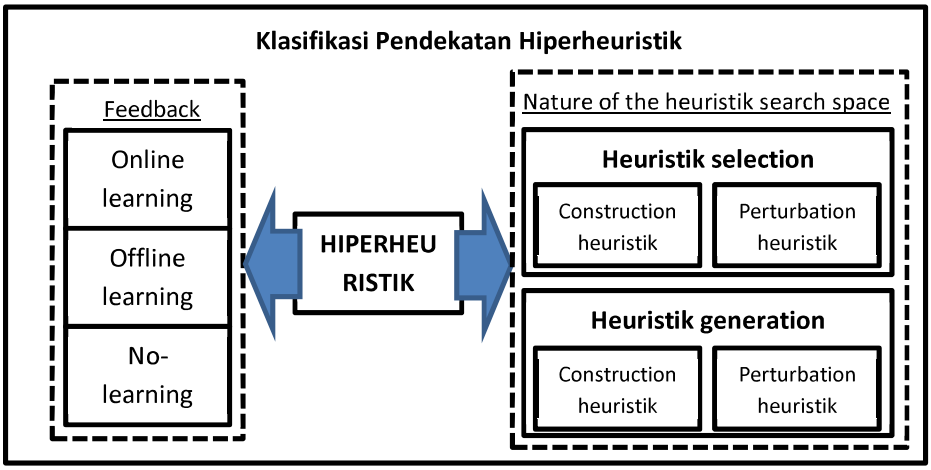
Hiperheuristik adalah algoritma yang diciptakan untuk mencapai generalitas dalam menyelesaikan berbagai domain permasalahan (Burke et al., 2013). Hiperheuristik memilih metode heuristik yang terbaik untuk menyelesaikan setiap domain permasalahan. Metode ini dapat menutupi kekurangan dari metode yang sebelumnya telah ada yaitu metode heuristik dan metaheuristik. Kedua metode tersebut hanya dapat digunakan untuk permasalahan yang spesifik atau permasalahan tunggal dalam sekali desain algoritma. Metode heuristik memiliki kekurangan karena kemungkinan solusi yang dihasilkan terjebak dalam *local optima*. Namun kekurangan dapat diselesaikan dengan metode metaheuristik dengan mampu mencapai *global optima*. Meskipun dapat mencapai *global optima*, namun pembuatan algoritma untuk suatu permasalahan dibutuhkan pengalaman dan pemahaman yang tinggi terhadap permasalahan yang akan diselesaikan. Selain itu, metaheuristik juga sensitif karena apabila ada perubahan komponen problem domain maka algoritma yang digunakan sebelumnya harus dirancang dan dimodifikasi sesuai dengan permasalahan yang baru. Hal ini menjadi motivasi peneliti untuk menciptakan algoritma yang lebih fleksibel dan general untuk berbagai permasalahan seperti Hiperheuristik.

Hiperheuristik dapat diklasifikasikan menjadi beberapa macam berdasarkan umpan baliknya (feedback) dan karakter ruang pencarian *low level* heuristiknya seperti yang ditunjukkan oleh Gambar 2.2. Berdasarkan umpan baliknya terdapat tiga jenis yaitu :

a. *Online learning*, yaitu pembelajaran dilakukan selama algoritma mencari solusi dari suatu dataset permasalahan.
b. *Offline learning*, yaitu ketika pembelajaran dilakukan diluar proses pencarian solusi.
c. *No learning*, yaitu ketika solusi dicari dan dihasilkan tidak berdasarkan sejarah pencarian dan solusi sebelumnya.

Berdasarkan karakter ruang pencarian *low level* heuristiknya dapat dibagi menjadi dua macam yaitu :

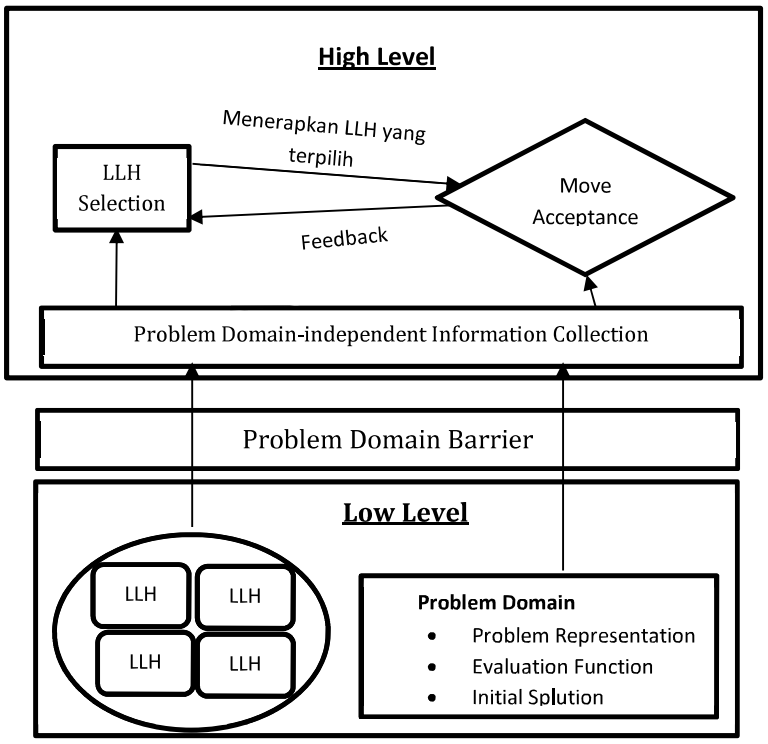
a. *Heuristic selection*, yaitu metode yang digunakan untuk memilih *low level* heuristik yang telah disediakan.
b. *Heuristic generation*, yaitu metode yang digunakan untuk membangun *low level* heuristik baru dengan memodifikasi atau menggabungkan *low level* heuristik yang telah ada sebelumnya.



Gambar 2. 2 Klasifikasi Hiperheuristik

Pada kedua jenis Hiperheuristik ini terdapat dua jenis *low level* heuristik yang bekerja sesuai dengan metode untuk mendapat solusinya. Metode yang pertama adalah *constructive* LLH, menghasilkan solusi dengan membangunnya mulai dari nol kemudian mengembangkan sedikit demi sedikit hingga menjadi satu kesatuan solusi yang utuh. Sedangkan metode yang kedua adalah *perturbative* LLH, yaitu berawal dari inisial solusi yang dihasilkan secara random maupun dengan metode tertentu kemudian dikembangkan dan dimodifikasi sampai

dihasilkan solusi yang *feasible*. Inisial solusi pada *perturbative* tidak harus *feasible* pada awalnya karena biasanya dihasilkan secara random, namun kemudian dimodifikasi untuk menjadikannya *feasible*. Kedua metode ini memiliki kelebihan kekurangan masing-masing. *Constructive* LLH lebih cocok digunakan untuk permasalahan yang memiliki *constraint* cukup banyak dan sulit untuk dihasilkan solusi yang *feasible* dengan metode random. Namun untuk permasalahan yang cenderung sedikit *constraint* nya maka cukup digunakan *perturbative* LLH. Biasanya peneliti akan menggabungkan kedua metode ini, *constructive* LLH hanya digunakan untuk menghasilkan inisial solution yang *feasible*, kemudian modifikasi dan pengembangannya digunakan *perturbative* LLH.



Gambar 2. 3 Kerangka Kerja Hiperheuristik

Terdapat dua tahapan inti pada metode Hiperheuristik yaitu memilih *low level* heuristik dan menentukan penerimaan atau penolakan terhadap solusi yang dihasilkan oleh *low level* heuristik yang telah dipilih *(move acceptance)*(Burke et al., 2013). Kerangka kerja baku dari Hiperheuristik ditunjukkan pada Gambar 2.3. High level adalah bagian dari Hiperheuristik yang berfokus untuk mengelola *low*

level heuristik, sedangkan *Low level* adalah bagian dari Hiperheuristik yang berfokus pada domain permasalahan dan menghasilkan solusi. Apabila dianalogikan, Hiperheuristik atau high level adalah sebagai manajer yang mempekerjakan tim pekerja heuristik (LLH). *Low level* heuristik adalah sebagai pekerjanya (Kendall and Hussin, 2005). Manajer bertugas memilih pekerja untuk menyelesaikan permasalahan tertentu sedangkan pekerja bertugas mengerjakan tugas yang diberikan oleh manajer. Seorang manajer tidak perlu tahu bagaimana pekerja mengerjakan pekerjaannya, namun manajer harus tahu dan mampu mengenali pekerja yang mampu menyelesaikan pekerjaan dengan baik dan kapan pekerjaan yang baik dapat diselesaikan. Performa pekerja kemungkinan ada yang baik maupun buruk untuk suatu pekerjaan, tugas manajer adalah memilih atau menggabungkan pekerja-pekerja dalam satu tim untuk menghasilkan pekerjaan yang baik. Manajer harus memantau hasil pekerjaan para pekerja sehingga dapat memperlajari kemampuan masing-masing pekerja dan sebagai pertimbangan dalam memilih pekerja atau anggota tim untuk pekerjaan selanjutnya.

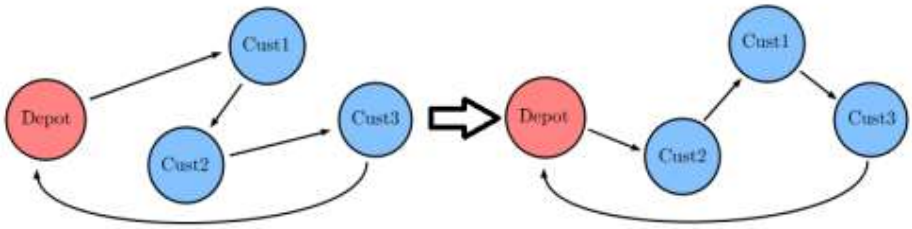
Metode Hiperheuristik yang baik adalah metode yang dapat menyeimbangkan antara intesifikasi dan diversifikasi. Intensifikasi adalah metode yang hanya mencari solusi yang terbaik, tujuannya untuk meningkatkan kualitas dari solusi yang dihasilkan oleh algoritma. Sedangkan diversifikasi adalah metode yang menerima solusi yang lebih buruk terlebih dahulu untuk menghasilkan solusi yang paling optimal, tujuan dari metode ini adalah untuk mendapatkan solusi yang lebih bervariasi sehingga tidak terjebak paa *local optima*.

2.1.4.1 Low level Heuristik *(LLH)*

Salah satu tahapan pada metode hiperheuristik adalah memilih *low level* heuristik yang dapat menghasilkan solusi yang optimal untuk setiap domain permasalahan. LLH adalah metode yang berfokus untuk menghasilkan solusi yang dapat memenuhi *constraint* dan fungsi objektif dari domain permasalahan (Walker, 2015a). Setiap domain permasalahan memiliki LLH masing-masing. Pada domain VRP terdapat empat LLH yang telah tersedia pada HyFlex, yaitu *mutation* heuristik*, crossover* heuristik*, ruin-recreate* heuristik dan *local search* heuristik (Burke et al., 2009; Walker, 2015a).

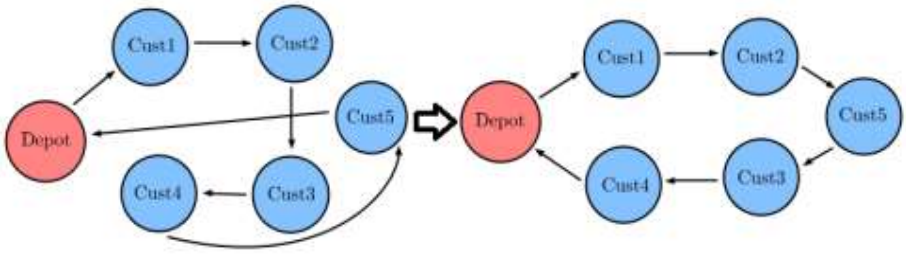
Mutation heuristik adalah tipe LLH yang melakukan sedikit perubahan terhadap solusi dengan menukar, mengubah, menghapus, menambahkan atau menghapus komponen solusi. Pertimbangan penting ketika mendesain mutation heuristik adalah memastikan jumlah variasi pemindahan simpul banyak dan pemindahan tidak hanya dilakukan pada satu simpul. Pada heuristik ini terdapat empat macam operator yaitu :

a. *Two-opt*, yaitu melakukan modifikasi minor pada solusi dengan menukar posisi dua buah simpul (*customer*) pada satu rute sehingga tidak mempengaruhi total panjang rute. Contoh penerapannya ditunjukkan pada Gambar 2.4 yaitu menukar posisi *customer* 1 dan 2.



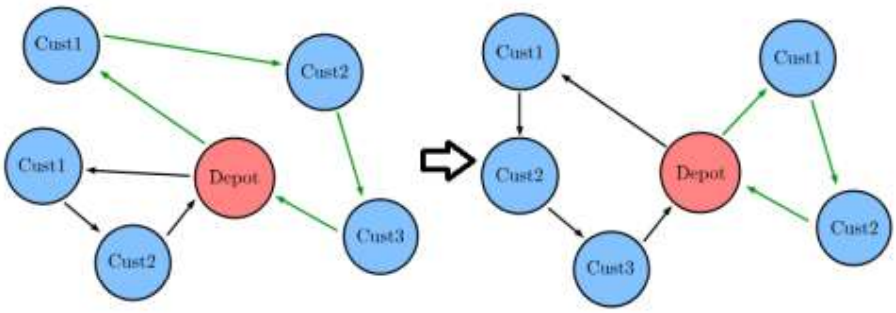
Gambar 2. 4 Contoh operator two-opt pada mutation heuristik (Walker, 2015a).

b. *Or-opt*, cara kerjanya hampir sama dengan *two-opt*. Pada operator ini dipilih dua simpul (*customer*) yang saling berurutan pada suatu rute, kemudian dipindah secara bersamaan pada suatu lokasi di rute yang sama seperti ditunjukkan pada Gambar 2.5. *Customer* 3 dan 4 dipindah letaknya secara bersamaan. Penerapan operator ini pada suatu solusi tidak mempengaruhi total panjang rute.



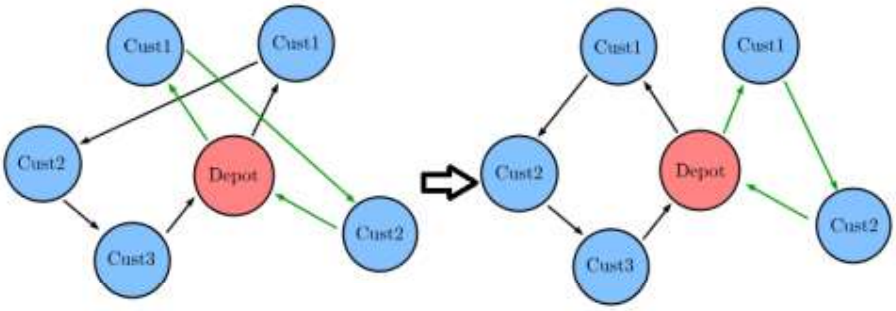
Gambar 2. 5 Contoh operator or-opt pada mutation heuristik (Walker, 2015a).

c. *Shift*, yaitu mengubah letak sebuah simpul (*customer*) dari suatu rute ke rute yang lainnya. Pemilihan simpul dan rute dilakukan secara random. Pada Gambar 2.6 dicontohkan bahwa *customer* 1 dipindahkan dari rute 1 ke rute 3, sehingga jumlah simpul dapat berkurang pada suatu rute dan bertambah pada rute yang lainnya. Hal ini akan mempengaruhi panjang rute.



Gambar 2. 6 Contoh operator shift pada mutation heuristik (Walker, 2015a).

Interchange adalah menukar dua buah simpul (*customer*) dari rute yang terpisah. Jenis operator ini dapat mengubah total jarak rute. Contoh penerapannya ditunjukkan pada Gambar 2.7.



Gambar 2. 7 Contoh operator interchange pada mutation heuristik (Walker, 2015a).

Ruin-recreate heuristik adalah jenis heuristik yang cara kerjanya merusak sebagian solusi dan membangun atau membuatnya kembali setelah itu. Heuristik ini berbeda dengan mutasi karena pada heuristik ini dapat menggabungkan heuristik lainnya untuk membangun kembali solusi.

Local search heuristik secara iteratif melakukan perubahan kecil terhadap solusi, menerima solusi yang lebih baik atau tidak memburuk sampai dicapai lokal

optima(Burke et al., 2009). Perbedaan heuristik ini dengan mutasi adalah adanya iterasi yang terjadi dan memberikan jaminan untuk menghasilkan solusi yang lebih baik. Operator yang digunakan pada local search hampir sama dengan operator yang digunakan pada mutasi yaitu *shift, interchange, two-opt* dan GENI. Operator GENI adalah operator yang cara kerjanya hampir sama dengan shift, perbedaannya pada proses penempatan simpul pada rute yang lain. Pada shift, penempatan didasarkan pada kemungkinan lokasi terbaik pada suatu rute. Sedangkan pada GENI, simpul ditempatkan ditempatkan diantara dua simpul lain yang saling berdekatan.

Crossover heuristik memilih dua solusi sebagai masukan dan akan dihasilkan solusi baru yang diturunkan dari masukan. Pada heuristik ini terdapat dua macam operator yaitu *combine* dan long *combine* .*Combine* adalah menghasilkan solusi baru hasil dari kombinasi antara dua solusi yang menjadi masukan. Kedua solusi masukan disebut sebagai orang tua, sedang solusi baru disebut anak. Perbedaan utama operator *combine* dan long *combine* adalah strategi memilih rute, pada *combine* dipilih rute secara acak sedangkan pada *long combine* dipilih rute sesuai dengan kualitasnya. Rute yang lebih panjang akan dipilih karena akan menghasilkan total jarak yang lebih kecil dan juga menghasilkan jumlah rute yang lebih sedikit (Walker, 2015a).

2.1.4.2 Metode Pemilihan Low level Heuristik

Tahapan pertama yang dilakukan pada metode Hiperheuristik setelah dihasillkan solusi awal adalah memilih LLH. Pemilihan LLH dilakukan di setiap iterasi yang dilakukan sampai mencapai batas waktu tertentu. Pada penelitian sebelumnya telah banyak metode pemilihan LLH yang diusulkan dan mampu menghasilkan solusi yang optimal seperti *Random Choice* (RC), *Fixed Sequence* (FS), *Choice Function* (CF) (Li et al., 2017).

Random Choice (RC) adalah metode pemilihan LLH secara random (Li et al., 2017). Metode ini merupakan metode yang paling sederhana diantara metode-metode yang lainnya. Selain itu, metode ini menjadi metode yang menjadi dasar

bagi metode lain dalam memilih LLH pada awal iterasi dan sebagai acuan dalam menilai performa LLH.

Fixed Sequence (FS) mempertimbangkan urutan LLH yang akan digunakan pada setiap iterasi. Metode ini bertujuan untuk memilih urutan LLH yang terbaik untuk dataset tertentu (Burke et al., 2013). Urutan ini ditentukan pada awal sebelum dilakukan proses iterasi selanjutnya sehingga setiap LLH yang terpilih dalam urutan dieksekusi secara berurutan sesuai urutan terbaik yang telah ditentukan.

Choice Function (CF) adalah metode pemilihan LLH yang diusulkan oleh (Maashi et al., 2014). Pada metode ini digunakan penilaian masing-masing LLH berdasarkan skor yang merepresentasikan performa dari LLH tersebut. Pemilihan dilakukan terhadap LLH yang memiliki skor yang tertinggi pada setiap iterasi. Metode ini cenderung lebih unggul dibandingkan metode seleksi LLH yang lain karena mempertimbangkan intensifikasi dan diversifikasi. Intensifikasi dilakukan dengan memilih LLH yang memiliki performa terbaik, sedangkan diversifikasi dilakukan dengan memberi kesempatan kepada LLH yang belum pernah terpilih untuk menghasilkan solusi. Indikator untuk diversifikasi digunakan detik CPU sejak suatu LLH dipilih. Intensifikasi dilakukan melalui dua tahapan yaitu memberikan peringkat berdasarkan indikator hypervolume, RNI, UD dan EA, kemudian di tahapan kedua digunakan frekuensi suatu LLH dapat mencapai peringkat terbaik pada indikator tertentu. Indikator performa yang digunakan pada metode ini adalah :

a. *Algorithm Effort* (EA), mengukur upaya komputasi suatu algoritma untuk menghasilkan sekumpulan solusi pareto optimal. Nilai antara 0 sampai tak terhingga, semakin kecil nilai EA maka performa LLH akan semakin baik.
b. *Ratio of non dominated individuals* (RNI), mengevaluasi pecahan individu yang tidak didominasi pada populasi. Nilainya antara 0 sampai 1. Apabila RNI=1, artinya semua individu pada populasi tidak didominasi.
c. *Size of space covered (Hypervolume)*, mengevaluasi ukuran ruang fungsi objektif yang dicakup oleh solusi di sekitar *Pareto Optimality Front* (POF).

Nilainya antara 0 sampai dengan tak terhingga, semakin besar nilainya maka akan semakin baik.

d. *Uniform Distribution of a non dominated population* (UD), mengevaluasi distribusi individu yang tidak didominasi melebihi POF. Nilainya antara 0 sampai dengan 1, semakin besar nilainya maka akan semakin baik.

2.1.4.3 Metode Peneriman Solusi (Move Acceptance)

Tahapan inti kedua dalam kerangka kerja Hiperheuristik adalah menentukan apakah menerima atau menolak solusi yang dihasilkan oleh LLH. Selain mengelola LLH, tugas lain dari Hiperheuristik adalan mengelola solusi yang telah dihasilkan oleh LLH (Burke et al., 2013). Metode penerimaan solusi telah banyak dikembangkan oleh peneliti pada penelitian sebelumnya. Pada bagian ini akan dijelaskan secara singkat tentang metode penerimaan yang telah banyak digunakan pada literatur.

Metode yang pertama adalah menerima semua solusi yang dihasilkan oleh LLH yang terpilih. Pada metode ini memiliki keuntungan dapat menyeimbangkan antara diversifikasi dan intensifikasi karena menerima solusi yang lebih baik maupun solusi yang lebih buruk dari solusi sebelumnya sehingga dapat dikatakan tidak ada strategi apapun pada metode ini (Li et al., 2017).

Great Deluge Acceptance (GDA) adalah metode yang dirancang untuk permasalahan objektif tunggal. Inti dari strateginya adalah menerima solusi yang lebih baik dan lebih buruk dari ambang batas yang telah ditentukan. Nilai ambang batas akan meningkat secara linier dengan parameter peningkatan yang tetap pada setiap penerimaan solusi. Oleh karena akan digunakan untuk permasalahan yang objektif jamak maka dilakukan modifikasi oleh (Maashi et al., 2015). Modifikasi dilakukan dengan menambahkan metrik D sebagai indikator apakah kandidat populasi diterima atau ditolak. Metrik D mengukur perbedaan cakupan ruang fungsi objektif antara dua pareto front. D (A,B) memiliki arti ruang yang dilingkupi pareto front A namun tidak untuk pareto front B.

Best Acceptance adalah salah satu cara sederhana dan banyak digunakan pada penelitian. Metode ini dilakukan dengan hanya menerima solusi yang lebih

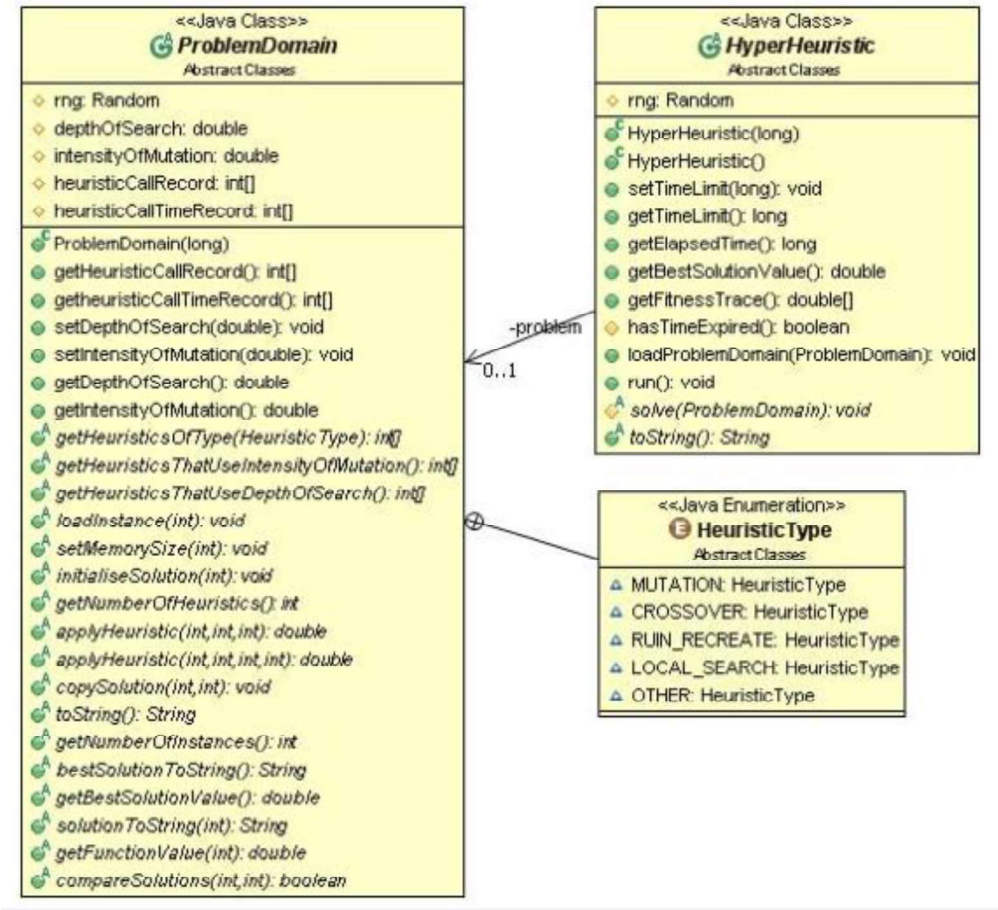
baik dari solusi yang sebelumnya. Pada setiap iterasi dilakukan perangkingan terhadap solusi yang telah diterima sebelumnya. Apabila solusi yang baru lebih baik daripada solusi yang berada pada rangking teratas, maka solusi baru tersebut akan diterima (Li et al., 2017). Setelah itu, akan dilakukan penghapusan solusi yang berada pada rangking terbawah karena posisinya pada penyimpanan akan digantikan oleh solusi yang baru diterima. Namun apabila solusi baru ditolak, maka tidak dilakukan penghapusan.

2.1.5 HyFlex *(Hyper*heuristic *Flexible Framework)*

HyFlex (*Hyperheuristic Flexible framework*) adalah salah satu kerangka kerja perangkat lunak yang didesain untuk mengembangkan, mengujicobakan dan membandingkan algoritma pencarian heuristik yang bersifat general seperti Hiperheuristik. Pada konsep Hiperheuristik telah diketahui bahwa terdapat problem domain barrier antara *low level* heuristik dengan heuristik, sehingga untuk menyesuaikan konsep ini maka desain kerangka kerja HyFlex terdiri dari dua class yaitu kelas *Hyperheuristic* dan kelas *Problem Domain*. Selain itu, juga terdapat sebuah kelas yang digunakan untuk mengeksekusi *low level* heuristik atau heuristik yang digunakan pada kerangka kerja untuk menghasilkan solusi. Diagram kelas dari kerangka kerja HyFlex ditunjukkan pada Gambar 2.4. Pada diagram kelas dapat diketahui bahwa terdapat hubungan atau relasi antara kelas *Problem Domain* dengan kelas Hiperheuristik dan kelas tipe heuristik. Informasi independen terkait problem domain dikumpulkan melalui hubungan antara kelas Problem Domain dan Hiperheuristik. Hubungan antara keduanya adalah asosiasi berarah yang berarti satu kelas digunakan oleh kelas yang lainnya. Hiperheuristik dapat digunakan oleh 1 atau 0 problem domain. Sedangkan hubungan antara kelas heuristik dan problem domain berarti banyak heuristik dapat digunakan di banyak problem domain.

Kerangka kerja ini dibuat dengan menggunakan bahasa pemrograman Java. Berbagai permasalahan optimasi kombinatorik telah diterapkan pada kerangka kerja, diantaranya adalah *Boolean Satisfiability (SAT), One-dimensional Bin Packing (BP), Permutation Flow Shop (FS), Personnel Scheduling (PS), Travelling Salesman Problem (TSP), Vehicle Routing Problem (VRP*). Sedangkan beberapa *low level* heuristik yang digunakan untuk menghasilkan solusi yaitu *mutation*

heuristik, *crossover* heuristik, *ruin-recreate* heuristik, dan *local search* atau h*ill climbing* heuristik.



Gambar 2. 8 Diagram kelas HyFlex

State of the art penerapan Hiperheuristik pada HyFlex hanya sebatas pada permasalahan optimasi untuk tujuan tunggal, sedangkan untuk permasalahan optimasi multi-tujuan hanya diterapkan untuk permasalahan penjadwalan (*timetabling*). Sehingga ke depannya diperlukan pengembangan dan penerapan Hiperheuristik untuk permasalahan optimasi multi-tujuan yang lainnya, seperti *Vehicle Routing Problem* yang akan dilakukan pada penelitian ini.

2.1.6 Indikator Uji Coba Optimasi Permasalahan Objektif Jamak

Indikator uji coba merupakan ukuran untuk membandingkan performa antar algoritma yang diujicobakan. Tidak sesederhana optimasi single objektif yang

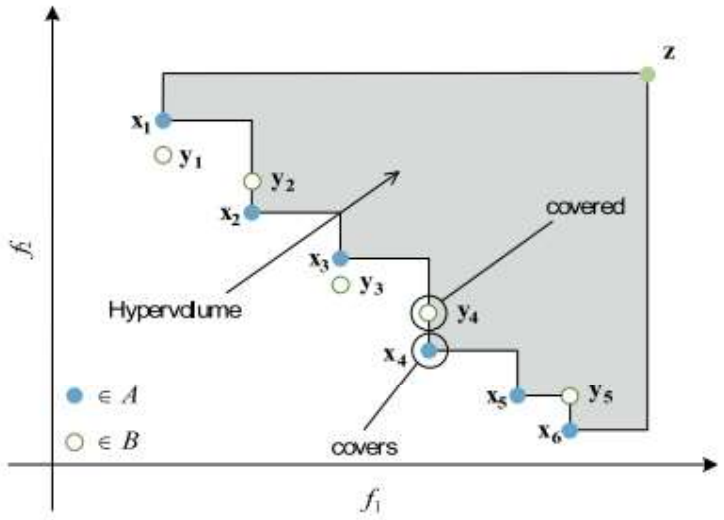
hanya menggunakan indikator nilai fungsi tujuan tunggal, pada optimasi permasalahan objektif jamak perlu digunakan beberapa indikator untuk merepresentasikan kemampuan algoritma dalam menghasilkan solusi yang optimal. Berdasarkan referensi dan penelitian sebelumnya (Baños et al., 2013b, 2013a; Garcia-Najera and Bullinaria, 2011; Melián-Batista et al., 2014) terdapat tiga indikator yang umum digunakan dalam mengukur performa algoritma yaitu jumlah solusi pareto, *coverage* dan *hypervolume*.

Indikator yang pertama adalah jumlah solusi pareto. Pada optimasi objektif jamak tidak hanya dihasilkan solusi tunggal tetapi himpunan solusi-solusi yang memenuhi fungsi tujuan. Indikator ini mengukur kemampuan algoritma dalam menemukan solusi yang tidak terdominasi oleh solusi lain (*non-dominated solution*). Semakin banyak solusi yang dihasilkan maka akan lebih baik performanya.

Coverage adalah indikator performa yang diukur berdasarkan persentase jumlah solusi dari suatu algoritma yang didominasi oleh solusi dari algoritma lain. Misalnya ada dua himpunan solusi pareto yaitu A dan B. Nilai coverage A adalah hasil pembagian antara jumlah solusi A yang didominasi oleh solusi B dengan jumlah total solusi A. Solusi B dikatakan mendominasi A ketika semua nilai fungsi tujuan B lebih kecil daripada A untuk permasalahan minimasi. Pada permasalahan maksimasi, solusi B mendominasi A ketika semua nilai fungsi tujuan B lebih besar daripada A. Nilai coverage adalah antara 0 sampai dengan 1, namun bukan berarti total perjumlahan dari nilai coverage A dan B adalah 1. Kedua nilai tersebut tidak dapat dijumlahkan. Semakin kecil akan menunjukkan performa algoritma yang semakin baik.

Indikator utama dan paling banyak digunakan pada penelitian objektif jamak adalah hypervolume. Indikator ini mengukur volum dari solusi yang didominasi oleh suatu himpunan solusi dengan mengacu pada titik acuan (*reference point*). Titik acuan ditentukan berdasarkan fungsi tujuan yang digunakan. Apabila untuk permasalahan minimasi, titik acuannya adalah titik yang nilai koordinatnya lebih besar dari nilai titik koordinat solusi yang paling besar. Sedangkan berbanding

terbalik untuk permasalahan maksimasi, nilai titik acuan adalah koordinat yang paling kecil daripada nilai koordinat solusi yang paling minimal. Dimensi volume ditentukan berdasarkan jumlah fungsi tujuan yang digunakan. Gambar 2.9 menunjukkan area yang diukur pada indikator hypervolume. Pada Gambar 2.9 merupakan contoh hypervolume untuk permasalahan minimasi dengan dua fungsi tujuan. Semakin besar nilai hypervolume maka performanya akan semakin baik.



Gambar 2. 9 Grafik penjelasan hypervolume

2.2 Kajian Penelitian Terdahulu

Pada bagian ini akan diuraikan mengenai penelitian-penelitian yang telah dilakukan sebelumnya. Penelitian terdahulu yang menjadi dasar dan acuan penelitian ini adalah penelitian yang berkaitan dengan *Vehicle Routing Problem* (VRP) dan metode yang digunakan untuk menyelesaikan multi objektif VRP. Selain itu juga diuraikan tentang perkembangan penggunaan metode Hiperheuristik untuk permasalahan objektif jamak.

2.2.1 Penelitian Terdahulu Objektif Jamak VRP

VRP merupakan domain permasalahan yang telah banyak diteliti dan dikembangkan dari TSP. Penelitian terkait VRP berkembang dari permasalahan dengan objektif tunggal sampai saat ini semakin mengarah kepada permasalahan dengan multi objektif. Hal ini karena permasalahan tersebut terbukti tidak dapat

merepresentasikan kehidupan nyata pada dunia industri sehingga solusi yang dihasilkan dari permasalahan ini sulit untuk diimplementasikan secara nyata. Pada kenyataannya kebutuhan dunia industri terhadap solusi permasalahan VRP semakin besar dan semakin mengarah pada tujuan yang multi objektif (Jozefowiez et al., 2008). Pada permasalahan VRP objektif utama yang menjadi fokusan sebagian besar peneliti adalah meminimalkan jarak tempuh kendaraan. Namun tren penelitian semakin mengarah pada objektif dengan mempertimbangkan aspek keseimbangan beban kerja (Borgulya, 2008).

Suatu permasalahan dapat dikatakan objektif jamak tidak hanya memiliki objektif yang lebih dari satu namun yang terpenting adalah antar objektifnya harus saling berkonflik. Apabila objektifnya saling berkonflik maka untuk menghasilkan solusi yang paling optimal pada suatu objektif harus menurunkan kualitas solusi terhadap objektif yang lainnya. Oleh karena itu ciri khas dari permasalahan objektif jamak adalah menghasilkan solusi yang pareto (Jozefowiez et al., 2007). Namun pada literatur tidak semua penelitian yang mengklaim objektif jamak adalah benar-benar objektif jamak. Beberapa penelitian tidak menghasilkan solusi pareto dan hanya menggunakan metode agregasi. Permasalahan yang dapat diselesaikan dengan metode agregasi menunjukkan bahwa antar objektifnya selaras dan tidak berkonflik. Beberapa objektif tersebut dioptimasi dan dimasukkan pada satu fungsi menggunakan kombinasi operasi matematika (Baños et al., 2013d). Kelemahan metode ini adalah sulitnya menentukan bobot untuk masing-masing objektifnya, terutama jika informasi terkait implementasi nyata VRP.

Objektif pada penelitian VRP sangat bervariasi. Jozefowiez et al., (2008) mengklasifikasikan berbagai objektif VRP berdasarkan komponen permasalahan yang terhubung menjadi beberapa kategori yaitu rute, sumber daya, dan aktivitas simpul. Berdasarkan klasifikasi tersebut diketahui bahwa dari kategori rute, objektif yang paling banyak digunakan adalah meminimalkan jarak atau waktu tempuh kendaraan karena objektif ini berhubungan langsung dengan biaya. Dari sisi sumberdaya sebagian besar penelitian bertujuan meminimalkan jumlah kendaraan, sedangkan dari sisi simpul penelitian lebih berfokus pada meminimalkan waktu tunggu untuk VRPTW. Penelitian yang dilakukan oleh (Qi et al., 2015)

meminimalkan jarak dan jumlah kendaraan secara bersamaan. Variasi lain penelitian objektif jamak VRP adalah meminimalkan jumlah kendaraan dan total jarak (Chiang and Hsu, 2014). Penelitian yang paling banyak menggunakan objektif adalah Wang et al., (2016). Pada penelitian tersebut menyelesaikan VRPTW dengan lima macam objektif yaitu meminimalkan jumlah kendaraan, total jarak, makespan, waktu tunggu dan waktu tunda. Makespan dalam konteks VRP diartikan sebagai selisih waktu antara kendaraan dari depot dan waktu kendaraan sampai ke depot kembali.

Kebutuhan dunia nyata dalam optimasi VRP, tidak hanya terbatas pada tujuan meminimalkan biaya tetapi lebih jauh perlu mempertimbangkan kesejahteraan pengendara kendaraan dan beban kerja kendaraan. Kesejahteraan pengendara sangat berpegaruh terhadap performa perusahaan dalam memberikan pelayanan terhadap pelanggan (Jozefowiez et al., 2008). Oleh karena itu aspek keseimbangan juga menjadi pertimbangan pada penelitian ini. Lee and Ueng, (1999) mengusulkan pengembangkan VRP dengan mempertimbangkan keseimbangan panjang rute antar kendaraan. Keseimbangan ini bertujuan untuk meningkatkan keadilan solusi yang dihasilkan bagi beban kerja sopir. Hal ini karena dibuktikan bahwa beban kerja akan mempengaruhi insentif yang diperoleh sopir dan juga mempengaruhi kesejahteraan para sopir. Pasia et al., (2007) menyelesaikan VRP dengan meminimalkan jarak dan keseimbangan rute. Pada studi kasus Terenife, Melián-Batista et al., (2014) menggunakan objektif meminimalkan total jarak dan menyeimbangkan beban kerja sopir. Penelitian tersebut juga mempertimbangkan aspek waktu dalam menentukan solusinya karena digunakan VRPTW sehingga keseimbangan beban kerja yang diteliti pada penelitian ini berasal dari sudut pandang waktu. Keseimbangan diukur dari selisih antara total waktu maksimal dan total waktu minimal pada masing-masing kendaraan. Penelitian lainnya (Baños et al., 2013d, 2013b) juga memperhatikan aspek keseimbangan, namun keseimbangan dilihat dari perspektif jarak dan beban yang diangkut oleh masing-masing kendaraan.

Berdasarkan hasil kajian terhadap penelitian terdahulu, maka dapat diketahui bahwa pentingnya mempertimbangkan aspek keseimbangan dalam

menentukan solusi VRP. Implementasi dalam dunia nyata tidak hanya mempertimbangkan sudut pandang optimasi biaya, namun juga perlu memperhatikan kesejahteraan pengendara dan keseimbangan beban kerja kendaraan. Oleh karena itu, pada penelitian ini digunakan dua objektif yang dapat memenuhi aspek optimasi biaya dan aspek keseimbangan beban kerja yaitu meminimalkan jarak tempuh kendaraan dan menyeimbangkan jarak tempuh antar kendaraan.

2.2.2 Penelitian Terdahulu Metode Heuristik

Metode heuristik bukan metode baru dalam permasalahan optimasi. Perkembangannya telah sampai pada level hiperheuristik karena kebutuhan solusi dunia nyata yang bertambah kompleks dan membutuhkan metode yang semakin canggih dan bersifat general. Berdasarkan penelitian sebelumnya telah banyak penelitian yang menggunakan metode hiperheuristik untuk menyelesaikan permasalahan optimasi. Namun permasalahan yang telah mampu diselesaikan oleh hiperheuristik adalah permasalahan optimasi yang memiliki objektif tunggal, sedangkan permasalahan objektif jamak yang telah diselesaikan dengan objektif jamak hiperheuristik adalah cenderung pada permasalahan yang bersifat kontinu. Padahal permasalahan di dunia nyata juga tidak sedikit yang bersifat diskrit dan kombinatorik. Selanjutnya akan dijelaskan tentang penelitian-penelitian yang telah dilakukan untuk menyelesaikan objektif jamak VRP.

Perkembangan penelitian objektif jamak VRP sebagian besar diselesaikan dengan menggunakan metode metaheuristik seperti yang ditunjukkan pada Tabel 2.1. Penelitian yang dilakukan oleh Baños et al., (2013d) menyelesaikan permasalahan permasalahan objektif jamak VRP dengan menggunakan algoritma *Multi Start Multiobjective Evolutionary Algorithm with Simulated Annealing* (MMOEASA). Metode tersebut merupakan hasil modifikasi dan penggabungan antara MOEA dan Simulated Annealing. Sebelumnya peneliti tersebut (Baños et al., 2013b) juga melakukan penelitian untuk menyelesaikan permasalahan objektif jamak VRP namun menggunakan metode yang digunakan berbeda. Pada penelitian tersebut menggunakan metode *Simulated Annealing* sebagai pengontrol dalam mengembangkan inisial solusi yang telah diciptakan pada awal proses. Penelitian

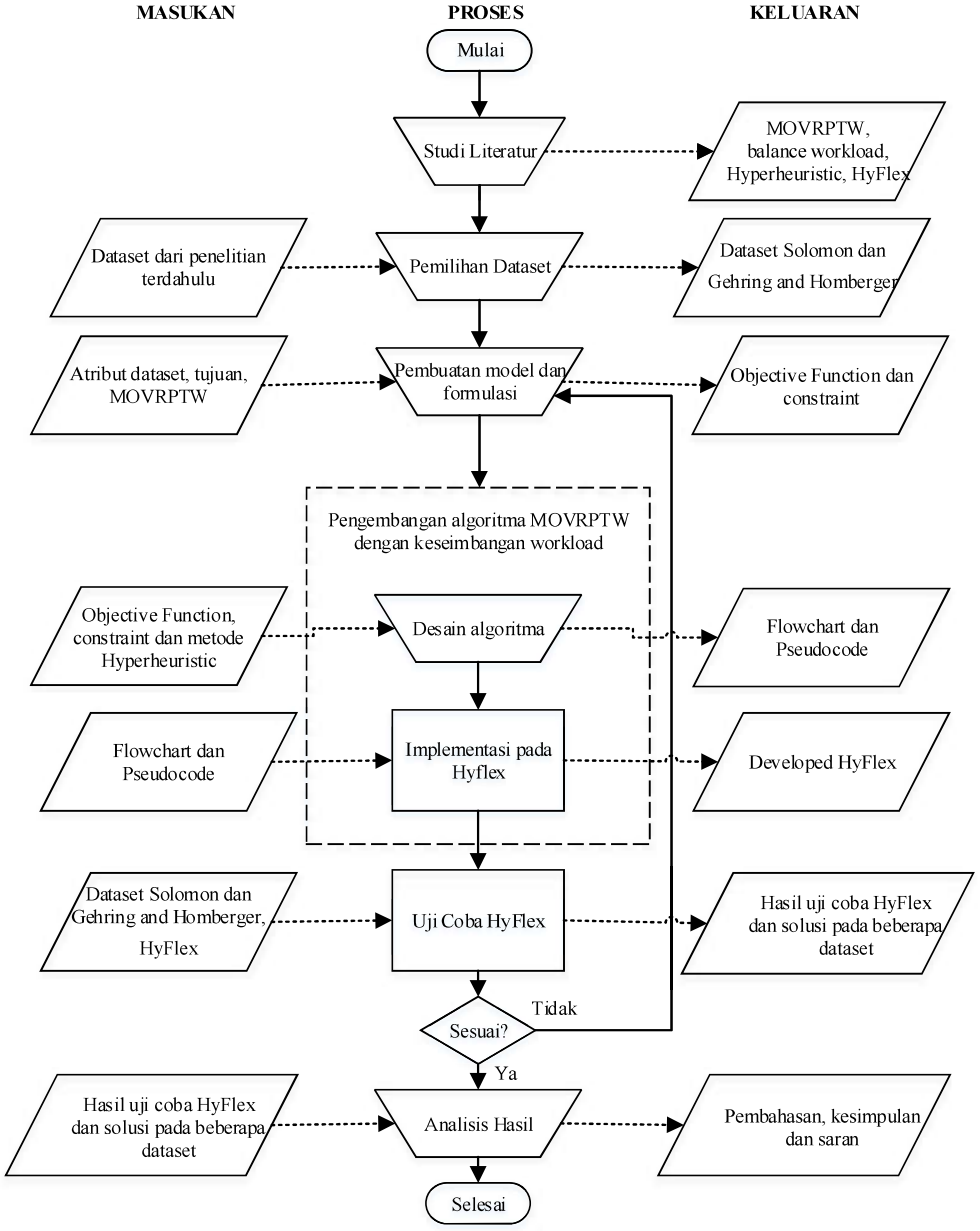
ini juga mengusulkan metode paralelisasi menggunakan *island model* untuk menghasilakan solusi yang tidak hanya lebih berkualitas baik namun juga dihasilkan dengan jumlah yang lebih banyak. Kedua metode yang diusulkan penelitian sebelumnya masih bersifat metaheuristik yang artinya masih sangat tergantung pada pengaturan parameter dan hanya dapat menyelesaikan permasalahan yang spesifik.

Berdasarkan penelitian terdahulu (Baños et al., 2013d) telah terbukti bahwa MOEA merupakan metode yang unggul untuk menyelesaikan permasalahan objektif jamak. Namun selama ini MOEA standar yang telah ada pada penelitian sebelumnya hanya dapat digunakan untuk menyelesaikan permasalahan yang kontinu, contohnya optimasi tata letak ladang pertanian (Li et al., 2017). Pada beberapa penelitian objektif jamak untuk permasalahan kontinu, MOEA digunakan sebagai LLH yang mampu menghasilkan sekumpulan solusi pareto. Apabila permasalahannya memiliki batasan yang sangat banyak dan ketat maka tidak dapat digunakan MOEA standar karena kemungkinan akan menghasilkan solusi yang tidak layak. Penelitian objektif jamak Hiperheuristik sebelumnya (Li et al., 2018, 2017) menggunakan MOEA standar seperti NSGA II dan SPEA sebagai LLHnya. Penelitian tersebut berfokus pada pengembangan metode pencarian LLH dan metode penerimaan solusi yang dihasilkan oleh LLH. Oleh karena itu perlu dikembangkan metode objektif jamak Hiperheuristik yang mampu menyelesaikan permasalahan objektif jamak yang kombinatorik seperti VRP. Strategi baru perlu dirumuskan untuk menghasilkan solusi yang tidak terdominasi oleh solusi yang lain tanpa menggunakan MOEA standar.

(Halaman ini sengaja dikosongkan)

BAB III
METODOLOGI PENELITIAN

Bab ini berisi uraian dan penjelasan terkait tahapan-tahapan sistematis dalam mengerjakan penelitian ini. Pada bagian ini juga dijelaskan mengenai input dan output dari masing-masing tahapan penelitian. Metodologi penelitian ditunjukkan pada Gambar 3.1.



Gambar 3. 1 Metodologi Penelitian

3.1.Studi Literatur

Studi literatur merupakan tahapan paling awal dalam melakukan penelitian. Tahapan ini dilakukan untuk mengkaji penelitian-penelitian terkait *Vehicle Routing Problem* (VRP) dan metode heuristik yang telah dilakukan sebelumnya sehingga dapat diketahui perkembangan penelitian pada permasalahan tersebut sebagai bahan untuk analisis GAP penelitian dan potensi penelitian selanjutnya. Pengkajian terhadap penelitian terdahulu juga dilakukan untuk mencari referensi terkait pengembangan metode dan analisis terhadap kekurangan dan kelebihan metode yang telah ada sebelumnya. Berdasarkan hasil pengkajian tersebut maka pada penelitian ini diusulkan penelitian terkait pengembangan objektif jamak VRPTW dengan mempertimbangkan aspek keseimbangan yang diterapkan pada kerangka kerja HyFlex. Aspek keseimbangan yang diusulkan dilihat dari panjang rute yang ditempuh oleh kendaraan.

Selain pengkajian terhadap penelitian sebelumnya, juga dilakukan kajian terhadap konsep-konsep teori dasar yang diperlukan dalam mengerjakan penelitian. Konsep teori yang perlu dipahami antara lain tentang *Vehicle Routing Problem* (VRP), Multi-objective VRP, Hiperheuristik dan kerangka kerja HyFlex. Teori dasar diperlukan untuk acuan dalam mengembangkan model dan pembuatan formulasi multi-objective VRPTW. Hasil studi literatur dijabarkan pada Bab 2 Dasar Teori dan Kajian Pustaka.

3.2.Pemilihan Dataset dan Formulasi Objektif Jamak VRP

Pada bagian ini akan dijelaskan mengenai tahapan pemilihan data dan pemodelan atau formulasi permasalahan VRP yang digunakan pada penelitian ini. Tahapan ini merupakan tahapan yang penting yang dilakukan pada penyelesaian permasalahan optimasi.

Pemilihan dataset VRP dilakukan berdasarkan kompleksitas dataset karena pada penelitian ini digunakan metode Hiperheuristik yang memiliki generalitas dalam menyelesaikan domain permasalahan. Pada penelitian ini dipilih dua macam dataset benchmark VRP yang umum digunakan pada penelitian-penelitian sebelumnya. Kriteria dataset yang dipilih untuk penelitian ini adalah:

a. Dataset umum digunakan oleh penelitian sebelumnya untuk memastikan kelayakan dataset tersebut jika digunakan untuk penelitian.
b. Antar dataset tersebut memiliki kompleksitas yang berbeda. Terdiri dari dataset yang memiliki kompleksitas yang rendah dengan jumlah simpul yang relatif sedikit dan dataset yang memiliki kompleksitas tinggi dengan jumlah simpul yang relatif banyak.
c. Dataset memiliki kategori kerapatan time windows yang berbeda-beda. Hal ini juga mempengaruhi kompleksitas dataset.
d. Dataset terdiri dari atribu-atribut yaitu ID, titik koordinat x dan y, time windows, maksimum jumlah kendaraan dan maksimum kuantitas dari masing-masing kendaraan. Data time windows sangat diperlukan karena penelitian ini menggunakan VRPTW.
e. Dataset dapat diinputkan ke dalam kode program Java karena penelitian ini menggunakan kerangka kerja HyFlex yang menggunakan Bahasa Java.

Tahapan selanjutnya dalam penelitian adalah membuat model matematis dan formula yang disesuaikan dengan multi objective yang digunakan pada VRPTW. Model matematis pada permasalahan riset operasi terdiri dari dua jenis yaitu fungsi tujuan dan batasan *(constraint)*. Model matematis dibuat dengan mengembangkan dari model matematis dasar pada VRP (Chiang and Hsu, 2014; Lee and Ueng, 1999; Wang et al., 2016). Fungsi tujuan yang berkaitan dengan keseimbangan jarak rute dibuat berdasarkan rumus simpangan baku yang disesuaikan dengan formulasi optimasi.

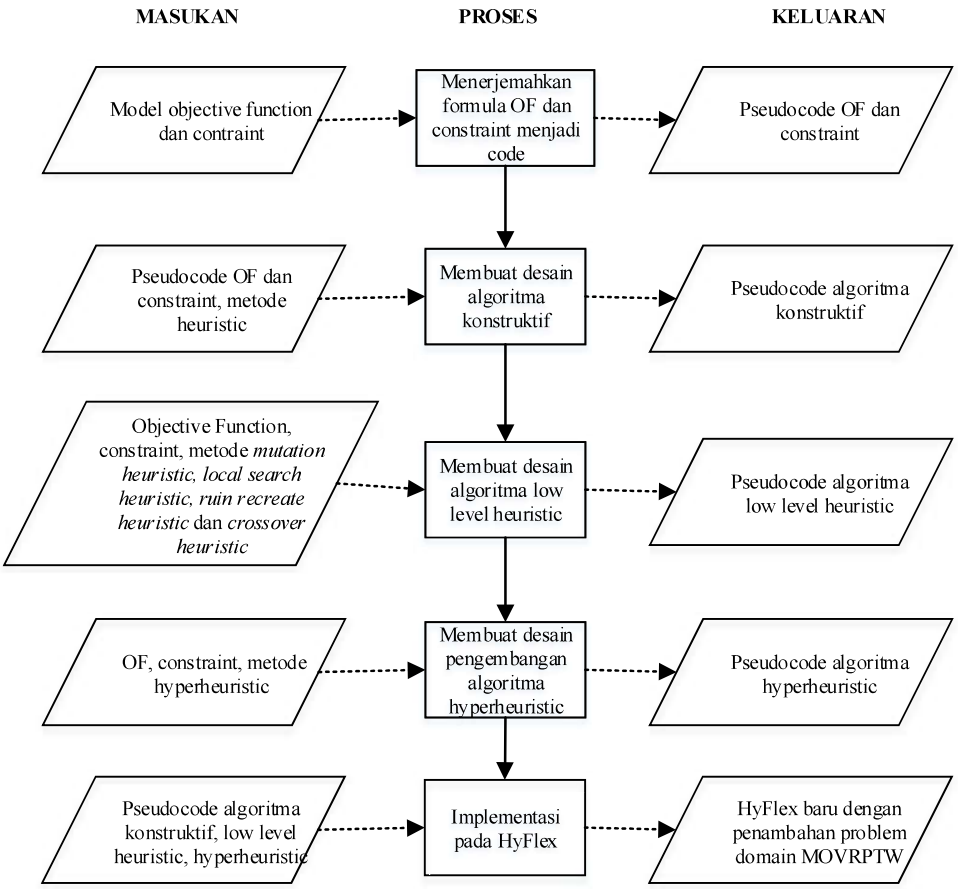
3.3.Pengembangan algoritma Hiperheuristik Objektif Jamak VRPTW

Setelah merancang model dan formulasi dari objektif jamak VRPTW, tahapan selanjutnya adalah merancang dan mendesain algoritma untuk memudahkan dalam proses implementasi pada HyFlex.

3.3.1. Desain Algoritma

Desain algoritma disesuaikan dengan struktur kerangka kerja yang digunakan. Pada penelitian ini digunakan kerangka kerja HyFlex yang merupakan aplikasi untuk menemukan solusi dari permasalahan optimasi yang diselesaikan

dengan menggunakan metode Hiperheuristik. Pembuatan desain algoritma dilakukan dengan beberapa tahapan yang ditunjukkan pada Gambar 3.3.



Gambar 3. 2 Alur pembuatan desain algoritma

Tahapan desain yang pertama adalah menerjemahkan model dan formula yang telah dibuat pada Subbab 3.2 menjadi *pseudocode*. Hal ini termasuk dalam tahapan desain domain permasalahan. Formula *objective* maupun *constraint* dijabarkan dalam beberapa *method* dalam pengkodean.

Selanjutnya adalah membuat pseudocode untuk *constructive* algorithm. Construstive algorithm merupakan algoritma yang digunakan untuk membentuk *initial solution* yang *feasible*. Tolok ukur suatu solusi *feasible* atau tidak dilihat dari keterpenuhannya terhadap *constraint* yang telah ditentukan pada Subbab 3.2. Tipe *constraint* yang harus dipenuhi pada initial solution adalah *hard constraint*.

Tahapan ketiga adalah menentukan *low level* heuristik yang akan digunakan pada *problem domain*. *Low level* heuristik merupakan algoritma yang digunakan untuk memodifikasi *initial solution* menjadi beberapa solusi yang optimal. Tingkat optimalitas didasarkan pada nilai atau skor fungsi objective yang dicapai sebuah solusi. Pada penelitian ini digunakan empat kategori *low level* heuristik, yaitu *mutation* heuristik*, local search* heuristik*, ruin recreate* heuristik dan *crossover* heuristik.

Setelah menentukan *low level* yang digunakan maka tahapan selanjutnya adalah membuat pesudocode dari masing-masing *low level* heuristik. Solusi baru yang dihasilkan oleh *low level* heuristik harus tetap memenuhi *constraint* dan objective function agar tidak merusak kelayakan dari solusi tersebut. Tugas dari *low level* heuristik adalah memodifikasi initial solution agar menjadi lebih optimal tanpa menghilangkan kelayakan solusi tersebut. *Mutation* heuristik adalah memodifikasi dengan hanya membuat perturbasi kecil pada solusi, seperti swap atau insert. *Local search* heuristik adalah merupakan pengembangan pertama dari algoritma hill climbing. *Ruin recreate* heuristik merusak atau memutasi sebagian atau semua bagian dari sebuah solusi sebelum merekonstruksi ulang solusi tersebut, pada ketegori ini dapat dilakukan penggabungan dengan metode pada kategori *low level* heuristik yang lainnya seperti mutation. Sedangkan *crossover* heuristik adalah solusi yang mengkombinasikan dua buah solusi menjadi solusi yang baru.

Tahapan terakhir pada desain adalah melakukan pengembangan metode pemilihan *low level* heuristik (Hiperheuristik). Pada kerangka kerja HyFlex, telah ada beberapa metode Hiperheuristik namun algoritma-algoritma tersebut didesain untuk menyelesaikan permasalahan single objective sedangkan pada penelitian ini menggunakan problem domain yang bersifat objektif jamak. Oleh karena itu, perlu dikembangkan metode Hiperheuristik untuk permasalahan objektif jamak. Pada tahap akhir ini dibuat desain dan pseudocode dari algoritma pemilihan *low level* heuristik yang ingin diusulkan pada penelitian ini.

3.3.2. Metode Objektif Jamak Hiperheuristik

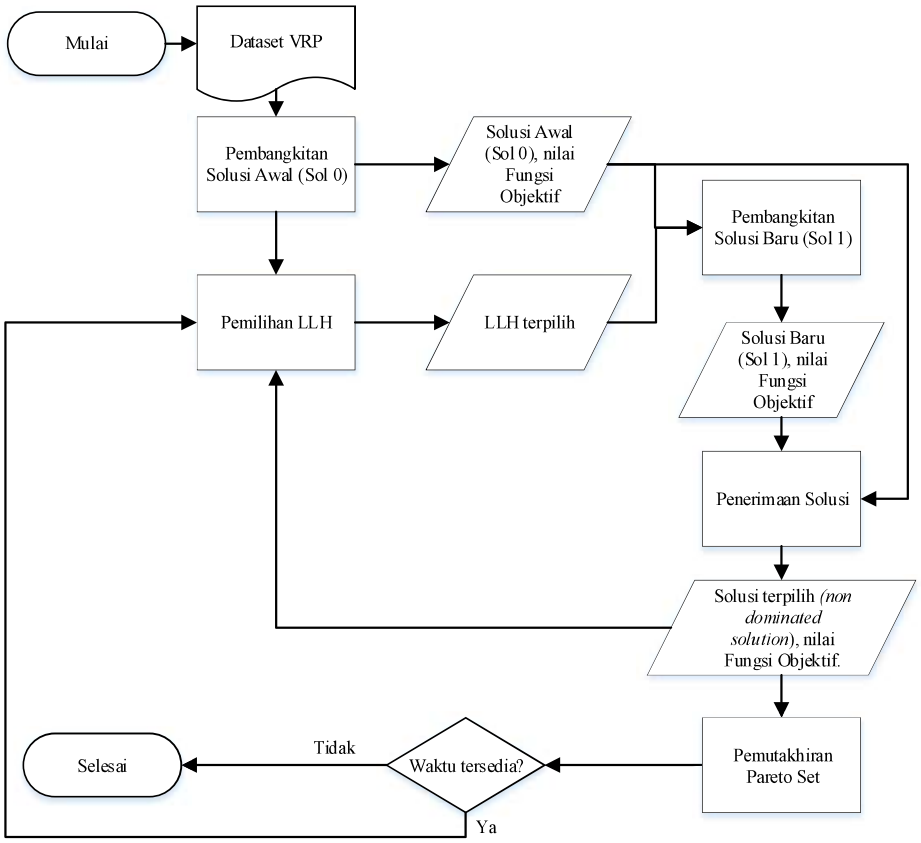
Pada Subbab 2.1.4 telah dijelaskan mengenai konsep metode Hiperheuristik. Secara umum Hiperheuristik memiliki dua tahapan yaitu :

1) Pemilihan *low level* heuristik (LLH)
2) Penentuan perimaan atau penolakan solusi yang dihasilkan oleh LLH *(move acceptance)*

Pada Gambar 3.4 ditunjukkan metode Hiperheuristik yang diusulkan pada penelitian untuk menyelesaikan permasalahan VRP yang objektif jamak.

Pada awal prosesnya dilakukan pembuatan inisial solusi yang *feasible*, sehingga digunaka metode contructive. Metode ini dilakukan dengan membangun solusi awal dari nol, dikembangkan dari bagian ke bagian sehingga akan menjadi kesatuan solusi utuh. Strategi yang digunakan untuk membangun inisial solusi adalah :

1) Berawal dari depot, kemudian memilih pelanggan yang memiliki start time paling awal sejumlah kendaraan yang digunakan karena jumlah rute sama dengan jumlah kendaraan.
2) Selanjutnya dipilih pelanggan yang paling mendekati dengan pelanggan yang sebelumnya dan memiliki time windows yang dimulai setelahnya.
3) Langkah 1 dan 2 diulang sampai semua pelanggan telah terpetakan pada kendaraan.
4) Setelah dihasilkan rute maka langkah selanjutnya adalah menghitung total jarak, total waktu dan total beban yang harus diangkut pada masing kendaraan.
5) Dilakukan penghitungan tingkat keseimbangan dengan menggunakan simpangan baku sesuai yang telah dijelaskan pada Subbab 2.1.1 Multionbjective VRP.
6) Langkah 1 sampai 6 dilakukan berulang sampai dihasilkan inisial solusi yang *feasible* berdasarkan *constraint* dan fungsi tujuan pada Subbab 3.2.2 Pembuatan Model dan Formulasi.



Gambar 3. 3 Metode Hiperheuristik yang diusulkan

Penelitian ini mengusulkan pemberian bobot secara random pada setiap iterasi yang dilakukan, maka tahapan selanjutnya adalah menentukan bobot untuk masing-masing fungsi tujuan secara random. Konsep pembobotan seperti yang dijelaskan pada Subbab 2.1.3 bagian metode agregation. Pembobotan ini hanya digunakan sementara dan akan berubah pada setiap iterasi. Total bobot pada semua fungsi tujuan adalah 100% atau 1. Formula fungsi objective yang menggunakan metode agregasi adalah sebagai berikut :

$$\sum_{i=1}^{n} w_i OF_i = w_1 OF_1 + w_2 OF_2 + \ldots + w_n OF_n \tag{3.19}$$

Tujuan dari strategi ini adalah untuk menghasilkan kemungkinan solusi pareto yang bervariasi. Hal ini untuk memenuhi kriteria diversifikasi pada metode Hiperheuristik yang baik seperti telah dijelaskan pada Subbab 2.1.4. Metode Hiperheuristik harus menyeimbangkan antara intensifikasi dan diversifikasi.

Sedangkan kriteria intensifikasi dipenuhi dengan menerima solusi yang lebih baik pada metode move acceptance.

Tahapan yang ketiga adalah memilih *low level* heuristik yang akan menghasilkan solusi pada setiap iterasi. Pada penelitian ini digunakan *low level* heuristik untuk domain permasalahan VRP yang telah digunakan pada HyFlex, yaitu mutation heuristik, crossover heuristik, hill climbing/local seach heuristik dan ruin-recreate heuristik. Penjelasan secara detail mengenai masing-masing *low level* heuristik terdapat pada (Walker, 2015a). Metode lain yang juga digunakan pada penelitian ini adalah *Fixed Sequence* dan *Random Choice* (Li et al., 2017). Penjelasan terkait masing-masing metode pemilihan *low level* heuristik terdapat pada Subbab 2.1.4 tentang Hiperheuristik.

Tahapan selanjutnya adalah mengeksekusi *low level* heuristik yang telah terpilih pada tahapan sebelumnya untuk menghasilkan solusi baru yang dikembangkan dari inisial solusi. Tahapan ini merupakan tahapan yang terjadi pada bagian *low level* karena hanya berfokus pada menghasilkan dan mengembangan solusi baru. Langkah-langkah dalam menghasilkan solusi tergantung pada *low level* heuristik yang terpilih.

Setelah dihasilkan solusi, maka tahapan selanjutnya adalah menentukan apakah solusi baru yang dihasilkan oleh LLH diterima atau ditolak. Tahapan ini disebut dengan tahapan *move acceptance*, artinya apakah solusi berpindah dari inisial solusi ke solusi baru yang dihasilkan. Solusi baru akan diterima apabila solusi tersebut memenuhi rumus fungsi objektif dengan pembobotan yang telah ditentukan secara random, artinya apabila solusi baru memiliki nilai fungsi yang lebih besar daripada nilai fungsi yang dimiliki oleh current solution maka solusi baru tersebut akan diterima dan dimasukkan ke dalam Pareto set. Metode lain yang digunakan untuk membandingkan solusi yang digunakan adalah memilih semua solusi, *Great Deluge Acceptance* (GDA) dengan metrik D dan memilih solusi yang terbaik atau *Hill Climbing* (HC)(Li et al., 2017).

Langkah kedua sampai dengan keenam akan di ulangi hingga batas waktu yang telah ditentukan di tahapan awal berakhir. Setelah proses iterasi selesai dan

mencapai terminasi, maka tahapan selanjutnya dilakukan pemilihan pareto front. suatu solusi disebut pareto front apabila solusi tersebut minimal lebih baik pada salah satu fungsi tujuan, namun tidak lebih jelek untuk fungsi tujuan yang lain jika dibandingkan dengan solusi yang lainnya. Pada tahapan ini dilakukan pemilihan solusi yang tidak didominasi oleh solusi yang lain dengan mengurutkan nilai fungsi pada masing-masing solusi. Selanjutnya dibuat ilustrasi dengan grafik pareto optimal sehingga akan dihasilkan sekumpulan solusi yang paling optimal diantara solusi-solusi yang lain pada pareto set.

3.3.3. Implementasi pada HyFlex

Desain yang telah dibuat pada tahapan sebelumnya akan diimplementasikan pada kerangka kerja HyFlex. Implementasi dilakukan dengan menggunakan bahasa pemrograman java dan dengan aplikasi netbeans. Kerangka kerja ini memiliki tiga class yang terdiri dari class problem domain, class heuristik dan class Hiperheuristik. Diagram class ditunjukkan pada Gambar 2.4.

Fokusan utama penelitian ini adalah pada class problem domain. Hasil dari tahapan pertama dan kedua desain algoritma diimplementasikan pada class problem domain. Class heuristik type dimodifikasi dengan hasil dari tahapan ketiga dan keempat dari tahapan desain algoritma sedangkan hasil dari pengembangan metode Hiperheuristik dimasukkan ke dalam class Hiperheuristik.

3.4. Uji Coba

Apabila implementasi pada kerangka kerja HyFlex telah selesai dilakukan, maka tahapan selanjutnya adalah melakukan uji coba. Tahapan ini dilakukan untuk memastikan bahwa HyFlex dapat dijalankan sesuai fungsinya. Selain itu juga untuk menguji performa algoritma yang digunakan, maka dilakukan uji coba. Tahapan uji coba terdiri dari :

a. Menentukan lingkungan uji coba, yaitu peralatan-peralatan yang digunakan pada saat uji coba, seperti spesifikasi komputer yang digunakan dan aplikasi pendukung yang menjadi media dalam melakukan uji coba.
b. Menentukan dan mengatur parameter, yakni batasan waktu untuk menjalankan algoritma dan nilai minimum value dan nilai maximum value dari masing-

masing fungsi objektif sesuai dengan hasil uji coba awal *(preliminary eksperiment)*.

c. Membuat skenario uji coba. Pada penelitian ini terdiri dari dua uji coba yaitu :
    1) Melakukan uji coba awal untuk menentukan nilai parameter pada tahapan normalisasi yaitu nilai maksimum dan minimum dari masing-masing fungsi objektif.
    2) Melakukan uji coba untuk membandingkan performa dari algoritma Hiperheuristik yang diusulkan pada penelitian ini. Selain itu juga untuk mencari kombinasi antara metode pemilihan LLH dan metode penerimaan solusi yang dapat menghasilkan solusi paling optimal.

Kombinasi metode Hiperheuristik yang digunakan untuk uji coba pada penelitian ini, antara lain :

a. Metode *Simple Random* dengan *Great Deluge (GD)*
b. Metode *Simple Random* dengan *Hill Climbing (HC)*

Indikator performa yang digunakan untuk membandingkan antar metode adalah jumlah solusi yang tidak terdominasi, coverage dan hypervolume. Pengukuran performa tersebut seperti dengan indikator yang digunakan pada metode *choice function*. Penjelasan lebih detail untuk masing-masing indikator performa terdapat pada Subbab 2.1.6.

Pada penelitian ini terdapat tahapan Verifikasi dan Validasi. Verifikasi dilakukan dengan membandingkan antara perhitungan manual dan output dari hasil implementasi kode program. Verifikasi memastikan bahwa hasil perhitungan fungsi objektif telah sesuai dan tepat. Sedangkan tahapan Validasi dilakukan dengan membandingkan antara solusi yang dihasilkan oleh single objektif VRP dengan solusi dari hiperheuristik multiobjetif VRP. Tujuan dilakukannya validasi adalah untuk membuktikan bahwa solusi yang dihasilkan oleh hiperheuristik multibjektif memiliki rute-rute yang lebih seimbang dibandingkan dengan hiperheuritik untuk single objektif.

Pada penelitian ini dilakukan dua macam eksperiment yaitu uji coba awal (preliminary experiment), dilakukan untuk menentukan nilai minimal dan nilai maksimal dari nilai fungsi tujuan di setiap instance. Uji coba ini untuk mendukung langkah normalisasi. Setiap instance dijalankan 25 kali untuk memberikan statistik yang andal, kemudian hasil dari setiap run dibandingkan untuk menemukan nilai minimal dan maksimal sebagai parameter normalisasi dalam setiap instance. Uji coba kedua adalah performa algoritma, untuk menguji algoritma yang diusulkan dan membandingkan dua algoritma penerimaan bergerak (HC dan GD). Sama seperti percobaan awal, keduanya memiliki waktu berjalan atau batas waktu 20, 40 dan 80 detik. Setiap waktu berjalan akan dijalankan 25 kali untuk masing-masing algoritma.

3.5. Analisis Hasil

Tahapan terakhir pada penelitian ini adalah melakukan analisis terhadap hasil dari penelitian dan hasil uji coba. Analisis dilakukan terhadap performa algoritma yang dihasilkan yaitu berdasarka jumlah solusi pareto dan nilai coverage. Nilai coverage direpresentasikan ke dalam boxplot, swarm plot dan histogram untuk memudahkan dalam menganalisis dan membandingkan antara algoritma HC dan GD. Hasil dari tahapan ini dijabarkan pada Bab Uji Coba dan Analisis Hasil. Selanjutnya dirangkum menjadi kesimpulan dan dirumuskan saran penelitian selanjutnya apabila ada yang perlu diperbaiki dari hasil penelitian ini.

(Halaman ini sengaja dikosongkan)

BAB IV
DESAIN DAN IMPLEMENTASI

Pada Bab ini akan dijelaskan mengenai dataset, formulasi, desain dan implementasi algoritma yang diusulkan pada penelitian ini. Bab ini terdiri dari hasil pemilihan dataset, formulasi Objektif jamak VRP, desain algoritma untuk Objektif jamak Hiperheuristik dan implementasinya pada HyFlex. Desain disajikan dalam bentuk pseudocode dan Flowchart.

4.1 Dataset Penelitian dan Formulasi Objektif Jamak VRP

Bagian ini terdiri dari penjelasan secara detail mengenai dataset yang digunakan pada penelitian. Selain itu juga diuraikan formula domain permasalahan yang digunakan dalam mengoptimasi Objektif jamak VRP.

4.1.1 Dataset Penelitian

Pada penelitian ini digunakan dua macam dataset yang biasa digunakan sebagai patokan *(benchmark)* penelitian-penelitian sebelumnya dalam mengusulkan algoritma atau pengembangan problem domain *Vehicle Routing Problem*, khususnya VRP dengan mempertimbangkan *time window.* Kedua dataset tersebut adalah dataset *Solomon* dan dataset *Gehring and Homberger* (Baños et al., 2013a; Chiang and Hsu, 2014; Walker, 2015a). Kedua dataset ini memiliki kompleksitas yang berbeda. Contoh *instance* dataset yang digunakan pada penelitian ini ditunjukkan pada Gambar 4.1 dan diketahui bahwa dataset terdiri dari beberapa atribut, sebagai berikut:

1. **Nama Instance**, yaitu kode nama dataset.
2. **Batasan dari kendaraan**, terdiri dari **jumlah maksimal kendaraan** yang dapat digunakan dan kapasitas maksimal masing-masing kendaraan.
3. **Data pelanggan (*pelanggan*)**, terdiri dari 7 sub atribut yaitu :
    1) Nomer atau ID pelanggan. Khusus untuk nomer *pelanggan* 0 adalah depot.
    2) Koordinat X, lokasi masing-masing *pelanggan* dan depot pada titik koordinat X.
    3) Koordinat Y, lokasi masing-masing *pelanggan* dan depot pada titik koordinat Y.

4) Permintaan (*demand*), nilai yang merepresentasikan permintaan atau kapasitas masing-masing *pelanggan* yang harus dikirim oleh kendaraan.
5) *Ready time*, waktu paling awal suatu *pelanggan* mulai mendapatkan pelayanan atau diberikan permintaannya.
6) *Due date*, batas akhir waktu suatu *pelanggan* mulai mendapatkan pelayanan atau diberikan permintaannya.
7) *Service time*, jumlah waktu yang dibutuhkan untuk melayani setiap pelanggan.

C101

VEHICLE

NUMBER	CAPACITY
25	200

CUSTOMER

CUST NO.	XCOORD.	YCOORD.	DEMAND	READY TIME	DUE DATE	SERVICE TIME
0	40	50	0	0	1236	0
1	45	68	10	912	967	90
2	45	70	30	825	870	90
3	42	66	10	65	146	90
4	42	68	10	727	782	90
5	42	65	10	15	67	90
6	40	69	20	621	702	90
7	40	66	20	170	225	90
8	38	68	20	255	324	90
9	38	70	10	534	605	90
10	35	66	10	357	410	90
11	35	69	10	448	505	90
12	25	85	20	652	721	90
13	22	75	30	30	92	90
14	22	85	10	567	620	90
15	20	80	40	384	429	90
16	20	85	40	475	528	90
17	18	75	20	99	148	90
18	15	75	20	179	254	90
19	15	80	10	278	345	90

Gambar 4. 1 Contoh instance dataset

Dataset *Solomon* dan *Gehring and Homberger* terdiri dari 56 *benchmark instance* namun yag membedakan adalah jumlah pelanggan pada masing-masing *instance*. Pada dataset *Solomon* setiap *instance* terdiri dari data 100 *pelanggan*, sedangkan dataset *Gehring and Homberger* terdiri dari 1000 data *pelanggan.* Terdapat tiga kategori *instance* pada dataset ini, yaitu:

a. **Kategori C**, terdiri dari subset pelanggan yang diklusterisasi berdasarkan lokasinya.
b. **Kategori R**, terdiri dari subset pelanggan yang tersebar secara random.
c. **Kategori RC**, terdiri kombinasi antara kedua kategori sebelumnya.

Pada kategori C letak antar pelanggan lebih dekat dibandingkan letak pelanggan pada kategori R. Sehingga dataset kategori R akan memiliki rute yang lebih panjang dibandingkan dengan kategori C. Hal ini menyebabkan penyelesaiannya akan lebih rumit dan sulit karena sedikit perubahan terhadap urutan pelanggan yang dikunjungi akan menyebabkan perubahan yang besar pada total panjang rute. Pada dataset Solomon juga dibagi menjadi kategori 1 (C1, R1, RC1) dan 2 (C2, R2, RC2). Perbedaan dari kedua kategori tersebut adalah kategori 1 memiliki time windows yang lebih kecil dibandingkan dengan kategori 2. Sehingga perubahan urutan kunjungan pelanggan yang dikunjungi pada kategori 1 lebih beresiko menyebabkan solusi yang dihasilkan menjadi tidak layak (*feasible*) dibandingkan pada kategori 2. Selain itu, time windows pada depot kategori 1 juga akan lebih kecil dibanding pada kategori 2 sehingga jumlah pelanggan yang dilayani masing-masing kendaraan pada dataset kategori 1 akan lebih sedikit dibandingkan dengan kategori 2.

Kedua dataset ini digunakan untuk menguji kemampuan algoritma *objektif jamak* VRPTW dalam menghasilkan sekumpulan solusi sesuai dengan batasan dan beberapa fungsi tujuan yang akan ditentukan pada tahapan selanjutnya. Selain itu juga digunakan untuk menguji kemampuan algoritma Hiperheuristik jika digunakan pada beberapa dataset yang berbeda. Kedua dataset ini berbeda terutama pada jumlah customer atau simpul yang harus diselesaikan sehingga kedua dataset tersebut memiliki kompleksitas yang berbeda. Dataset *Gehring and Homberger* lebih kompleks dengan data customer yang lebih banyak dibandingkan dengan dataset *Solomon*.

Pada penelitian ini digunakan masing-masing tiga *instance* dari setiap dataset *Solomon* dan dataset *Gehring and Homberger.* Instance yang digunakan untuk uji coba adalah RC207 (0), R101 (1), C208 (13) dari dataset Solomon.

Sedangkan instance dataset *Gehring and Homberger* adalah C1_10_1 (5), RC2_10_1 (6) dan R1_10_1 (7). Pemilihan instance didasarkan pada kategori pada dataset. Masing-masing kategori R, C dan RC dipilih satu instance. Dari dataset Solomon dipilih dua instance kategori 2 karena agar dapat membandingkan hasil uji coba dari dataset yang sangat sederhana sampai dengan dataset yang paling kompleks seperti pada dataset *Gehring and Homberger.* Hal itu juga menjadi alasan dipilihnya kategori 1 sebanyak 2 instance dari dataset *Gehring and Homberger.* Tujuan yang ingin dicapai dari hal tersebut yaitu untuk mengukur kemampuan dan fleksibilitas algoritma dalam menyelesaikan dataset sederhana dan kompleks.

4.1.2 Pembuatan Model dan Formulasi Objektif Jamak VRPTW

Tahapan selanjutnya dalam penelitian adalah membuat model matematis dan formula yang disesuaikan dengan multi objective yang digunakan pada VRPTW. Model matematis pada permasalahan riset operasi terdiri dari dua jenis yaitu fungsi tujuan dan batasan *(constraint)*. Model matematis dibuat dengan mengembangkan dari model matematis dasar pada VRP (Chiang and Hsu, 2014; Lee and Ueng, 1999; Wang et al., 2016). Pada penelitian ini digunakan dua *objective* dengan formulasi yang ditunjukkan oleh Rumus 4.1 dan 4.2.

a. Meminimalkan jarak yang ditempuh masing-masing kendaraan (rute)

$$Min\sum_{k=1}^{K}\sum_{i-1}^{N}\sum_{j=1}^{N}x_{ij}^{k}C_{ij} \tag{4.1}$$

b. Meminimalkan ketidakseimbangan jarak pada masing-masing rute. Pengukuran keseimbangan digunakan simpangan baku

$$Min\left(\sqrt{\left(\frac{1}{K}\sum_{k=1}^{K}\left(\sum_{i=1}^{N}\sum_{j=2}^{N}x_{ij}^{k}C_{ij}-\left(\frac{\sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=2}^{N}x_{ij}^{k}C_{ij}}{K}\right)\right)^{2}\right)},\left(\forall k\in\{1,...,K\}\right) \tag{4.2}$$

Batasan atau *constraint* yang diperlukan untuk mencapai dua *objective* di atas diformulasikan menjadi persamaan matematika pada Rumus 3.3 sampai dengan 3.18.

a. Jika suatu kendaraan k pergi dari suatu simpul i ke simpul yang lain j maka nilai x_{ij}^{k} adalah 1 namun jika tidak maka nilainya 0.

$$x_{ij}^{k} \in \{0,1\} (\forall i, j \in \{1,...,N\}, \forall k \in \{1,...,K\}) \tag{4.3}$$

$x_{ij}^{k} = \begin{cases} 1, \textit{jika kendaraan} \text{ k pergi dari node i ke node j} \\ 0, \text{kondisi lainnya} \end{cases}$

Perhitungan jarak atau waktu antara depot dan pelanggan maupun antar pelanggan dihitung dengan menggunakan rumus Eucledian Distance :

$$c_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{4.4}$$

b. Rute masing-masing kendaraan harus berawal dan berakhir di satu depot.

$$\sum_{j=2}^{N} x_{1j}^{k} - \sum_{j=2}^{N} x_{j1}^{k} = 0 \left(\forall k \in \{1,...,K\} \right) \tag{4.5}$$

c. Masing-masing pelanggan harus dilayani oleh tepat satu kendaraan.

$$\sum_{k=1}^{K} \sum_{i=1}^{N} x_{ij}^{k} = 1 \left(\forall j \in \{2,...,N\} \right) \tag{4.6}$$

d. Suatu kendaraan harus berjalan dari satu pelanggan ke pelanggan lain yang berbeda, setiap pelanggan hanya dikunjungi sekali.

$$X_{ii}^{k} = 0 \left(\forall i \in \{1,...,N\}, \forall k \in \{1,...,K\} \right) \tag{4.7}$$

e. Masing-masing kendaraan memiliki maksimal 1 rute perjalanan sehingga terdapat sejumlah maksimal K rute yang keluar dari depot pengiriman.

$$\sum_{k=1}^{K} \sum_{j=2}^{N} x_{ij}^{k} \leq K \tag{4.8}$$

f. Tidak boleh ada subtour.

$$\sum_{i \in S} \sum_{j \in S} x_{ij}^{k} \leq |S| - 1 \left(\forall k \in \{1,...,K\}, \forall \in S \{2,...,n\} \right) \tag{4.9}$$

g. Total kapasitas demand yang dibawa oleh masing-masing kendaraan tidak boleh melebihi kapasitas maksimal yang telah ditentukan (Q). Batasan ini belum sepenuhnya diimplementasikan pada penelitian ini.

$$\sum_{i=1}^{N} \sum_{j=2}^{N} x_{ij}^{k} q_j \leq Q \left(\forall k \in \{1,...,K\} \right) \tag{4.10}$$

h. Suatu kendaraan harus datang pada pelanggan j sebelum *time windows* pada pelanggan tersebut berakhir.

$$a_j \le l_j \left(\forall j \in \{2,...,N\} \right) \tag{4.11}$$

i. Suatu kendaraan harus datang dan melayani *pelanggan* j setelah mulainya *time windows* pada *pelanggan* j tersebut. Service suatu kendaraan terhadap suatu *pelanggan* j diberikan pada antara waktu mulai dan waktu akhir *time windows*.

$$b_j = \max\left(a_j, e_j \right) \left(\forall j \in \{2,...,N\} \right) \tag{4.12}$$

$$e_j \le s_{kj} \le l_j \left(\forall j \in \{2,...,N\}, \forall k \in \{1,...,K\} \right) \tag{4.13}$$

j. Apabila kendaraan sampai pada suatu pelanggan j pada waktu sebelum dimulainya waktu time windows (*ready time*) maka akan menyebabkan adanya waktu tunggu (*waiting time*). Sehingga waktu tunggu dihasilkan dari selisih antara *arrival time* dan *ready time*.

$$w_{ij}^{k} \in \{0,1\} \left(\forall i, j \in \{1,...,N\}, \forall k \in \{1,...,K\} \right) \tag{4.14}$$

$$w_{ij}^{k} = \begin{cases} 0, jika\ a_j \ge e_j \\ a_j - e_j, \text{ kondisi lainnya} \end{cases} \tag{4.15}$$

k. Semua kendaraan harus kembali ke depot pada waktu sebelum waktu akhir time windows depot berakhir.

$$a_{j1}^{k} \le l_1 \left(\forall j \in \{2,...,N\}, \forall k \in \{1,...,K\} \right) \tag{4.16}$$

l. Total waktu yang dibutuhkan masing-masing kendaraan adalah hasil penjumlahan total waktu service, total waktu perjalanan dan total waktu tunggu.

$$T_k = \sum_{j=2}^{N} \left(s_{k,j} + \sum_{i=1}^{N-1} t_{ij} + w_j \right) \tag{4.17}$$

m. Nilai *demand, arrival time, waiting time*, dan *service time* pada depot adalah 0.

$$q_{1j} = a_{1j} = w_{1j} = s_{1j} = 0 \left(\forall j \in \{1,...,N\} \right) \tag{4.18}$$

Masing-masing *constraint* dibedakan menjadi dua tipe yaitu ***Hard Constraint*** **(HC)** dan ***Soft Constraint*** **(SC)**. Hard *constraint* adalah batasan yang wajib dipenuhi dalam menentukan solusi yang layak *(feasible)*. Sedangkan *soft constraint* adalah batasan yang harus dipenuhi untuk mencapai tujuan (*objective*) dari optimasi yang dilakukan. Penjelasan terkait notasi-notasi yang digunakan pada formula disajikan pada Tabel 4.1.

Tabel 4. 1 Keterangan notasi pada formula

Notasi	Keterangan
N	Sekumpulan simpul yang harus dikunjungi oleh kendaraan
K	Sekumpulan kendaraan yang tersedia untuk mengunjungi simpul-simpul
Q	Maksimal kapasitas pada masing-masing kendaraan
q_j	*Demand* pada pelanggan j
C_{ij}	Jarak atau waktu yang dibutuhkan untuk berjalan dari simpul i ke simpul j
$X_{k,ij}$	Fungsi yang merepresentasikan perjalanan dari simpul i ke simpul j
$\bar{X}$	Rata-rata jarak tempuh semua kendaraan
s_{kj}	Waktu yang dibutuhkan kendaraan k untuk memberikan *service* kepada pelanggan j
e_j	*Ready time*, batasan waktu awal pelanggan j dapat menerima *service*
l_j	*Due date*, batasan waktu akhir pelanggan j dapat menerima *service*
a_j	Waktu kendaraan sampai pada pelanggan j
b_j	Waktu kendaraan memulai service pada pelanggan j
t_{ij}	Waktu yang dibutuhkan kendaraan untuk berjalan dari simpul i ke simpul j
d_{ij}	Jarak yang ditempuh kendaraan untuk berjalan dari simpul i ke simpul j, dihitung dengan menggunakan rumus eucledian distance.
W_{ij}	Waktu tunggu kendaraan dari pelanggan i untuk memulai service pada pelanggan j

4.2 Desain Algoritma Hiperheuristik

Pada bagian desain algoritma ini berisi desain dan penjelasan dari inisialisasi solusi, desain algoritma pemilihan LLH, desain algoritma LLH, desain algoritma move acceptance dan desain algoritma seleksi solusi pareto. Masing-masing desain akan dijelaskan dengan lebih detail pada Subbab 4.1.1 sampai dengan Subbab 4.1.5. Pseudocode 4.1 merupakan design dari algoritma Hiperheuristik yang diusulkan pada penelitian ini. Penjelasan secara rinci untuk masing-masing tahapannya akan

diberikan pada bagian selanjutnya. Secara umum, metode Hiperheuristik terdiri dari 3 tahapan yaitu:

1. Inisialisasi solusi, tahapan awal untuk menghasilkan solusi inisial.
2. LLH Selection, tahapan memilihan atau seleksi terhadap LLH untuk melakukan modifikasi terhadap solusi inisia1.
3. Move Acceptance, tahapan yang dilakukan untuk menentukan apakah solusi baru diterima atau ditolak.

Selain tahapan-tahapan di atas, pada penelitian ini juga dilakukan tahapan *Pareto Sorting* untuk menyeleksi solusi-solusi yang tidak terdominasi oleh solusi yang lain. Tahapan ini merupakan tahapan tambahan agar metode Hiperheuristik dapat beradaptasi dengan permasalahan optimasi objektif jamak. Tahapan ini dikembangkan dari penelitian-penelitian sebelumnya yang berkaitan dengan optimasi permasalahan objektif jamak (Baños et al., 2013b, 2013d; Melián-Batista et al., 2014)

```
1   //Membangkitkan solusi awal dengan metode constructive heuristic
    Initialize solution using constructive heuristic
2   //Menciptakan solusi yang lebih optimal dari solusi awal
3   while (time is not expired) do
4       Select Low Level Heuristic; //menyeleksi LLH yang ada
5       Apply selected LLH; //mengimplementasikan LLH yang terpilih pada solusi
        sebelumnya
6       Generate/update SolutionSet; //menambahkan solusi baru pada
        SolutionSet
7       Apply move acceptance; //menentukan solusi yang akan menjadi solusi
        awal pada iterasi selanjutnya
8   end while
9   //Menyeleksi solusi-solusi pada SolutionSet untuk dimasukkan ke ParetoSet
    (nondominated solutions)
10  Generate/update ParetoSet;
11  Sort ParetoSet;
12  return ParetoSet
```

Kode Program 4. 1 Pseudocode Algortima Hiperheuristik untuk permasalahan Objektif jamak (Usulan)

Tahapan yang membedakan antara Hiperheuristik untuk permasalahan tunggal dan permasalahan multiobjetif adalah terletak pada method Generate ParetoSet dan Sort Pareto Set. Hal ini berkaitan dengan solusi akhir yang dihasilkan dari implementasi algoritma. Pada permasalahan objektif tunggal dihasilkan solusi

tunggal juga, namun permasalahan objektif jamak perlu menciptakan beberapa solusi optimal untuk dapat memenuhi semua objektifnya. Pseudocode multiobjective Hyperheuristic disajikan secara lebih detail oleh Pseudocode 4.2.

```
1   Procedure SOLVE (Problem Domain)
2       Set time limit
3       Initialize Solution
4       Set old_OF1 to positive infinity
5       Set old_OF2 to positive infinity
6       While time not expired do {
7           Set random weight, w1
8           Set w2 = 1 - w1
9           Apply simple random selection to LLH
10          Apply selected LLH
11          Get new_OF1 and new_OF2
12          Normalize new_OF1 and new_OF2
13          Get new_OF from (w1*new_OF1  + w2*new_OF2)
14          Add new solution to SolSet
15          Apply move acceptance}
16      End while
17      Select non-dominated solution
18      Sort ParetoSet
19  End procedure
```

Kode Program 4. 2 Pseudocode Detail Algoritma Objektif jamak Hiperheuristik (usulan)

Pada pseudocode 4.2 terdapat fungsi pembobotan (baris ke 7 dan 8) yang digunakan untuk mengatur kontribusi masing-masing objektif pada nilai fungsi objektif. Nilai pembobotan berubah-ubah pada setiap iterasi sehingga digunakan fungsi random number. Sesuai dengan penelitian sebelumnya bahwa Random Number digunakan untuk meningkatkan diversifikasi solusi sehingga tidak terjebak local optima (Muklason, 2017; Walker, 2015b). Hal ini karena algortima yang seimbang adalah algoritma yang memenuhi dua aspek yaitu diversifikasi dan eksplorasi. Aspek eksplorasi dipenuhi dengan penggunaan LLH untuk memodifikasi solusi pada setiap iterasi.

Baris ke 14 dan 18 pseudocode 4.2 berisi tentang strategi pengelolaan solusi. Terdapat 2 Set yaitu SolSet dan ParetoSet. SolSet untuk menampung solusi yang diterima pada setiap iterasi dan menyimpan solusi yang akan dimodifikasi oleh LLH pada iterasi selanjutnya. ParetoSet adalah himpunan solusi optimal yang akan dihasilkan pada akhir running.

Pengelolaan solusi ini sedikit mengadopsi dari penelitian sebelumnya (Melián-Batista et al., 2014) yang menyimpan solusinya pada dua macam Set dengan fungsinya masing-masing. Pada penelitian terdahulu yang menggunakan metode metaheuristik di Pseudocode 4.3 baris 12, 13 dan 19 dapat diketahui bahwa terdapat dua macam Set yang digunakan untuk menyimpan solusi. EfficientSet menyimpan solusi-solusi optimal sedangkan RefSet menyimpan solusi yang dijadikan input pada combination process dan solusi yang dihasilkan dari setiap iterasi.

```
1   For i ← 0, number oof iterations do
    //Menciptakan Initial Population
2     while (|Pop| < PopSize or a maximu number of iterations is not reached) do
3       Construct a feasible solution S
4       Apply the Local Searches in the following order:
5       S^1 ← LocalSearchDistance (S)
6       S^2 ← LocalSearchTimeBalance (S^1)
7       S^3 ← LocalSearchDistance (S^2)
8       S^4 ← LocalSearchCompromise (S^3)
9       Pop ← Pop U { S^4}
10    end while
    //Generate the Efficient Set, Ê, and run the remaining steps of Scatter Search
11    while (new solutions are added to the EfficientSet or a maximum number of
      iterations is not  reached) do
12      Generate or Update the Efficient Set, Ê
13      Generate the RefSet consisting of b feasible solutions
14      Select subsets of solutions in RefSet and apply the combination process to
15    those subsets
16      Run the LocalSearchCompromise  to the solutions obtained in the previous
17    step
18      Update RefSet
19    end while
    end for
    return Efficient Set, Ê
```

Kode Program 4. 3 Pseudocode Algoritma Metaheuristic BioSS (prior research)

4.2.1. Desain Algoritma Inisialisasi Solusi

Tahapan awal dari metode Hiperheuristik adalah menginisiasi solusi awal. Pada penelitian ini digunakan *constructive heuristic* untuk menghasilkan solusi awal dengan tahapan seperti pada Pseudocode 4.4. Metode ini menciptakan solusi awal dengan menyeleksi dan menyusunnya dari setiap simpul sampai semua simpul (*customer*) masuk ke dalam rute tertentu. Seleksi terhadap setiap simpul dilakukan berdasarkan nilai score yang terdiri dari nilai jarak dan waktu setiap customer.

Customer yang memiliki jarak dan waktu yang lebih dekat dengan simpul sebelumnya akan lebih dipilih.

```
1   //Menginisiasi CustomerList C, depot: C[0]
2   Initiate CustomerList
3   //Menginisiasi rute kosong r
4   Create Empty Route (depot)
5   //Menginisiasi RouteList R
6   Initiate RouteList
7   //Memasukkan customer satu per satu ke dalam rute
8   while size (C) >0 do
9       if feasibleCustExistForRoute (r,C) then
10          GetFinalCustomerInRoute r = cLast
11          //Memilih customer yang memiliki score terendah untuk dimasukkan ke
    rute
12          // score = (c_dist + c_time)*randomNumber
13          SelectBestCustomer c
14          // Memasukkan customer yang terpilih ke dalam rute r dengan posisi
    paling belakang
15          InsertAtTheEndOfRoute (r,c)
16          //Menghapus customer dari CustomerLists
17          RemoveFromList (C,c)
18      Else
19          //Menginisiasi rute baru yang kosong
20          CreateEmptyRoute
21      end if
22  end While
23  //Membuat solusi baru s dari RouteList R
24  CreateSolutionFromRoutes
```

Kode Program 4. 4 Pseudocode Generate solusi awal

4.2.2. Desain Algoritma Seleksi LLH

Metode seleksi LLH yang digunakan pada penelitian ini adalah metode simple random (Li et al., 2017). Metode ini memilih LLH secara random untuk memodifikasi solusi yang telah dibangkitkan pada tahapan sebelumnya. Pada penelitian (Li et al., 2017) solusi inisialnya berbasis populasi dan dibuat secara random. Namun pada penelitian ini solusi inisial dihasilkan dari tahapan sebelumnya secara *constructive*, hanya LLH yang dipilih secara random.

4.2.3 Desain Algoritma LLH

Domain VRP memiliki 12 LLH yang tergolong pada 4 kategori heuristik. Kategori heuristic terdiri dari Mutasi, Ruin Recreate, Crossover dan Local Search.

Desain LLH yang digunakan pada penelitian ini mengacu pada penelitian sebelumnya (Walker, 2015b).

4.2.3.1 Heuristik Mutasi

Heuristic mutasi adalah memindahkan suatu simpul dari suatu simpul ke tempat yang lain. Mutasi terdiri dari empat macam operasi yaitu two-opt, or-opt, shift dan interchange.

4.2.3.1.1 Two-opt

Pada heuristic ini dilakukan penukaran dua simpul yang berdekatan dalam satu rute. Pseudocode 4.5 menunjukkan tahapan yang dilakukan ketika memodifikasi solusi dengan menggunakan LLH two-opt.

```
1   Procedure TWO-OPT (s)
2     For i ← 0, intensityOf Mutation * numOfRoutes do
      // jumlah iterasi ditentukan berdasarkan hasil kali dari jumlah rute dan parameter jumlah
      penukaran/swap yang harus dilakukan.
3       Route r ← selectRandomRoute (s) //memilih rute secara random pada solusi
4       Route r' ← r
5       Customer cj ← selectRandomCustomer (r') //memilih 1 customer secara random dari
        rute yang terpilih
6       r' ← swapPosition (cj, c j-1, r')
        //menukar posisi customer yang terpilih dengan customer yang posisinya berada tepat
        sebelum customer tersebut, pada rute yang sama
7       If feasible (r') then
        // jika rute yang terbentuk feasible maka rute tersebut didefinisikan menjadi rute baru
        dan solusinya dapat diterima
8         r ← r'
9       End if
10    End for
```

Kode Program 4. 5 Pseudocode Two-opt Mutasi

Implementasi LLH two-opt dapat merubah total jarak, namun tidak merubah jumlah kendaraan atau rute yang diperlukan. Parameter *intensity of mutation* digunakan untuk menentukan jumlah penukaran/swap yang harus dilakukan terhadap solusi. Contoh implementasi two-opt adalah misalnya terdapat rute dengan susunan D→1→2→3→D maka implementasi two-opt akan merubahnya menjadi D→2→1→3→D. Perubahan yang dilakukan adalah menukar simpul 1 dan 2.

4.2.3.1.2 Or-opt

Pada LLH Or-opt dilakukan pemindahan 2 simpul yang berurutan dalam rute yang sama ke lokasi lain pada rute yang sama. Proses tersebut dijelaskan secara lebih detail pada Pseudocode 4.6. LLH ini jika diimplementasikan akan mempengaruhi jarak, namun tidak mempengaruhi jumlah kendaraan atau rute. Contoh implementasi or-opt adalah D→1→2→3→4→5→D menjadi D→1→2→5→3→4→D.

```
1   Procedure OR-OPT (s)
2     For i ← 0, intensityOf Mutation * numOfRoutes do
      // jumlah iterasi ditentukan berdasarkan hasil kali dari jumlah rute dan parameter jumlah
      penukaran/swap yang harus dilakukan.
3       Route r ← selectRandomRoute (s) //memilih rute secara random pada solusi
4       Route r' ← r
5       Customer c_j ← selectRandomCustomer (r') //memilih 1 customer secara random
        (selain customer yang posisinya paling belakang) dari rute yang terpilih
6       r' ← removeCustFromRoute (r',c_j)
        //menghapus customer yang terpilih dari rutenya
7       r' ← removeCustFromRoute (r',c_j+1)
        //menghapus customer yang berposisi tepat setelah customer yang terpilih, dari
        rutenya
8       Customer c_k ← selectRandomCustomer (r') //memilih 1 customer yang lain secara
        random dari rute yang sama
9       r' ← insertCustBeforeCust (c_j,c_k, r')
        //menambahkan customer yang telah dihapus pada posisi setelah customer yang
        terpilih
10      r' ← insertCustBeforeCust (c_j+1,c_k, r')
        //menambahkan customer yang telah dihapus pada posisi setelah customer yang
        terpilih
11      If feasible (r') then
        // jika rute yang terbentuk feasible maka rute tersebut didefinisikan menjadi rute baru
        dan solusinya dapat diterima
12        r ← r'
13      End if
14    End for
15  End Procedure
```

Kode Program 4. 6 Pseudocode Or-opt Mutasi

4.2.3.1.3 Shift

Shift adalah heuristic yang memindahkan suatu simpul dari satu rute ke rute yang lain seperti tahapan yang digambarkan pada Pseudocode 4.7. Heuristik ini jika diimplementasikan pada solusi akan mempengaruhi jarak dan jumlah rute atau kendaraan yang diperlukan. Potensi perubahan jumlah rute disebabkan oleh adanya pemindahan customer dari satu rute ke rute yang lainnya. Apabila tidak ada rute

yang feasible untuk menampung customer tersebut maka akan dibuat rute baru, hal ini akan menambah jumlah rute atau kendaraan yang diperlukan. Proses penyisipan customer dilakukan dengan tahapan yang ditunjukkan pada Pseudocode 4.8.

```
1   Procedure SHIFTMUTATE (s)
2     For i ← 0, intensityOf Mutation * numOfRoutes do
      // jumlah iterasi ditentukan berdasarkan hasil kali dari jumlah rute dan parameter jumlah
      penukaran/swap yang harus dilakukan.
3       Route r ← selectRandomRoute (s) //memilih rute secara random pada solusi
4       Customer c← selectRandomCustomer (r) //memilih 1 customer secara random dari
        rute yang terpilih
5       r ← removeCustFromRoute (c,r)
        //menghapus customer yang terpilih dari rutenya
6       s ← insertCust (s,c)
        //memasukkan customer yang telah dihapus ke rute lain
7     End for
8   End Procedure
```

Kode Program 4. 7 Pseudocode Shift Mutasi

Pertimbangan penting ketika mendesain heuristik mutasi adalah menjamin adanya variasi dan meningkatkan keberagaman terhadap pemindahan simpul sehingga tidak berkali-kali memindahkan simpul yang sama. Strategi untuk mengatasi hal ini adalah dengan pemilihan simpul dan rute secara random.

```
1   Procedure INSERTCUST (s,c)
2     If thereExistsFeasibleRoute (s,c) then
      //iterasi dilakukan jika masih ada rute yang feasible untuk ditempati customer yang akan
      dipindahkan
3       Customer route c',r ← selectBestInsertionPosition (s,c)
        //menentukan posisi terbaik untuk menempatkan suatu customer, pemilihan
      didasarkan pada waiting time yang paling kecil.
4       r← insertBeforeCust (r,c,c')
        //menempatkan customer pada posisi yang paling optimal
5     Else //apabila tidak ada rute yang feasible untuk ditempati maka akan dibuat rute baru
6       Routes R ←getRoutes (s)
7       Routes r ← createNewRoutes ( ) //membuat rute baru
8       r ←insertAtEnd (r,c) // customer ditempatkan pada posisi paling belakang atau
      setelah depot
9       R← {R:r} // rute baru masuk ke himpunan Rute
10    End if
11  End Procedure
```

Kode Program 4. 8 Pseudocode Penyisipan Customer

Contoh implementasi shift mutasi adalah Rute 1: D→1→2→D, Rute 2: D→1→2→3→D menjadi Rute 1: D→1→1→2→D, Rute 2 : D→2→3→D. Customer 1 pada Rute 2 dipindahkan ke Rute 1.

4.2.3.1.4 Interchange

Heuristic interchange mengakibatkan perubahan jarak pada solusi namun tidak merubah jumlah rute atau kendaraan yang dibutuhkan. Hal ini karena pada LLH ini dilakukan penukaran simpul dari rute yang berbeda. Penentuan customer pertama dilakukan dengan menggunakan metric yang didasarkan pada seberapa "out of place" masing-masing customer pada rute pertama. Rumusnya adalah selisih antara due date customer sebelum c_1 (c_{j-1}) dan customer setelahnya c_{j+1}). Selain itu juga selisih antara nilai eucledian distance antara customer-customer tersebut. Cara perhitungan score dilakukan sesuai dengan Rumus 4.1.

Score = (due date + nilai eucledian distance) * randNumber
(4.1)

Pemilihan rute kedua dilakukan dengan mencari rute yang secara umum customer-customernya dekat dengan customer yang akan disisipkan. Rumus untuk menghitung score pemilihan rute adalah nilai rata-rata dari jarak antara semua customer pada rute dengan customer pertama. Semakin kecil nilai rata-ratanya maka akan semakin dipilih. Pemilihan customer kedua dilakukan dengan rumus yang sama dengan pemilihan customer pertama. Customer kedua dipilih dari score yang paling rendah. Pseudocode 4.9 adalah cara kerja heuristic interchange.

```
1   Procedure INTERCHANGE (s)
2       For all Route r in s do
3           For i ← 0, intensityOf Mutation * numOfRoutes do
4               Customer c1 ← selectFirstCustomer (r)
                //menentukan customer yang akan dipindahkan dari rute pertama
5               Route r2 ← selectSwapRoute (c1, s)
                //menentukan rute kedua yang layak untuk disisipi customer dari rute pertama
6               Customer c2 ← selectBestSwapCustomer (r)
                //menentukan customer dari rute kedua yang paling layak untuk ditukar dengan customer
            terpilih dari rute pertama
7               r, r2 ← swapCustomers (c1, r, c2, r2)
                //menukar customer dari rute pertama dan kedua
8           End for
9       End if
10  End Procedure
```

Kode Program 4. 9 Pseudocode Interchange Mutasi

4.2.3.2 Heuristik Ruin Recreate

Pada heuristic ini dipilih sejumlah customer untuk dihapus berdasarkan kedekatannya (*proximity*) dengan base Customer, kemudian posisi customer yang dihapus akan diatur kembali. Customer yang memiliki nilai proximity dengan baseCustomer yang besar pada batasan (Limit) tertentu akan dihapus. Proximity dihitung dengan berdasarkan nilai eucledian distance dan selisih time windows. Oleh karena itu heuristic ini terdiri dari dua jenis yaitu Radial Ruin yang berdasarkan lokasi (jarak) dan radial ruin yang berdasarkan pada waktu.

4.2.3.2.1 Location-based Radial Ruin

LLH ini menggunakan ukuran jarak untuk menentukan Customer yang akan dihapus dan disisipkan kembali. Pesudocode 4.10 menunjukkan langkah kerja LLH ini. Parameter yang digunakan pada LLH ini adalah Distance Limit. Distance limit digunakan sebagai pedoman ukuran untuk menghapus customer dalam jumlah yang banyak, tetapi tidak semuanya. Jika semua customer dihapus maka akan tidak ada bedanya dengan pembuatan solusi inisial. Nilai 4/5 pada rumus distance limit dipilih secara acak, nilai ini dipilih agar jumlah mutasinya besar ketika diperlukan oleh nilai parameter intersityOfMutation. FurthestFromDepot adalah jarak antara depotnya baseCustomer dan customer yang paling jauh dari depot tersebut.

```
1   Procedure LOCATIONRR (s)
2     Customer baseC ← selectRandomCustomer (s)
      //menentukan baseCustomer secara random
3     distLimit ← (intensityOfMutation * 4/5 furthestFromDepot (s,baseC))
       //rumus distance limit,
4     removedCusts ← baseC //menghapus baseCustomer
5     For all customer c in s do
6       If calcDistance (c, baseC) > distLimit then
        //jika jarak antara customer dengan baseCustomer lebih besar maka customer
        tersebut dihapus
7         s← removedCustFromSol (c,s)
8         removedCusts ← c
9       End If
10    End for
11    For all Customer c in removedCusts do  //semua customer yang telah dihapus akan
    disisipkan kembali
12      s ← insertCust (s,c) //penyisipan dilakukan seperti pada shift mutate
13    End for
14  End Procedure
```

Kode Program 4. 10 Pseudocode Location-based Radial Ruin

4.2.3.2.2 Time -based Radial Ruin

Secara umum, prosedur pada heuristic ini sama dengan location based radial ruin. Namun yang membedakan adalah pengukuran proximity pada heuristic ini dilakukan berdasarkan selisih waktu kedatangan antara baseCustomer dan customer yang dipertimbangkan untuk dihapus atau tidak.

4.2.3.3 Heuristik Local Search

Hampir semua Heuristik pada LS memiliki nama yang sama dengan heuristic pada Mutasi, kecuali GENI. Namun dari sisi cara kerja, semua heuristic tersebut saling memiliki perbedaan. Perbedaan yang paling terlihat antara Mutasi dan LS adalah adanya solution acceptance "first improvement", operator atau heuristic akan menerima solusi pertama yang ditemukan ketika solusi tersebut memiliki nilai Objective Function (OF) yang sama atau lebih kecil daripada solusi sebelumnya (permasalahan minimasi), OF lebih besar untuk permasalahan maksimasi. Selain itu juga terdapat "best improvement", ketika semua kemungkinan moves dipertimbangkan sebelum memilih solusi akhir yang dihasilkan dari solusi yang memiliki penurunan nilai OF yang paling signifikan. Kekurangan dari heuristic ini adalah:

1. Lebih mahal jika akan melakukan moves dengan jumlah yang banyak karena harus mempertimbangkan nilai OF dari solusi sebelum solusi baru tersebut diterima.
2. Penerimaan solusi terbaik dapat menyebabkan stuck/ terjebak local optima dan merusak keseluruhan pencarian (Walker, 2015b).

Solusi yang memiliki nilai OF yang sama akan tetap diterima karena mempunyai komposisi solusi yang berbeda, hal ini untuk meningkatkan diversifikasi sehingga berpotensi mencegah local optimum. Parameter "depth of search" digunakan untuk menentukan berapa kali proses move dan penerimaan akan diulangi.

4.2.3.3.1 Shift

Shift adalah LLH yang memindahkan satu simpul/customer dari satu rute ke rute yang lainnya. Customer yang akan dipindahkan dipilih berdasarkan metric yang berisi jarak antara masing-masing customer dengan customer lain pada rute

yang lain. Pada metric tersebut terdapat nilai randomness untuk mencegah pemilihan berulang-ulang pada customer yang sama. Customer yang memiliki nilai metric yang paling besar disebut sebagai "most out place" dan dihapus dari rute asal. Penghapusan dilakukan untuk menghilangkan customer yang berkontribusi menjadikan total jarak yang lebih besar. Oleh karena itu, pada akhirnya akan dihasilkan total jarak tempuh yang lebih pendek. Pada heuristic ini solusi yang mengalami penurunan atau memiliki nilai OF yang sama akan diterima. Pseudocode 4.11 menunjukkan langkah kerja Shift pada heuiristik Local Search.

```
1   Procedure SHIFTLS (s)
2     For i ← 0, intensityOf Mutation * numOfRoutes do
      // jumlah iterasi ditentukan berdasarkan hasil kali dari jumlah rute dan parameter jumlah
      penukaran/swap yang harus dilakukan.
3       Route r ← selectRandomRoute (s) //memilih rute secara random pada solusi
4       Customer c← selectCustomerBasedOnMetric (r) //memilih 1 customer berdasarkan
        metric (yang membedakan dengan shift Mutate)
5       r ← removeCustFromRoute (c,r)
        //menghapus customer yang terpilih dari rutenya
6       s ← insertCust (s,c)
        //memasukkan customer yang telah dihapus ke rute lain (cara sama dengan mutasi)
7     End for
8   End Procedure
```

Kode Program 4. 11 Pseudocode Shift Local Search

```
1   Procedure INTERCHANGELS (s)
2     For all Route r in s do
3       For i ← 0, intensityOf Mutation * numOfRoutes do
4         Customer c1 ← selectFirstCustomer (r)
          //menentukan customer yang akan dipindahkan dari rute pertama
5         Route r2 ← selectSwapRoute (c1, s)
          //menentukan rute kedua yang layak untuk disisipi customer dari rute pertama
6         Customer c2 ← selectBestSwapCustomer (r)
          //menentukan customer dari rute kedua yang paling layak untuk ditukar dengan
          customer terpilih dari rute pertama
7         r, r2 ← swapCustomers (c1, r, c2, r2)
          //menukar customer dari rute pertama dan kedua
8       End for
9     End if
10  End Procedure
```

Kode Program 4. 12 Pseudocode Interchange Local Search

4.2.3.3.2 Interchange

Interchange adalah LLH yang memodifikasi solusi dengan cara menukar dua simpul dari rute yang berbeda. Algoritma langkah kerja dari heuristic ini ditunjukkan pada Pseudocode 4.12. Cara kerja sama dengan heuristic interchange pada mutasi, namun yang membedakan adalah heuristic ini hanya menerima solusi yang mengalami penurunan nilai OF atau memiliki nilai OF yang sama dengan solusi yang sebelumnya.

```
1   Procedure TWO-OPT (s)
2     For i ← 0, depthOf Search * numOfRoutes do
      // jumlah iterasi ditentukan berdasarkan hasil kali dari jumlah rute dan parameter jumlah
      penukaran/swap yang harus dilakukan.
3       Solution s' ← s
4       Route r1 ← selectRandomRoute (s) //memilih rute pertama secara random pada
      solusi
5       Route r2 ← selectRandomRoute (s) //memilih rute kedua secara random pada solusi
6       bestScore ← getMaximumNumber ( )
7       Customer best c1 ← -1
8       Customer best c2 ← -1
9      // memilih start point untuk masing-masing rute dengan mempertimbangkan "best
      improvement process"/start point yang akan menghasilkan solusi yang lebih optimal
       For all customer c1 in r1 do
10       For all customer c2 in r2 do
11         Score = calcSwapScore (c1, c2, r1, r2) //semua kemungkinan start poin pada kedua
      rute akan dipertimbangkan berdasarkan skor dan dicek kelayakannya (feasibility)
12         If score < bestScore then
13           bestScore ← score
14           best c1 ← c1
15           best c2 ← c2
16         End if
17         End for
18       End for
19      s' ← performSwap (best c1, best c2, r1, r2)
       //menukar/swap antara customer yang mengikuti c1 pada r1 dengan customer yang
      mengikuti c2 pada r2 sehingga semua customer yang mengikuti start point akan ditempatkan
      di ujung rute yang lain.
20      If OF(s') < OF(s) then
21      s←s'
22      End if
23    End for
24  End Procedure
```

Kode Program 4. 13 Pseudocode Two-opt Local Search

4.2.3.3.3 Two-opt

Cara LLH two-opt dalam memodifikasi solusi adalah menukar masing-masing 2 simpul yang letaknya diakhir dari 2 rute untuk menghasilkan 2 rute baru. Heuristik ini powerful untuk menghasilkan solusi yang lebih optimal dari segi jarak maupun jumlah rute. Jumlah rute dapat berkurang pada kondisi di mana start poin dari suatu rute dipilih sebagai kunjungan terkahir ke depot dan start poin dari rute yang lain adalah customer pertama pada rute tersebut sehingga hal ini sama seperti menggabungkan kedua rute tersebut. Pseudocode 4.13 merupakan algoritma two-opt pada heuristic local search. Parameter depth of search digunakan untuk mengontrol berapa kali proses akan dilakukan. Depth of search dikalikan dengan jumlah rute agar jumlah iterasi sesuai dengan ukuran solusi.

```
1  Procedure GENI (s)
2    For i ← 0, intensityOf Mutation * numOfRoutes do
       // jumlah iterasi ditentukan berdasarkan hasil kali dari jumlah rute dan parameter jumlah
     penukaran/swap yang harus dilakukan.
3      Route r ← selectRandomRoute (s) //memilih rute secara random pada solusi
4      Customer c← selectCustomerBasedOnMetric (r) //memilih 1 customer berdasarkan
       metric (yang membedakan dengan shift Mutate)
5      r ← removeCustFromRoute (c,r)
       //menghapus customer yang terpilih dari rutenya
6      s ← insertCust (s,c)
       //memasukkan customer yang telah dihapus ke rute lain dan diletakkan diantara 2
       customer lain pada rute terdekat (diukur dengan metric yang sama seperti pemilihan
       customer yang dihapus)
7    End for
8  End Procedure
```

Kode Program 4. 14 Pseudocode GENI Local Search

4.2.3.3.4 GENI

Perbedaan shift dan GENI adalah berkaitan dengan penempatan customer yang dihapus pada rute yang lain. Pada GENI, penempatannya diantara 2 customer pada rute terdekat dengan customer yang dihapus. Misalnya kedua customer tersebut adalah c_1 dan c_2 serta customer yang akan disisipkan adalah ic1. Terkadang c_1 dan c_2 tidak berurutan pada rute sehingga perlu pengurutan ulang customer pada rute. Customer yang akan disisipkan ditempatkan setelah c_1 atau c_2 (customer yang lebih awal urutannya). Misalnya c_1 berada pada urutan yang lebih awal sehingga semua customer yang berada sebelum c_1 akan memiliki urutan yang tetap. c_2 terletak setelah ic1 dan c_1. Semua customer yang tetap pada posisinya akan

disisipkan pada posisi mengikuti c_2. Pada setiap iterasi, customer yang memiliki jarak terdekat disisipkan pada akhir rute. Proses ini diulang-ulang sampai semua customer telah masuk pada rute. Kemudian dicek kelayakan solusinya, penerimaan solusi dilakukan terhadap solusi yang memiliki nilai OF yang lebih optimal atau sama. Psudocode LLH GENI ditunjukkan pada Pseudocode 4.14.

```
1   Procedure COMBINE (s1, s2)
2     Solution first ← s1 //solusi parent 1
3     Solution first ← s2 //solusi parent 2
4     Solution newS ← createEmptySolution ( ) //solusi child
5     If randomNumber ( ) < 0.5 then //menentukan solusi yang akan terlebih dahulu
      dipertimbangkan
6       First ← s2
7       Second ← s1
8     End if
9     randVal ← randomNumberBetween (0.25, 0.75) //untuk mengontrol jumlah rute dan
      diversifikasi solusi
10    For all Route r in first do
11      If randomNumber () < randVal then //jika benar, maka rute r akan dimasukkan ke
        solusi child
12        NewS ← newS:r
13      End if
14    End for
15    For all Route r in second do
16      If randomNumber () < randVal then
17        If noCustomerConflicts (r, newS) then //rute dimasukkan ke solusi baru jika dan
          hanya jika tidak terdiri dari customer yang telah ada pada solusi child
18          New S ← newS:r
19        End if
20      End if
21    End for
22    ListUnroutedC ← determineUnroutedCustomer (newS, s1) //customer-customer yang
      belum masuk ke solusi akan disisipkan satu per satu ke solusi child
23    For all Customer C inUnroutedC do
24      newS ← insertCust (c, newS) //cara penyisipan seperti pada heuristic shift mutate
25    End for
26    return newS //solusi child selalu diterima, tanpa mempertimbangkan apakah solusi
      tersebut memiliki nilai OF yang lebih optimal atau tidak
27  End Procedure
```

Kode Program 4. 15 Pseudocode Combine Crossover

4.2.3.4 Heuristik Crossover

Crossover adalah heuristic yang menjadikan 2 solusi sebagai input dan mengambil solusi baru hasil turunan dari input. Tujuan utamanya adalah mengkombinasikan kualitas terkuat dari 2 parent agar dihasilkan superior child.

Heuristik ini terdiri dari dua operasi/LLH yaitu combine dan combine long yang masing-masing memiliki langkah kerja yang ditunjukkan pada Pseudocode 4.15 dan 4.16.

4.2.3.4.1 Combine

Pada LLH Combine dilakukan kombinasi antara 2 solusi parent menjadi solusi child. Penggunaan parameter yang dihasilkan dari random number karena untuk mencegah pemilihan solusi yang sama dalam setiap iterasinya. Selain itu juga digunakan untuk mengontrol jumlah rute.

4.2.3.4.2 Combine Long

Pada combine long, pemilihan rute didasarkan pada panjang jarak atau jumlah customer yang ada pada rute. Rute yang panjang lebih dipilih karena akan menghasilkan solusi yang memiliki total jarak tempuh yang lebih pendek dan membutuhkan rute yang lebih sedikit. Oleh karena itu akan dihasilkan child yang lebih berkualitas. Heuristic ini memiliki kemungkinan conflict yang lebih besar dibandingkan dengan combine karena dihasilkan dari kombinasi parent yang sama-sama berkualitas tinggi.

```
1   Procedure COMBINE (s1, s2)
2     Solution newS ← createEmptySolution ( ) //solusi child
3     Routes ← orderRoutesBySize (s1, s2) //indicator pemilihan rute berdasarkan urutan
      jumlah customer atau panjang jarak secara descending
4     For all Route r in Routes do
5       If noCustomerConflicts (r, newS) then //rute dimasukkan ke solusi baru jika dan
      hanya jika tidak terdiri dari customer yang telah ada pada solusi child
6           New S ← newS:r
7       End if
8     End for
9     ListUnroutedC ← determineUnroutedCustomer (newS, s1) //customer-customer
      yang belum masuk ke solusi akan disisipkan satu per satu ke solusi child
10    For all Customer C inUnroutedC do
11      newS ← insertCust (c, newS) //cara penyisipan seperti pada heuristic shift mutate
12    End for
13    return newS //solusi child selalu diterima, tanpa mempertimbangkan apakah solusi
      tersebut memiliki nilai OF yang lebih optimal atau tidak
14  End Procedure
```

Kode Program 4. 16 Pseudocode Combine Long

4.2.4 Desain Algoritma *Move Acceptance*

Metode penerimaan solusi yang digunaka pada penelitian ini adalah Hill Climbing dan Great Deluge (Demeester et al., 2012). Kedua metode tersebut digunakan dalam eksperimen sehingga dapat diketahui metode move acceptance yang lebih baik dan dapat menghasilkan solusi dengan nilai coverage yang lebih kecil. Flowchart kedua algoritma disajikan pada Lampiran B.

4.2.5 Desain Algoritma Seleksi Solusi Pareto (*Pareto Sorting*)

Solusi yang dihasilkan pada setiap iterasi akan disimpan pada SolSet. Namun ini bukanlah solusi akhir, selanjutnya perlu dilakukan seleksi terhadap solusi-solusi pada SolSet sehingga didapatkan solusi yang tidak terdominasi oleh solusi yang lainnya (solusi pareto). Cara menyeleksi solusi-solusi tersebut dilakukan dengan algoritma pareto sorting yang ditunjukkan pada pseudocode 4.17.

```
1   //Menginisiasi ParetoSet
2   Initiate pareto optimal solution set
3   //Mengurutkan Solusi-solusi dari yang terkecil sampai terbesar berdasarkan OF1
4   Sort SolutionSet by OF1
5   //Initiate b as positive infinity
6   b= ∞
7   for i=1 to size of SolutionSet do
8     if OF2(si) < b then
    // OF2(si) adalah nilai fungsi objektif kedua, si adalah solusi ke i dari SolutionSet
    yang diurutkan berdasarkan OF1
9         ParetoSet ← ParetoSet U si
10        b ← OF2(si)
11    end if
12  end for
13  return ParetoSet
```

Kode Program 4. 17 Pseudocode Algoritma selesi solusi pareto

4.3 Implementasi Algoritma

Algoritma yang telah didesain pada bagian sebelumnya, diimplementasikan pada kerangka kerja *HyFlex*. Bahasa pemrograman yang digunakan adalah Java dengan menggunakan aplikasi NetBeans. Implementasi terdiri dari beberapa tahapan yang akan dijelaskan pada masing-masing subbab selanjutnya.

4.3.1 Implementasi Algoritma pada Kerangka Kerja HyFlex

HyFlex merupakan kerangka kerja khusus yang digunakan untuk implementasi algoritma Hiperheuristik. Seperti yang dijelaskan pada Bab 2 Subbab HyFlex, kerangka kerja ini terdiri dari tiga kelas yaitu *Problem Domain*, *Hyperheuristic* dan *Heuristic Type*. Pada penelitian ini implementasi algoritma dilakukan pada kelas *Problem Domain* dan *Hyperheuristic*. Kedua kelas tersebut merupakan kelas abstract yang saling terhubung.

Kelas *Problem Domain* berisi tentang fungsi-fungsi yang berkaitan dengan permasalahan optimasi. Setiap permasalahan optimasi/domain permasalahan memiliki batasan, fungsi tujuan dan variable keputusan. Ketiga komponen tersebut diimplementasikan pada kelas turunan *Problem Domain*. Kelas *Problem Domain* memiliki kelas turunan yaitu domain permasalahan optimasi, seperti VRP, TSP. Penelitian ini melakukan modifikasi terhadap kelas turunan problem domain VRP karena untuk menambahkan fungsi tujuan yang menjadi usulan pada penelitian ini. Pada kelas VRP ditambahkan tahapan perhitungan nilai fungsi objektif yang berkaitan dengan keseimbangan jarak rute kendaraan.

Kelas Hiperheuristik adalah kelas *abstract* yang berisi fungsi-fungsi untuk menghasilkan solusi optimal untuk domain permasalahan. Kelas ini sebagai media untuk mengimplementasikan algoritma Hiperheuristik. Pada kelas ini juga dilakukan modifikasi untuk menghasilkan solusi pareto. Algoritma Hiperheuristik objektif jamak diimplementasikan pada kelas turunan dari kelas Hiperheuristik. Setiap kelas turunan berisi satu metode Hiperheuristik.

Kelas *Problem Domain* dan Hiperheuristik pada penelitian ini memiliki hubungan asosiasi yaitu fungsi-fungsi pada kelas *Problem Domain* dapat dipanggil dan digunakan pada kelas Hiperheuristik. Pada kelas turunan Hiperheuristik dapat memanggil method dan menggunakan nilai fungsi objektif yang dihasilkan oleh kelas *Problem Domain*. Oleh karena itu, pada penelitian ini mengambil nilai kedua fungsi objektif dari kelas VRP.

4.3.2 Fungsi -Fungsi pada HyFlex yang Dibutuhkan

Pada HyFlex telah disediakan fungsi-fungsi untuk menunjang implementasi algoritma Hyperheuristric. Fungsi-fungsi tersebut dapat dipanggil pada kelas turunan Hiperheuristik yang akan dimodifikasi pada penelitian ini. Pada Tabel 4.1 dijelaskan fungsi-fungsi di masing-masing kelas yang dibutuhkan pada penelitian ini. Fungsi-fungsi tersebut hanya digunakan dan tidak dilakukan modifikasi apapun. Fungsi dikelompokkan berdasarkan abstract class yang mengampunya.

Tabel 4. 2 Fungsi-fungsi HyFlex yang dibutuhkan

No	Fungsi/Metode	Penjelasan
Problem Domain Class		
1	*setMemorySize ()* *CopySolution ()*	Digunakan untuk mengkonfigurasi memori penyimpanan solusi yang akan dihasilkan
2	*getNumberOfHeuristics()*	Digunakan untuk mendapatkan index LLH
3	*initialiseSolution (i)*	Untuk menghasilkan solusi awal, i merupakan index dari array solusi
4	*applyHeuristic (i,j,k)*	Untuk memanggil sekumpulan LLH yang digunakan untuk memodifikasi solusi. i: index LLH j: index solusi yang akan dimodifikasi k: index solusi baru hasil modifikasi
5	*copySolution(k, j)*	Untuk mengkonfigurasi memori solusi memindahkan letak penyimpanan solusi yang dimodifikasi pada memori selama iterasi. Solusi dipindahkan dari indeks memory k ke indeks memori j.
6	*loadInstance (a)*	Untuk memanggil instance (dataset) yang akan dicari solusinya a: index instance
Hyperheuristic Class		
7	*toString()*	Untuk memberikan nama pada metode Hiperheuristik
8	*Solve()*	Berisi strategi pemilihan LLH dan move acceptance untuk menghasilkan solusi terbaik.
9	*setTimeLimit()*	Untuk mengatur batas waktu dalam melakukan iterasi
10	*loadProblemDomain()*	Untuk memanggil kelas Problem Domain dan memberikan referensi dlam memanggil objek dari problem domain
11	*run()*	Fungsi yang memulai pengatur waktu dan memanggil fungsi solve pada kelas Hiperheuristik untuk memulai pencarian solusi
12	*hasTimeExpired()*	Untuk memonitor batasan waktu.

Pengembangan masalah menjadi objektif jamak membutuhkan modifikasi dan penambahan fungsi pada kelas Hiperheuristik maupun kelas *Problem Domain*. Beberapa metode atau fungsi yang telah ada pada HyFlex ditunjukkan pada Tabel 4.1 sedangkan fungsi – fungsi yang harus ditambahkan untuk menunjang pengembangan permasalahan objektif jamak terdapat pada Tabel 4.2.

Tabel 4. 3 Fungsi-fungsi yang ditambahkan pada HyFlex

No	Fungsi/Metode	Penjelasan
Problem Domain Class		
1	getFunctionValues(i)	Untuk memanggil Fitness function dari kedua fungsi objektif. i adalah index solusi
VRP Class		
2	CalcOF1()	Untuk memanggil nilai fungsi objektif pertama (jarak total solusi)
3	CalcOF2()	Untuk memanggil nilai fungsi objektif kedua (nilai simpangan baku atau keseimbangan antar jarak rute pada solusi)
Hyperheuristic Inheritance Class		
4	normalisasi(x, y, z)	Untuk merubah nilai fungsi objektif menjadi nilai antara 0 sampai 1
5	denormalisasi(x, y, z)	Untuk mengembalikan nilai fungsi objektif dari 0 sampai 1 menjadi nilai aslinya
6	int toInt ()	Untuk mengubah tipe data double menjadi integer
7	toDouble ()	Untuk mengubah tipe data integer menjadi double
8	SortOF1()	Untuk mengurutkan solusi berdasarkan nilai fungsi objektif pertama dari terkecil sampai terbesar
9	SortOF2()	Untuk mengurutkan solusi berdasarkan nilai fungsi objektif kedua dari terkecil sampai terbesar
Kelas yang lain		
10	Solutionx	Untuk mengatur format solusi dan memastikan tidak ada nilai yang sama untuk menunjangnya disimpan dalam Set

Selain menambahkan fungsi baru, pada penelitian ini dilakukan perubahan terhadap beberapa fungsi operator LLH pada kelas VRP agar return fungsi objektif yang didapatkan dari fungsi-fungsi tersebut berbentuk array. Hal ini karena terdapat dua nilai fungsi objektif yang akan dikeluarkan. Salah satu strategi untuk mengubah konsep single objektif menjadi objektif jamak Hyperheuristic.

4.3.3 Implementasi Fungsi Objektif

Pada penelitian ini digunakan dua fungsi tujuan yaitu meminimalkan total jarak semua rute dan menyeimbangkan jarak antar rute-rutenya. Implementasi fungsi objektif dilakukan pada class vrp.java. Implementasi dilakukan dengan membuat empat fungsi yang ditunjukkan pada Kode Program 4.18 sampai dengan Kode Program 4.20.

Kedua fungsi tujuan ini diimplementasikan melalui fungsi getFunctionValues yang ditunjukkan pada Kode Program 4.18. Fungsi tersebut bertipe double array karena berisi dua nilai dari fungsi tujuan dengan indeks 0 dan 1. Indeks 0 adalah untuk menyimpan nilai fungsi objektif yang pertama sedangkan indeks 1 untuk nilai fungsi objektif kedua. Pada baris 7 terlihat bahwa fungsi tersebut mengembalikan nilai *of* yang telah digandakan (*clone*), sehingga ketika dilakukan modifikasi maka nilai asli *of* tetap original. Hal ini karena fungsi clone akan menciptakan objek baru dan kemudian mengopi isian objek pemanggil ke objek baru.

```
1  @Override
2    public double [] getFunctionValues(int solutionIndex) {
3       ArrayList<Route> routes = solutions[solutionIndex].getRoutes();
4       double [] of = new double[2];
5       of[0]= calcOF1(routes);//total jarak rute
6       of[1]= calcOF2(routes);//simpangan baku
7       return of.clone();
8    }
```

Kode Program 4. 18 Fungsi getFunctionValues

Formula untuk menghitung masing-masing fungsi objektif diimplementasikan pada fungsi yang berbeda yaitu fungsi calcOF1 dan fungsi calcOF2. Pada Kode Program 4.19 ditunjukkan implementasi formula untuk menghitung jarak total semua rute. Fungsi ini menggunakan parameter rute-rute pada solusi. Nilai total jarak ditunjukkan oleh variabel *distance*. Seperti yang ditunjukkan pada baris 16, nilai variabel distance diperoleh dari hasil penjumlahan semua jarak rute yang dihitung dengan menggunakan fungsi calcDistance. Fungsi calcDistance diimplementasikan pada baris 24 sampai dengan 28. Fungsi ini merupakan implementasi dari rumus eucledian distance. Rumus tersebut digunakan untuk menghitung jarak antar titik simpul berdasarkan nilai koordinat x dan y dari

masing-masing simpul. Rumus eucledian distance telah dijelaskan sebelumnya pada Rumus 3.4.

```
 9  public double calcOF1(ArrayList<Route> rs){
10    ArrayList<Route> routes = rs;
11    int numRs = routes.size();
12    double distance = 0;
13    for(Route r: routes){
14       RouteItem rItem = r.getFirst();
15       while(rItem.getNext()!=null){
16       distance += calcDistance(rItem.getCurrLocation(),
   rItem.getNext().getCurrLocation());
17       rItem = rItem.getNext();
18       }
19    }
20    double value = distance;
21    return value;
22  }
23
24  double calcDistance(Location l1, Location l2){
25     int xdiff = Math.abs(l1.getXCoord()-l2.getXCoord());
26     int ydiff = Math.abs(l1.getYCoord()-l2.getYCoord());
27     return Math.sqrt((xdiff*xdiff)+(ydiff*ydiff));
28  }
```

Kode Program 4. 19 Fungsi calcOF1

Fungsi tujuan kedua dihitung dengan menggunakan rumus simpangan baku. Simpangan baku digunakan untuk mengukur keseimbangan antar jarak rute pada solusi. Rumus simpangan baku diimplementasikan pada fungsi calcOF2 Kode Program 4.20. Fungsi tersebut memiliki parameter yang sama dengan fungsi untuk fungsi tujuan pertama yaitu rute-rute. Perhitungan simpangan baku dilakukan dengan tahapan:

a. Menghitung jarak setiap rute (baris 12 sampai dengan 18). Perhitungannya sama dengan perhitungan pada fungsi objektif pertama.
b. Menghitung total semua rute dengan variabel sum pada baris 19.
c. Menghitung rata-rata jarak rute (baris 23)
d. Mengitung selisih antara jarak rute dan jarak rata-rata rute (baris 25)
e. Menghitung nilai kuadrat selisih (baris 26)
f. Menghitung total nilai kuadrat selisih dengan kode program seperti pada baris 27.
g. Menghitung nilai simpangan baku dengan mengakarkan nilai total kuadrat selisih yang telah dibagi dengan jumlah rute (baris 29)

```
1  public double calcOF2(ArrayList<Route> rs){
2     ArrayList<Route> routes = rs;
3     int numRs = routes.size();
4     double distance = 0.0;
5     double sum = 0.0;
6     double mean =0.0;
7     double selisih =0.0;
8     double kuadratselisih = 0.0;
9     double tot_kuadrat_selisih =0.0;
10    int a =0;
11    double [] tot_jarak_rute = new double [numRs+1];
12    for(Route r: routes){
13        RouteItem rItem = r.getFirst();
14        distance = 0;
15        while(rItem.getNext()!=null&&a<=numRs){
16           distance +=
   calcDistance(rItem.getCurrLocation(),rItem.getNext().getCurrLocation
   ());
17           rItem = rItem.getNext();
18        }
19        sum +=distance;
20        tot_jarak_rute[a] = distance;
21        a++;
22    }
23    mean = sum/numRs;
24    for (int i=0; i<routes.size(); i++){
25        selisih = tot_jarak_rute[i]-mean;
26        kuadratselisih = selisih*selisih;
27        tot_kuadrat_selisih += kuadratselisih;
28    }
29    double SD = Math.sqrt(tot_kuadrat_selisih/numRs);
30    double value = SD;
31    return value;
32  }
```

Kode Program 4. 20 Fungsi calcOF2

4.3.4 Implementasi Metode Hiperheuristik Objektif Jamak

Implementasi metode objektif jamak hiperheuristik dilakukan pada turunan class Hiperheuristik. Detail kode program pada masing-masing bagiannya ditunjukkan pada Kode Program 21 sampai dengan Kode Program 27.

Pada Kode Program 4.21 ditunjukkan library yang digunakan untuk implementasi algoritma hiperheuristik. Baris pertama merupakan nama package yang digunakan untuk menyimpan file hiperheuristik pada HyFlex. Baris kedua adalah untuk pemanggilan fungsi-fungsi pada abstract class *hyperheuristic*. Baris ketiga digunakan untuk memanggil fungsi-fungsi pada abstract class domain permasalahan. Baris keempat digunakan untuk memanggil fungsi untuk menyimpan solusi yang dihasilkan setelah kode program dijalankan. Class solutionx merupakan class yang mendefinisikan solusi yang disimpan pada

TreeSet. Baris kelima merupakan pemanggilan domain permasalahan yang menjadi objek pada class ini. Baris keenam sampai dengan kesembilan merupakan library untuk tipe penyimpanan solusi. Pada class ini digunakan empat jenis tipe penyimpanan data yaitu array list, lis, Sorted Set dan TreeSet. Semuanya digunakan untuk mengelola solusi pareto.

```
1  package Examples;
2  import AbstractClasses.HyperHeuristic;
3  import AbstractClasses.ProblemDomain;
4  import VRP.Solutionx;
5  import VRP.VRP;
6  import java.util.ArrayList;
7  import java.util.List;
8  import java.util.SortedSet;
9  import java.util.TreeSet;
10 /**
11  *
12  * @author Sasmi
13  */
```

Kode Program 4. 21 Fungsi pemanggilan dari kelas lain

Mulai dari Kode program 4.22 merupakan kode program utama pada implementasi hiperheuristik. Pada baris 14 ditunjukkan deklarasi kelas objektif jamak VRP yang merupakan turunan dari kelas Hiperheuristik. Oleh karena itu kelas ini dapat menggunakan semua fungsi yang ada pada kelas Hiperheuristik. Baris 15 digunakan untuk menginisiasi tempat penyimpanan solusi dalam bentuk TreeSet. Digunakan TreeSet karena untuk menyimpan solusi yang unik dan telah diurutkan berdasarkan nilai OF1 atau OF2. Baris 16 dan 17 digunakan untuk membuat objek HHMOVRP2 dengan seed yang acak. Baris 19 adalah fungsi yang digunakan untuk mengimplementasikan algoritma objektif jamak hiperheuristik.

```
14 public class HHMOVRP2 extends HyperHeuristic{
15    SortedSet<Solutionx> SolSet_byF1 = new TreeSet<>();
16    public HHMOVRP2(long seed){
17    super(seed);
18    }
19    public void solve(ProblemDomain problem) {
20    int number_of_heuristics =problem.getNumberOfHeuristics();
21     double old_OF1 =Double.POSITIVE_INFINITY;
22    double old_OF2 =Double.POSITIVE_INFINITY;
23    double old_OF_value=Double.POSITIVE_INFINITY;
24     problem.initialiseSolution(0);
25     int iterasi=1;
26    double [] Solset=new double[iterasi+1];
```

Kode Program 4. 22 variabel-variabel yang dibutuhkan

Baris 20 merupakan kode program untuk memanggil LLH yang akan digunakan. Baris 21 sampai dengan 23 digunakan untuk menginisiasi nilai fungsi

objektif. Tahap pembentukan inisial solusi dilakukan pada kode program baris 24. Inisial solusi disimpan pada memory 0. Variabel iterasi digunakan untuk menghitung jumlah iterasi yang terjadi. variabel Solset bertipe array untuk menyimpan solusi yang dihasilkan pada setiap iterasi dengan ukuran array sesuai dengan jumlah iterasi ditambah dengan 1.

Pada baris 27 dari Kode Program 4.23 dapat diketahui proses mulainya iterasi untuk mencari solusi, iterasi berjalan sampai mencapai parameter batas waktu. Baris 28 dan 29 merupakan inisiasi untuk variabel bobot yang digunakan dalam perhitungan nilai fungsi objektif pada tahap selanjutnya. Nilai bobot diperoleh secara random dengan jumlah w1 dan w2 adalah 1. Baris 30 digunakan untuk memilih secara acak LLH yang akan digunakan untuk memodifikasi solusi. baris 31 digunakan untuk menginisiasi nilai fungsi objektif berdasarkan LLH yang diimplementasikan pada solusi. Baris 32 digunakan untuk mencetak nilai fungsi objektif pertama maupun kedua. Baris 33 dan 34 digunakan untuk menghitung nilai normalisasi dari fungsi objektif pertama dan kedua. Detail fungsi normalisasi ditunjukkan pada fungsi normalisasi Kode Program 4.26. Pengisian parameter nilai minimum dan maksimum disesuaikan dengan hasil uji coba awal pada Subbab 5.2.1. Baris 35 digunakan untuk menghitung nilai fungsi objektif berdasarkan nilai bobot untuk masing-masing fungsi objektif. Pada baris 36 dan 37 dilakukan konversi tipe data double menjadi integer agar memudahkan untuk proses seleksi solusi unik yang disimpan pada TreeSet. Nilai fungsi objektif dari solusi yang baru disimpan pada TreeSet dan diurutkan dari kecil ke besar berdasarkan nilai OF1, dilakukan dengan kode program baris 38. Perhitungan delta pada baris 39 dilakukan untuk menunjang proses move acceptance. Pada Kode Program 4.23 menggunakan move acaceptance Hill Climbing dengan mempertimbangkan probabilitas. Strategi hill climbing dilakukan dengan kode program pada baris 40 sampai dengan 53. Jika nilai delta lebih besar daripada 0 maka solusi baru akan disimpan menggantikan solusi sebelumnya, nilai fungsi objektif yang baru akan menjadi nilai fungsi objektif awal untuk iterasi selanjutnya. Jika nilai delta lebih kecil daripada 0 maka akan ditentukan probabilitasnya secara acak apakah solusi tersebut diterima atau ditolak,

kemungkinannya adalah 50:50. Proses pada baris 28 sampai 53 akan berulang terus menerus sampai batas waktunya habis.

```
27 while (!hasTimeExpired()) {
28   double w1 = Math.random();
29   double w2 = 1-w1;
30   int heuristic_to_apply = rng.nextInt(number_of_heuristics);
31   double []new_obj_function_value =
   problem.applyHeuristics(heuristic_to_apply, 0, 1);
32
   System.out.println(new_obj_function_value[0]+","+new_obj_function_va
   lue[1]);
33    double n_OF1=normalisasi(new_obj_function_value[0],300000,50000);
34            double
   n_OF2=normalisasi(new_obj_function_value[1],500,100);
35            double new_OF_value = (w1* n_OF1)+(w2*n_OF2);
36            int F1_i =toInt(n_OF1);
37            int F2_i =toInt(n_OF2);
38            SolSet_byF1.add(new Solutionx(F1_i,F2_i));
39            double delta = toInt(old_OF_value) -
   toInt(new_OF_value);
40  if (delta > 0) {
41                problem.copySolution(1, 0);
42                old_OF_value = new_OF_value;
43                old_OF1=new_obj_function_value[0];
44                old_OF2=new_obj_function_value[1];
45            }
46        else {
47           if (rng.nextBoolean()) {
48           problem.copySolution(1, 0);
49                 old_OF_value = new_OF_value;
50                 old_OF1=new_obj_function_value[0];
51                 old_OF2=new_obj_function_value[1];
52                 }
53            }
54            iterasi++;
55         }
```

Kode Program 4. 23 Seleksi LLH dan move acceptance Hill Climbing

Kode Program 4.24 merupakan tahapan untuk menyeleksi solusi-solusi yang tidak terdominasi oleh solusi lainnya. Baris 56 adalah deklarasi tempat penyimpanan solusi yang bertipe arraylist. Solusi -solusi yang sebelumnya disimpan dalam bentuk TreeSet dipindahkan ke tempat penyimpanan yang bertipe array list untuk memudahkan dalam pemanggilan setiap elemennya. Baris 56 digunakan untuk mendeklarasikan penyimpanan solusi yang diurutkan berdasarkan OF1, sedangkan baris 57 digunakan untuk mendeklarasikan penyimpanan solusi yang diurutkan berdasarkan OF2. Baris 58 mendeklarasikan tempat penyimpanan solusi pareto yang berbentuk TreeSet. Baris 59 digunakan untuk menginisiasi nilai variabel b. Baris 60 sampai dengan 65 digunakan untuk menyeleksi solusi pareto (tak terdominasi). Selama i lebih kecil dari jumlah elemen solusi pada arraylist

SolSetbyF2 maka jika nilai OF2 dari solusi ke i lebih kecil dari b maka solusi tersebut dimasukkan ke dalam TreeSet pareto. Selanjutnya nilai OF2 dari solusi tersebut akan menjadi nilai b untuk iterasi selanjutnya.

```
56 List<Solutionx> SolSetList = new ArrayList<>(SolSet_byF1);
57         List<Solutionx> SolSet2List = new ArrayList<>(SolSet_byF2);
58          SortedSet<Solutionx> ParetoSet = new TreeSet<>();
59          int b=Integer.MAX_VALUE;
60          for (int i=0;i<SolSet_byF2.size();i++){
61              if(SolSetList.get(i).of2<b||SolSetList.get(i).of2<b){
62                  ParetoSet.add(new
   Solutionx(SolSetList.get(i).of1,SolSetList.get(i).of2));
63                  b=SolSetList.get(i).of2;
64              }
65          }
```

Kode Program 4. 24 Seleksi solusi pareto

Kode Program 4.25 dilakukan untuk menampilkan solusi-solusi pada TreeSet pareto (baris 67 sampai dengan 71) dan mengembalikan nilai asli dari fungsi objektif yang sebelumnya telah dinormaliasi pada baris 73 sampai dengan 79. Sebelum ditampilkan, nilai fungsi objektif dikonversikan ke double terlebih dahulu. Baris 68 digunakan untuk mengonversi nilai OF1 dari integer ke double. Sedangkan baris 69 untuk OF2. Selain itu, proses tersebut juga dilakukan sebelum dilakukan denormalisasi. Fungsi denormalisasi dipanggil secara terpisah antara OF1 dan OF2. Proses denormalisasi secara detail ditunjukkan pada Kode Program 4.26. Denormalisasi dilakukan untuk mengembalikan nilai asli dari masing-masing fungsi objektif sehingga proses ini merupakan kebalikan dari proses normalisasi.

```
66 System.out.println("OF1,OF2");
67     for (Solutionx s:ParetoSet){
68      double x =toDouble(s.of1);
69      double y =toDouble(s.of2);
70              System.out.println(x+","+y);
71          }
72          System.out.println("PARETOSET:X "+ParetoSet.size());
73          for (Solutionx s:ParetoSet){
74              double x =toDouble(s.of1);
75              double y =toDouble(s.of2);
76              double OF1 = denormalisasi(x,300000,50000);
77              double OF2 = denormalisasi(y,500,100);
78              System.out.println(OF1+","+OF2);
79          }
```

Kode Program 4. 25 Menampilkan output solusi

Kode Program 4.26 berisi fungsi-fungsi pendukung yang digunakan dalam proses seleksi solusi maupun proses menampilkan solusi. Kode Program ini terdiri dari 4 fungsi yaitu normalisasi, denormalisasi, toInt dan toDouble. Normalisasi

dilakukan untuk mengonversi suatu nilai menjadi antara nilai 0 sampai dengan 1. Baris 80 sampai dengan 83 merupakan kode program untuk proses normalisasi nilai fungsi objektif. Rumus normalisasi diimplementasikan pada baris 81.

Fungsi yang kedua adalah denormalisasi yang merupakan kebalikan dari normalisasi. Mengubah nilai antara 0 sampai dengan 1 menjadi nilai aslinya. Denormalisasi diimplementasikan pada baris 84 sampai dengan 87. Rumus denormalisasi diimplementasikan pada baris 85.

```
80 private double normalisasi(double x, double max, double min){
81     double norm=(x-min)/(max-min);
82         return norm;
83      }
84      private double denormalisasi(double n, double max, double min){
85     double x=min+(n*(max-min));
86         return x;
87      }
88      int toInt (double val){
89         return (int)(val*100000);
90      }
91      double toDouble (int val){
92         return (double)val/100000;
93      }
```

Kode Program 4. 26 Fungsi-fungsi pendukung

Fungsi ketiga dan keempat merupakan fungsi untuk konversi integer menjadi double dan sebaliknya. Kedua fungsi ini juga berkebalikan. Pada konversi ini digunakan nilai pengali 100000 agar nilai fungsi objektif dapat dibedakan dan dijaring yang unik.

Kode Program 4.27 merupakan implementasi algoritma pembanding dari move acceptance Hill Climbing. Algoritma move acceptance yang digunakan untuk pembanding adalah Great Deluge. Parameter yang digunakan pada algoritma ini adalah Suhu maksimum (Tmax), dan Solusi awal (So). W adalah bobot. To merupakan suku awal sedangkan t adalah suhu pada waktu tertentu. Target merupakan nilai solusi yang menjadi acuan suatu solusi diterima atau tidak. Jika solusi baru lebih baik daripada target solusi pada Tmax, maka nilai target akan berubah. Jika solusi lebih baik daripada target atau lebih baik daripada solusi sebelumnya, maka solusi baru tersebut diterima. Tetapi jika tidak lebih baik daripada keduanya, maka solusi baru tersebut akan ditolak.

```
1  class GreatDelugeAcceptance {
2           private double W;
3           private long To, t, Tmax;
4           private double So, target, Beta;
5           public GreatDelugeAcceptance(long Tmax, double So) {
6                 this.Tmax = Tmax;
7                 this.So = 1.2 * So;
8                 this.To = System.currentTimeMillis();
9                 this.Beta = (So - So * (1 - W)) / Tmax;
10                this.W = 1 / (So + 1);
11          }
12          public boolean accept(double oldSolution, double
    newSolution) {
13                t = System.currentTimeMillis() - To;
14                target = So - Beta * t;
15                if (newSolution <= So * (1 - W)) {
16                      So = newSolution;
17                      Beta = (So - So * (1 - W)) / (Tmax - t);
18                }
19                return newSolution <= target || newSolution <
    oldSolution ? true
20                                  : false;
21          }
22    }
```

Kode Program 4. 27 Move Acceptance Great Deluge

4.3.5 Implementasi Kode Program Grafik dan Indikator Uji Coba pada Python

Pada penelitian ini proses uji coba tidak hanya diimplementasikan pada HyFlex, namun terdapat beberapa tahapan yang diimplementasikan menggunakan Bahasa pemrograman python. Implementasi dilakukan pada perhitungan indikator uji coba dan pembuatan boxplot, swarmplot, histogram dan scatter plot. Kode Program untuk membuat boxplot, swarmplot, histogram dan scatter plot disajikan pada Bagian Lampiran A.

Indikator yang diimplementasikan pada python adalah indikator jumlah solusi pareto dan indikator coverage. Sedangkan indikator hypervolume dihitung dengan menggunakan kerangka kerja JMetal yang diimplementasikan dengan bahasa pemrograman java.

Kode program 4.28 digunakan untuk menghitung nilai coverage dari himpunan solusi yang dihasilkan oleh algoritma Hiperheuristik. Package yang digunakan adalah csv dan sys. Csv digunakan untuk membaca file solusi dengan format .csv. sedangkan package sys digunakan untuk mengakses beberapa variabel yang disediakan oleh python. Baris 4 dan 5 digunakan untuk membaca file solusi.

baris 6 dan 7 adalah untuk menghitung jumlah elemen dari himpunan solusi. variabel counter digunakan untuk menghitung jumlah solusi A yang didominasi oleh solusi B. Pada baris 8 sampai 13 digunakan untuk membandingkan antar solusi dan menjaring solusi yang terdominasi. Baris 14 merupakan rumus untuk menghitung coverage. Baris 15 digunakan untuk menampilkan nilai coverage yang dihasilkan.

```
1  import csv
2  import sys
3
4  dataA = list(csv.reader(open("C:\GD\sol13\pareto\sol131_80.csv")))
5  dataB = list(csv.reader(open("C:\HC\sol13\pareto\sol131_80.csv")))
6  A=len(dataA)-1
7  B=len(dataB)-1
8  counter=0
9  for i in range (1,A+1):
10     for j in range (1,B+1):
11         if dataB[j][0] <= dataA[i][0] and dataB[j][1] <= dataA[i][1]
   :
12             counter+=1
13             break
14 coverageA= counter/A
15 print("COVERAGE =", coverageA)
```

Kode Program 4. 28 Perhitungan nilai coverage

Indikator jumlah solusi pareto dihitung dengan menggunakan kode program pada Kode Program 4.29. package yang digunakan sama dengan yang digunakan untuk menghitung indikator coverage. Baris 3 dan 4 digunakan untuk membaca data solusi. Baris 6 untuk menghitung nilai elemen himpunan solusi digunakan fungsi len yang berarti menghitung panjangnya data yang tersimpan pada file .csv yang telah dibaca pada baris sebelumnya.

```
1  import csv
2  import sys
3  HC51_20 = list(csv.reader(open("datahc\hom51_20.csv")))
4  GD51_20 = list(csv.reader(open("datagd\hom51_20.csv")))
5
6  print(len(HC51_20)-1,",",len(GD51_20)-1)
```

Kode Program 4. 29 Perhitungan jumlah solusi pareto

BAB V
UJI COBA DAN ANALISIS HASIL

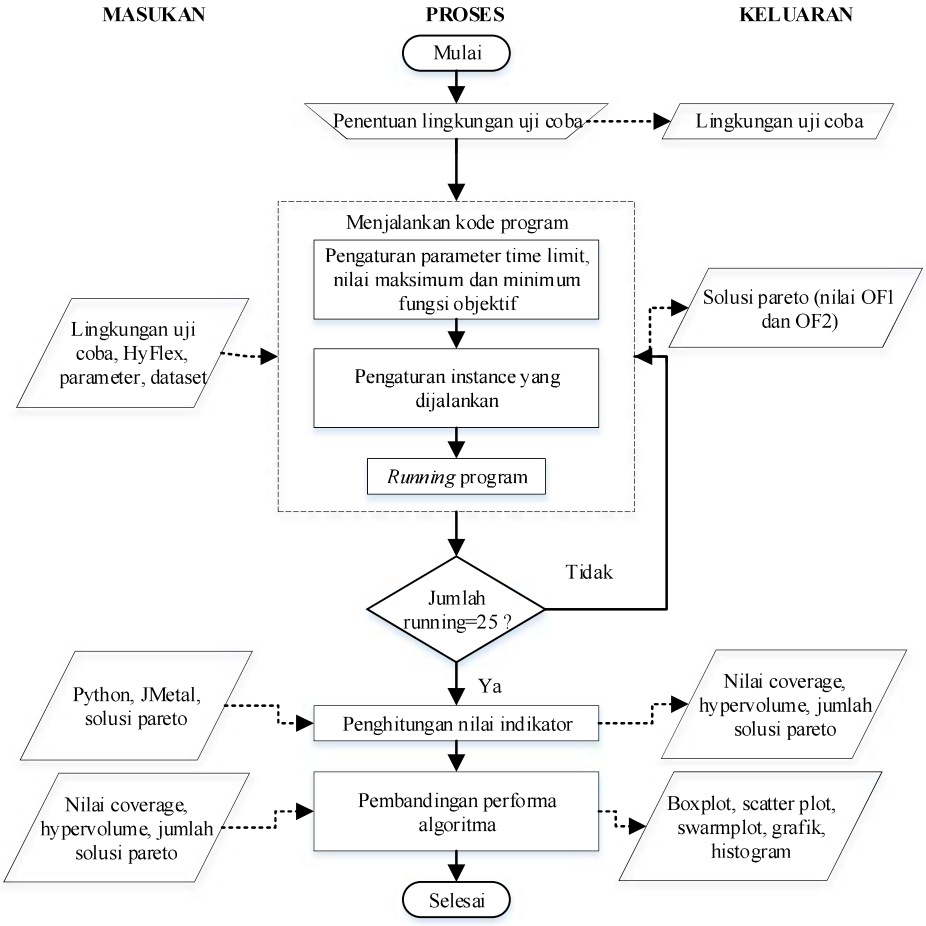
Pada bagian ini dijelaskan mengenai uji coba dan hasil analisis berdasarkan hasil uji coba. Uji coba dilakukan untuk mengukur performa algoritma yang telah diimplementasikan pada bagian sebelumnya. Bagian ini terdiri dari Lingkungan Uji Coba, Hasil Uji Coba dan Analisis Hasil.

5.1. Lingkungan dan Tahapan Uji Coba

Uji Coba dilakukan pada lingkungan uji coba tertentu. Komputer yang digunakan untuk uji coba memiliki spesifikasi prosesor Intel core i5 64bit 2.5 GHz dan Sistem Operasi Windows 10 RAM 4 GB. Perangkat Lunak yang digunakan untuk uji coba adalah Netbeans IDE 8.2 dan Python 3.7.

Tahapan uji coba yang dilakukan pada penelitian ini ditunjukkan pada Gambar 5.1. Uji coba berawal dari penentuan lingkungan uji coba. Kode program hasil implementasi algoritma yang diusulkan dijalankan dengan menggunakan aplikasi Netbeans 8.2. Output dari Netbeans adalah solusi pareto yang direpresentasikan dengan nilai fungsi objektif. Output tersebut menjadi input pada tahapan perhitungan nilai indikator coverage dan jumlah solusi pareto yang dijalankan dengan aplikasi Python 3.7. Masing-masing nilai indikator kemudian digambarkan menjadi boxplot, swarm plot dan histogram yang akan menjadi bahan dalam tahapan analisis hasil. Tidak semua indikator dihitung dengan menggunakan phyton. Khusus untuk indikator hypervolume dihitung menggunakan kerangka kerja JMetal.

Pada penelitian ini batas waktu yang digunakan dalam menjalankan kode program adalah 20, 40, dan 80 detik. Dataset yang digunakan dalam uji coba adalah enam instance dari yang diambil dari dataset gehring homberger dan dataset Solomon seperti yang dijelakan pada Subbab 3.4 Uji Coba. Setiap instance dijalankan 25 kali untuk memberikan statistik yang reliable.



Gambar 5. 1 Tahapan uji coba

5.2. Hasil Uji Coba

Pada penelitian ini dilakukan dua jenis uji coba yaitu uji coba awal dan uji coba inti. Uji coba awal dilakukan untuk menentukan parameter yang digunakan pada tahapan normalisasi. Sedangkan uji coba dilakukan untuk mengukur dan membandingkan performa algoritma yang diusulkan pada penelitian ini. Hasil uji coba awal dan uji coba inti dijabarkan pada subbab 5.2.1 dan 5.2.2.

5.2.1. Hasil Uji Coba Awal

Uji coba awal dilakukan sebagai bentuk persiapan dalam melakukan uji coba inti dari penelitian ini. Melalui proses uji coba awal ini akan dihasilkan parameter nilai minimal dan maksimal fungsi objektif pada masing-masing instance. Uji coba ini menjadi penunjang proses normalisasi. Normalisasi adalah proses penyetaraan

suatu nilai menjadi nilai antara 0 sampai dengan 1. Proses ini dilakukan agar nilai dari fungsi objektif dapat dibandingkan karena jika menggunakan nilai asli maka rentang nilainya terlalu jauh antara nilai fungsi objektif pertama dan kedua. Ketika nilainya sudahnya dikonversi menjadi antara 0 sampai dengan satu maka kedua fungsi objektif tersebut dapat digabungkan dalam satu persamaan setelah dikalikan dengan nilai bobot. Strategi pembobotan digunakan pada penelitian ini untuk menghasilkan solusi yang lebih bervariasi.

Uji coba awal dilakukan dengan tata cara sebagai berikut:

a. Waktu running kode program yaitu 20, 40, dan 80 detik.
b. Pada masing-masing waktu running dan masing-masing instance, kode program dijalankan sebanyak 25 kali untuk mendapat hasil yang sesuai dengan reliabilitas statistik.
c. Setelah didapatkan nilai fungsi objektif pada setiap iterasi maka data diolah dengan menggunakan Microsoft Excel.
d. Digunakan formula Min dan Max untuk mendapatkan nilai minimal dan maksimal untuk masing-masing fungsi objektif.

Hasil uji coba awal dirangkum dalam Tabel 5.1. Berdasarkan Tabel 5.1 diketahui bahwa setiap instance memiliki rentang nilai yang tidak sama. Oleh karena itu parameter nilai minimum dan maksimum diatur pada setiap instance dataset. Namun, terdapat kesamaan nilai minimum OF2 pada instance dataset Solomon yaitu bernilai 0. Pada kode program parameter nilai minimum dan maksimum didefinisikan untuk masing-masing dataset yang digunakan dalam uji coba inti karena nilai parameternya bervariasi untuk setiap instance.

Berdasarkan hasil uji coba awal pada Tabel 5.1 juga dapat diketahui bahwa rentang nilai fungsi objektif untuk solusi dataset Solomon dan dataset Gehring Homberger sangat berbeda dan memiliki rentang yang jauh. Dataset Solomon memiliki nilai fungsi objektif pertama yang berkisar antara satuan sampai dengan ratusan dan nilai fungsi objektif kedua yang berkisar antara puluhan sampai dengan ribuan. Sedangkan dataset Gehring Homberger lebih besar daripada itu yaitu pada rentang antara ratusan sampai dengan ribuan untuk fungsi objektif kedua dan

rentang puluhan ribu sampai dengan ratusan ribu untuk fungsi objektif pertama. Oleh karena itu, perlu dilakukan tahap normalisasi untuk menyeimbangkan nilainya sehingga dapat digabungkan dalam satu persamaan agregasi untuk menghasilkan nilai fungsi objektif gabungan sebagai pertimbangan pada tahap *move acceptance*.

Tabel 5. 1 Hasil Uji Coba Awal

Instance	OF1		OF2	
	Min	Max	Min	Max
Dataset Solomon				
RC207(0)	68	4352	0	288
R101(1)	500	5378	0	62
C208(13)	214	3786	0	308
Dataset Gehring Homberger				
C1_10_1(5)	136170	274948	125	295
RC2_10_1(6)	66275	181342	488	1691
R1_10_1(7)	106051	223859	153	525

Berbeda dengan single objektif, optimasi objektif jamak pada penelitian ini menggunakan nilai hasil persamaan agregasi untuk menentukan penerimaan solusi. Nilai persamaan agregasi dihitung dengan terlebih dahulu mengalikan nilai hasil normalisasi OF1 dan OF2 dengan bobot yang secara periodik berubah nilainya dan ditentukan secara random. Penentuan secara random digunakan agar solusi yang dihasilkan lebih bervariasi sehingga tidak terjebak pada local *optimum.* Nilai bobot apabila dijumlahkan bernilai 1.

5.2.2. Hasil Uji Coba Inti

Uji coba ini dilakukan untuk mengukur performa algoritma Hiperheuristik apabila diimplementasikan pada permasalahan objektif jamak. Berbeda dengan permasalahan single objektif yang menggunakan indikator nilai fungsi objektif untuk mengukur performa algoritma, pada permasalahan objektif jamak digunakan indikator coverage, jumlah solusi pareto dan hypervolume sebagai ukuran dalam menilai performa suatu algoritma dibandingkan dengan algoritma yang lainnya. Penjelasan detail terkait konsep indikator performa dijelaskan pada Subbab 2.16.

Pada uji coba ini dibandingkan performa algoritma Hiperheuristik Hill Climbing dan algoritma Hiperheuristik Great Deluge. Uji coba ini bertujuan untuk

mengetahui komposisi algoritma seleksi LLH dan move acceptance yang paling sesuai untuk berkolaborasi dengan modifikasi algoritma Hiperheuristik dalam menyelesaikan permasalahan objektif jamak.

Tabel 5.2 menunjukkan hasil uji coba pada algoritma HC dan GD. Hasil uji coba ini berupa hasil rekapitulasi rata-rata dari masing-masing indikator nilai coverage, nilai hypervolume dan jumlah solusi pareto. Hasil uji coba masing indikator yang didapatkan pada setiap menjalankan kode program disajikan pada bagian Lampiran B, C dan D.

Pada Tabel 5.2 nilai indikator yang lebih baik ditandai dengan angka yang penulisannya ditebalkan. Berdasarkan Tabel 5.2 dapat diketahui bahwa pada indikator jumlah solusi pareto, algoritma GD unggul pada 15 nilai. Sedangkan algoritma GD hanya unggul pada 3 nilai yaitu hasil dari instance 5 dengan waktu 40 detik, instance 6 dengan waktu 20 detik dan 40 detik. Semakin banyak solusi pareto yang dihasilkan maka semakin baik performa algoritma tersebut.

Pada indikator nilai coverage, semakin kecil nilainya maka performa algoritma akan semakin baik. Nilai 0 menunjukkan bahwa tidak ada solusi dari suatu algoritma yang didominasi oleh algoritma lain. Sedangkan nilai 1 menunjukkan bahwa semua solusi pada suatu algoritma didominasi oleh algoritma lainnya. Hasil uji coba menunjukkan bahwa nilai coverage algoritma HC lebih kecil pada semua instance dan waktu running dibandingkan dngan nilai coverage yang dihasilkan oleh algoritma GD.

Enterpretasi nilai hypervolume berbanding terbalik dengan nilai coverage. Semakin besar nilai hypervolume maka performa algoritma akan semakin baik. Berdasarkan Tabel 5.2 terlihat bahwa HC lebih unggul dibandingkan dengan GD pada semua instance dan waktu running. Nilai hypervolume 0 menunjukkan bahwa tidak ada solusi dari suatu algoritma yang nilainya mendekati true front. Hal ini menyebakab nilai volum dari cakupan solusinya bernilai 0.

Tabel 5. 2 Rekapitulasi Hasil Uji Coba

Dataset_Time Limit	Avg. No of fronts		Coverage Value		Hypervolume	
	HC	***GD***	***HC***	***GD***	***HC***	***GD***
0_20	8.32	**28.4**	**0**	0.996923077	**0.765553**	0
0_40	8.48	**27.84**	**0**	0.956069	**0.828296**	0
0_80	9.44	**29.28**	**0**	0.995897	**0.760763**	0
1_20	13.88	**25.36**	**0.009333**	0.987273	**0.327111**	0.023684
1_40	11.84	**33.56**	**0**	1	**0.206804**	0.006802
1_80	11.48	**42.84**	**0**	1	**0.264396**	0.010199
13_20	5.88	**28.44**	**0.04**	0.872583	**0.370493**	0
13_40	4.68	**30.96**	**0**	0.939478	**0.42448**	0
13_80	5.84	**28.72**	**0**	0.933932	**0.390252**	0
5_20	14.2	**16.04**	**0.079821**	0.83513	**0.58042**	0.230166
5_40	**18.48**	17.8	**0.156394**	0.761872	**0.629088**	0.270498
5_80	16.96	**17.2**	**0.103841**	0.83828	**0.564754**	0.272206
6_20	**14.2**	12.36	**0.173811**	0.320657	**0.68019**	0.590525
6_40	**13.68**	12.48	**0.294101**	0.340387	**0.587547**	0.546457
6_80	13.96	**15.92**	**0.170128**	0.339509	**0.649463**	0.543274
7_20	17.04	**18.16**	**0.00401**	0.809147	**0.58592**	0.080776
7_40	15.48	**19.88**	**0.006667**	0.793838	**0.626925**	0.019294
7_80	19.84	**21.88**	**0.004444**	0.884258	**0.569061**	0.015103

5.3. Hasil Validasi dan Verifikasi

Validasi merupakan tahapan yang dilakukan untuk membuktikan bahwa algoritma yang diusulkan pada penelitian ini menghasilkan solusi yang memiliki jarak antar rute yang lebih seimbang. Validasi dilakukan dengan membandingkan hasil ouput kode program antara hiperheuristik VRP yang menerapkan fungsi objektif tunggal dengan hiperheuristik VRP yang menerapkan objektif jamak. Pada penelitian ini digunakan salah satu instance untuk uji validasi ini yaitu instance 13. Kode program dijalankan dengan tiga macam time limit seperti pada tahapan uji coba dan dijalankan sebanyak 25 kali pada setiap time limit tersebut. Kemudian dilakukan rekapitulasi dan dihitung nilai minimum, maksimum, rata-rata dan simpangan baku dari masing-masing nilai fungsi objektif.

Hasil validasi disajikan pada Tabel di Lampiran E. Berdasarkan hasil validasi tersebut dapat diketahui bahwa implementasi algoritma Hiperheuristik objektif jamak mampu menghasilkan solusi yang memiliki jarak antar rute yang lebih seimbangan dibandingkan dengan solusi yang dihasilkan oleh algoritma hiperheuristik untuk fungsi objektif tunggal. Hal ini dibuktikan dengan nilai simpangan baku solusi objektif jamak yang paling kecil dibandingkan dengan solusi single objektif pada semua hasil running. Sebaliknya, nilai total jarak rute yang paling minimal sebagian besar dihasilkan oleh algoritma yang menerapkan single objektif. Hal ini juga membuktikan bahwa tidak mudah memenuhi kedua fungsi objektif agarnya kedua minimal. Oleh karena itu pada optimasi objektif jamak digunakan pendekatan pareto untuk menemukan kompromi antar solusi yang dihasilkan.

Tahapan selanjutnya adalah Verifikasi. Tahapan ini dilakukan untuk membuktikan bahwa output yang dihasilkan dari kode program sudah tepat perhitungannya. Untuk membuktikan hal tersebut maka dibandingkan nilai fungsi objektif dan nilai masing-masing rute antara output hasil running kode program dan hasil perhitungan manual.

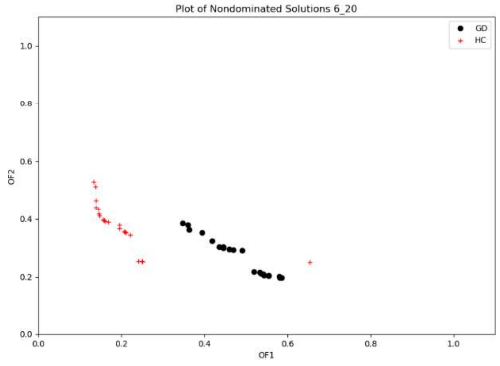
Contoh hasil running kode program dan hasil perhitungan manual disajikan pada table di Lampiran F. Berdasarkan hasil tersebut dapat diketahui bahwa perhitungannya sudah tepat dan sesuai dengan rumus dan hasil perhitungan manual. Nilai-nilai yang akan dibuktikan dengan perhitungan manual ditandai dengan warna kuning dapat dilihat pada Lampiran F.

5.4.Analisis Hasil

Pada bagian ini akan dijabarkan hasil analisis berdasarkan hasil uji coba yang telah disajikan sebelumnya. Bagian ini terdiri dari dua Subbab yaitu hasil analisis perbandingan algoritma Hiperheuristik GD dan HC serta hasil analisis perbandingan solusi yang dihasilkan dari algoritma single objektif dan objektif jamak.

5.4.1. Perbandingan Algoritma Hiperheuristik HC dan GD

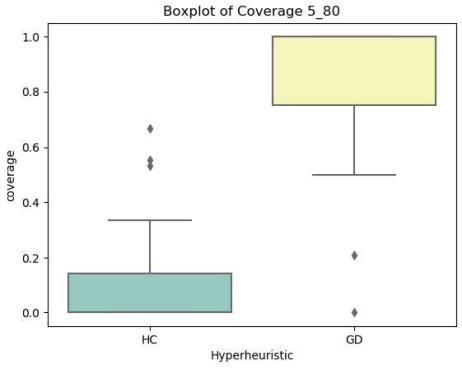
Setelah proses normalisasi, nilai fungsi objektif yang dihasilkan dari masing-masing solusi adalah antara 0 dan 1. Nilai ini adalah hasil akhir dari program yang berjalan dari usulan algoritma yang diimplementasikan di Java. Langkah selanjutnya adalah membandingkan keluaran penerimaan langkah GD dan HC. Pada Gambar. 5.2 disajikan perbandingan scatterplot antara solusi Pareto yang diproduksi oleh HC dan GD.



Gambar 5. 2 Scatterplot perbandingan HC dan GD

Hasil indikator coverage juga disajikan dalam boxplot, swarm plot, dan histogram. Ini untuk membuatnya lebih mudah untuk membandingkan kedua algoritma dan menganalisis hasilnya. Pada dasarnya, ketiga tipe representasi data berguna untuk mengukur keragaman data dan distribusinya.

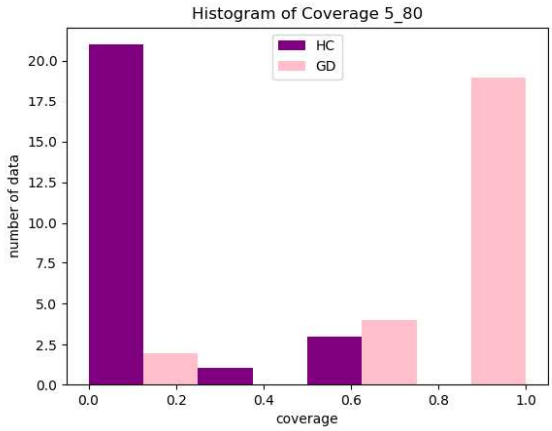
Gambar. 5.3 menunjukkan boxplot nilai coverage. Boxplot biru adalah hasil HC dan boxplot kuning adalah hasil GD. Berdasarkan hasil yang ditunjukkan pada Gambar.5.3 dapat dilihat bahwa GD memiliki rentang nilai yang lebih besar daripada HC. Ini berarti bahwa GD memiliki hasil yang lebih beragam dibandingkan dengan HC. Tetapi meskipun lebih beragam, GD memiliki nilai cakupan rata-rata yang lebih besar dari HC. Ini berarti HC memiliki lebih banyak solusi yang tidak didominasi daripada GD.



Gambar 5. 3 Boxplot perbandingan HC dan GD

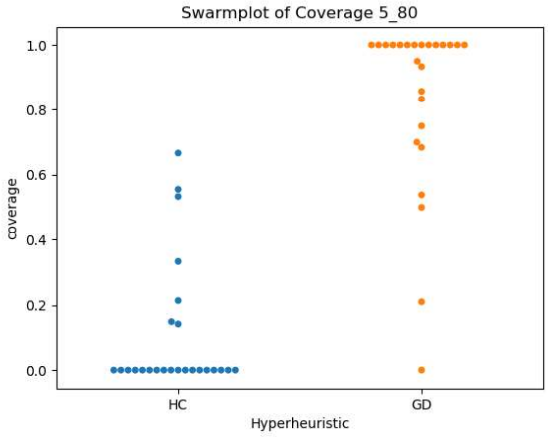
Swarm plot disajikan pada Gambar. 5.4. Distribusi data diwakili oleh suatu titik sehingga posisi setiap datum dapat diketahui. Berdasarkan Gambar. 5.4 dapat

dilihat bahwa sebagian besar nilai coverage HC adalah 0, sedangkan nilai coverage yang mendominasi pada GD adalah 1. Ini berarti bahwa solusi pareto HC mendominasi sepenuhnya solusi pareto GD di sebagian besar instance dan *time limit*.



Gambar 5. 4 Histogram perbandingan HC dan GD

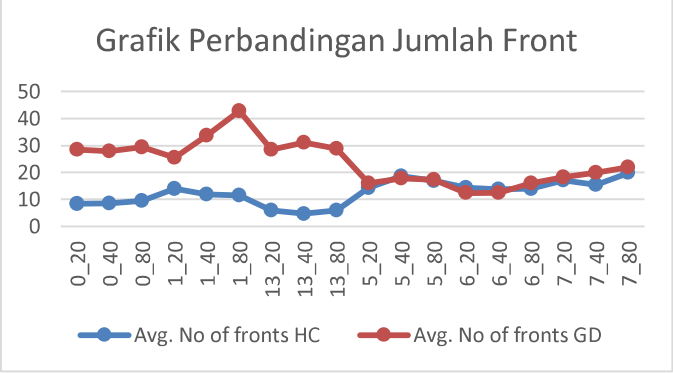
Grafik coverage dengan menggunakan histogram ditunjukkan pada Gambar. 5.5. Berdasarkan hasil ini diketahui bahwa dalam kasus, HC memberikan hasil yang unggul dibandingkan dengan GD. Nilai HC cakupan tertinggi adalah di angka 0, sedangkan untuk GD tertinggi adalah 1.



Gambar 5. 5 Swarmplot perbadingan HC dan GD

Pada penjelasan selanjutnya akan dijabarkan hasil analisis kinerja pada masing-masing indikator uji coba. Grafik perbandingan pada setiap indikator disajikan pada Gambar 5.6, Gambar 5.7 dan Gambar 5.8.

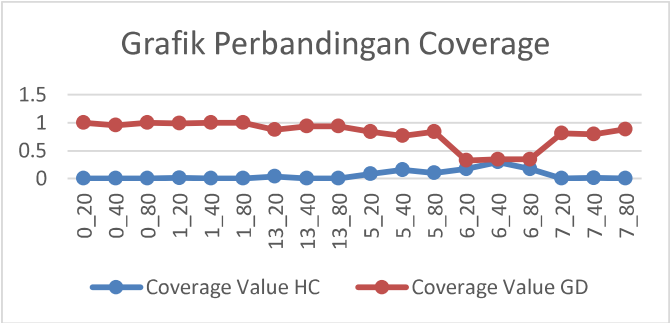
Pada Gambar 5.6 ditunjukkan grafik perbandingan nilai rata-rata jumlah solusi pareto yang dihasilkan oleh algoritma HC dan GD. Berdasarkan Gambar 5.6 dapat diketahui bahwa GD rata-rata memiliki lebih banyak jumlah solusi pareto dibandingkan dengan HC. Namun hal tersebut hanya berlaku pada instance 0,1, dan 13 yang merupakan bagian dari dataset Solomon. Sedangkan pada instance dataset Gehring Homberger kedua algoritma tersebut memiliki jumlah solusi pareto yang berimbang dan saling bersaing. Perbedaannya tidak terlalu terlihat dibandingkan dengan dataset Solomon. Apabila ditinjau berdasarkan kompleksitas dataset yaitu antara instance R (random) untuk instance 1 dan 7, RC (random clustered) untuk instance 0 dan 6 dan C (clustered) untuk instance 13 dan 5, hasilnya tidak terlihat adanya pola nilai jumlah solusi yang dipengaruhi oleh kompleksitas instance dataset. Indikator ini lebih cenderung dipengaruhi oleh perbedaan dataset. Berdasarkan nilai jumlah solusinya, GD tetap lebih unggul dibandingkan dengan HC. Oleh karena itu dapat disimpulkan bahwa berdasarkan indikator jumlah solusi pareto, GD lebih baik dibanding HC. Hal ini terjadi karena solusi yang dihasilkan oleh GD lebih bervariasi dibandingkan dengan solusi HC.



Gambar 5. 6 Grafik Perbandingan Jumlah Front

Pada Gambar 5.7 dapat ketahui perbandingan nilai coverage antara solusi GD dan HC yang disajikan dengan grafik. Coverage merupakan indikator yang mengukur tingkat dominasi algoritma lain terhadap suatu algoritma sehingga

semakin kecil nilai coverage maka semakin baik performa algoritma tersebut. Hal ini membuktikan bahwa algoritma tersebut memiliki sedikit solusi yang terdominasi oleh solusi dari algoritma yang lainnya.

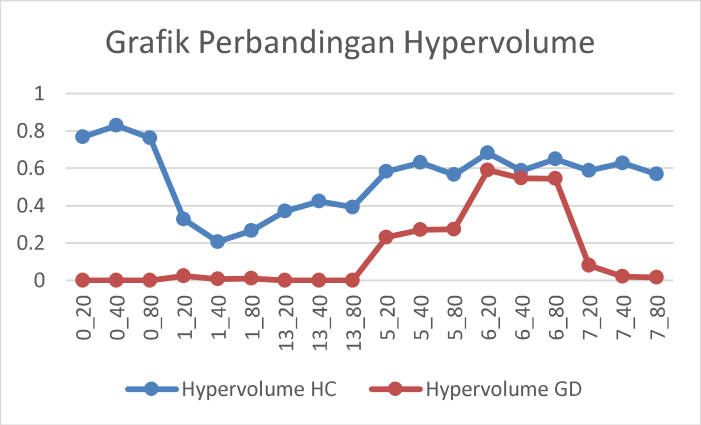


Gambar 5. 7 Grafik Perbandingan Nilai Coverage

Berdasarkan Gambar 5.7 diketahui bahwa HC memiliki nilai coverage yang lebih kecil pada semua skenario uji coba. Hanya pada instance 6 yang nilainya hampir sama dengan GD. Hal ini menunjukkan bahwa solusi yang dihasilkan oleh HC memiliki nilai fungsi objektif yang lebih minimal daripada fungsi objektif dari solusi yang dihasilkan oleh GD. Hal ini karena dibuktikan dengan banyaknya solusi GD yang didominasi oleh HC. Bahkan pada beberapa instance diketahui bahwa solusi GD didominasi penuh oleh HC. Dominasi penuh ditandai dengan nilai coverage yang bernilai 1, sedangkan apabila mendominasi secara penuh maka nilai coverage solusinya adalah 0. Apabila ditinjau berdasarkan kompleksitas dataset yaitu antara instance R (random) untuk instance 1 dan 7, RC (random clustered) untuk instance 0 dan 6 dan C (clustered) untuk instance 13 dan 5, hasilnya tidak terlihat adanya pola nilai coverage yang dipengaruhi oleh kompleksitas instance dataset.

Perbandingan nilai hypervolume antara GD dan HC ditunjukkan pada Gambar 5.8. Apabila ditinjau berdasarkan kompleksitas dataset yaitu antara instance R (random) untuk instance 1 dan 7, RC (random clustered) untuk instance 0 dan 6 dan C (clustered) untuk instance 13 dan 5, hasilnya terlihat adanya pola nilai coverage yang dipengaruhi oleh kompleksitas instance dataset. Nilai hypervolume berubah secara drastis untuk setiap instance yang berbeda. Hal tersebut terutama pada dataset Gehring Homberger untuk algoritma Hiperheuristik

GD. Pada Grafik tersebut terlihat bahwa GD memiliki nilai hypervolume yang lebih kecil dibandingkan dengan HC. Hal ini berarti bahwa HC memiliki volume daerah solusi yang lebih luas dibandingkan dengan GD sehingga dapat disimpulkan bahwa performa HC berdasarkan indikator hypervolume lebih baik dibandingkan dengan GD. Sama seperti indikator coverage, pada indikator ini nilai hypervolume GD dan HC untuk instance 6 memiliki nilai yang hampir sama. Namun sangat berlawanan pada instance yang lainnya.



Gambar 5. 8 Grafik Perbandingan Nilai Hypervolume

Berdasarkan hasil eksperimen yang dijelaskan sebelumnya, dapat disimpulkan bahwa HC dapat memberikan solusi yang lebih optimal dibandingkan dengan GD. Ini merupakan indikasi bahwa HC lebih cocok jika digunakan untuk menyelesaikan masalah objektif jamak. Selanjutnya, berdasarkan indikator jumlah solusi pareto, dapat dilihat bahwa GD menghasilkan lebih banyak front Pareto daripada HC. Hasil ini berbanding terbalik dengan indikator coverage. Oleh karena itu, dapat dilihat bahwa semakin banyak solusi Pareto tidak menjamin kualitas solusi akan lebih baik karena belum tentu nilai coverage dan hypervolumenya lebih kecil. Jadi dapat disimpulkan bahwa ketiga indikator tersebut tidak memiliki hubungan dan pengaruh satu sama lain. Namun antara coverage dan Hypervolume saling berkebalikan atau memiliki hubungan saling berlawanan. Ketika suatu solusi memiliki nilai coverage kecil maka sudah dapat dipastikan nilai hypervolumenya akan besar. Keduanya menandakan bahwa algoritma HC secara umum lebih baik

dibandingkan GD jika digunakan untuk menyelesaikan permasalahan objektif jamak.

Selanjutnya perlu ditinjau performa setiap Algoritma Hiperheuristik HC maupun GD dalam menyelesaikan masing-masing instance dataset. Hal ini bertujuan untuk mengetahui tingkat generalitas algoritma dan performanya dalam menghasilkan solusi pada dataset dengan kompleksitas yang berbeda-beda.

Berdasarkan Gambar 5.6 indikator jumlah solusi pareto, antara GD dan HC menghasilkan solusi dengan jumlah yang berselisih cukup banyak pada instance 0,1 dan 13 (dataset Solomon) yaitu antara 11 sampai dengan 26 solusi. Sedangkan pada dataset Gehring Homberger, keduanya memberikan nilai jumlah solusi pareto yang selisihnya cenderung sedikit yaitu antara 1 sampai dengan 2 solusi. Sehingga dapat diketahui bahwa berdasarkan nilai jumlah solusi pareto, performa kedua algoritma saling berlawanan untuk dataset yang sederhana dan performanya hampir sama untuk dataset yang lebih kompleks.

Berdasarkan indikator coverage, performa GD dan HC cenderung stabil karena grafik yang disajikan pada Gambar 5.7 menunjukkan perubahan nilai coverage yang tidak fluktuatif kecuali pada instance 6. Pada instance tersebut performa keduanya hampir sama, hanya berselisih antara 0,05 sampai dengan 0,17. Nilai itu lebih kecil dibandingkan dengan selisih nilai coverage antara GD dan HC pada instance lainnya yaitu antara 0,6 sampai dengan 1. Hal ini menunjukkan bahwa performa kedua algoritma HC dan GD tidak dipengaruhi oleh kompleksitas dataset jika didasarkan pada indikator coverage. Namun performa indikator coverage yang lebih baik ditunjukkan oleh algoritma HC.

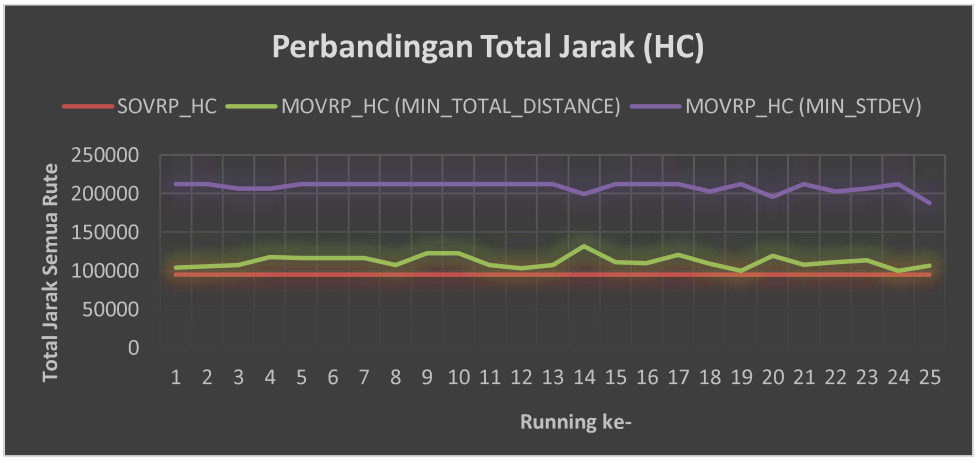
Apabila didasarkan pada nilai hypervolume, maka performa GD lebih stabil untuk dataset Solomon yang kompleksitasnya rendah. Sebaliknya, performa HC stabil pada dataset yang kompleks. Berdasarkan nilai hypervolume, performa HC dan GD hanya stabil pada dataset yang memiliki kompleksitas tertentu. Namun khusus pada dataset 6, kedua algoritma tersebut memiliki selisih rata-rata nilai coverage yang relatif kecil yaitu antara 0,04 sampai dengan 0.1 sehingga

performanya hampir sama. Sedangkan pada dataset yang lainnya, HC lebih unggul dibandingkan dengan GD.

5.4.2 Perbandingan Solusi Single Objektif dan Objektif Jamak

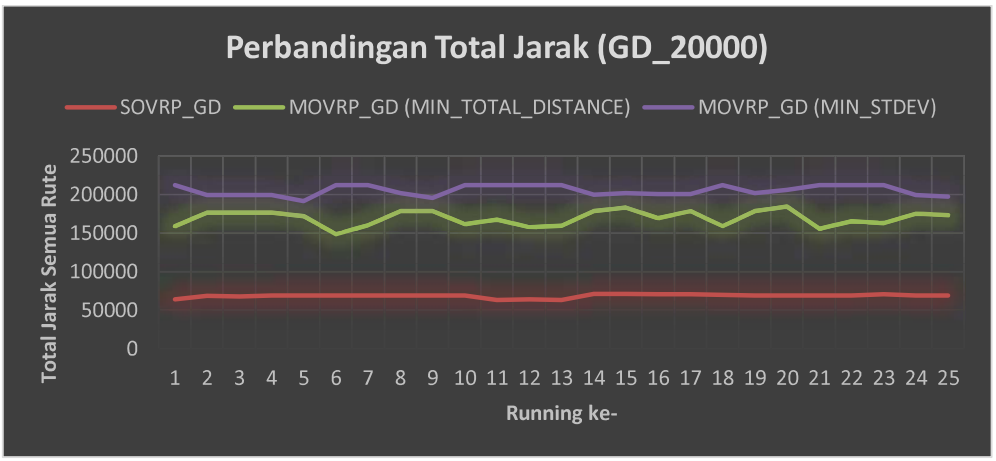
Analisis selanjutnya dilakukan pada perbandingan nilai fungsi objektif dari solusi single objektif dan objektif jamak. Hasil running secara detail disajikan di Lampiran E. Hasil Perbandingan disajikan pada Gambar 5.9, Gambar 5.10, Gambar 5.11 dan Gambar 5.12.

Pada Gambar 5.9 ditampilkan perbandingan nilai jarak total antara solusi single objektif dan objektif jamak. Solusi-solusi tersebut dihasilkan dari implementasi algoritma Hiperheuristik HC. Berdasarkan Gambar 5.9 dapat diketahui bahwa solusi single objektif (warna merah) memiliki nilai jarak total yang paling minimal dibandingkan dengan jarak total solusi objektif jamak. Namun selisih tidak terlalu apabila dibandingkan dengan solusi objektif jamak yang memiliki jarak total minimal (warna hijau muda). Apabila dibandingkan dengan solusi objektif jamak yang memiliki simpangan baku minimal, selisihnya sangat jauh. Solusi objektif jamak yang memiliki nilai simpangan baku yang kecil, pada umumnya akan memiliki jarak total yang besar sehingga dapat disimpulkan bahwa dalam memenuhi fungsi tujuan meminimalkan jarak total, solusi single objektif lebih optimal.



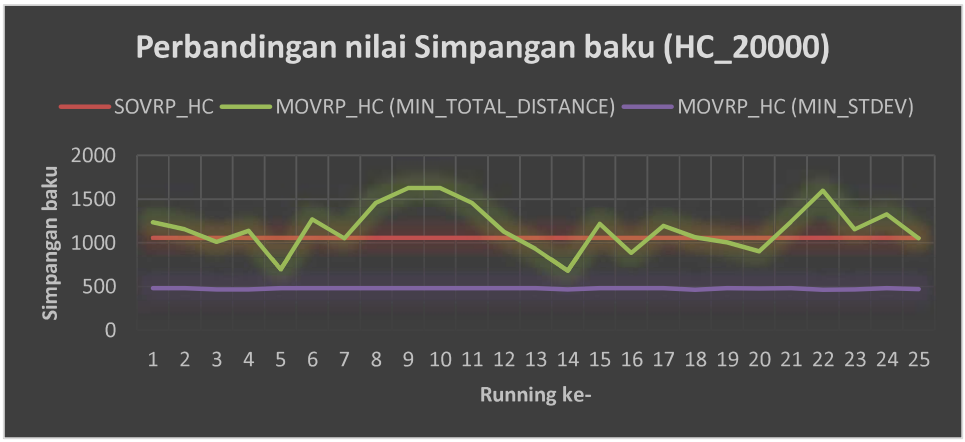
Gambar 5. 9 Grafik Perbandingan Single Objektif dan Objektif jamak berdasarkan Jarak Total menggunakan algoritma HC

Selanjutnya adalah perbandingan nilai jarak total antar solusi single obektif dan objektif jamak yang dihasilkan dari algoritma Hiperheuristik GD. Berbeda dengan hasil pada Gambar 5.9, pada Gambar 5.10 dapat diketahui bahwa nilai jarak total solusi objektif jamak lebih besar dengan selisih yang cukup jauh dengan jarak total dari solusi single objektif. Solusi single objektif tetap memiliki nilai jarak total yang paling minimal jika dibandingkan dengan jarak total solusi objektif jamak.



Gambar 5. 10 Grafik Perbandingan Single Objektif dan Objektif jamak berdasarkan Jarak Total menggunakan algoritma GD

Nilai simpangan baku merupakan nilai yang menggambarkan keseimbangan jarak antar rute dalam setiap solusi. Pada Gambar 5.11 dan Gambar 5.12 disajikan grafik yang membandingkan nilai simpangan baku antara solusi single objektif dan objektif jamak yang dihasilkan oleh algoritma HC dan GD.



Gambar 5. 11 Grafik Perbandingan Single Objektif dan Objektif jamak berdasarkan Simpangan baku menggunakan algoritma HC

Kedua gambar tersebut memberikan kesimpulan yang hampir sama bahwa solusi objektif jamak memiliki nilai simpangan baku yang lebih kecil dibandingkan dengan solusi single objektif. Nilai simpangan baku yang kecil menunjukkan bahwa jarak antar rute pada solusi objektif jamak lebih seimbang dan lebih adil. Hal ini memenuhi fungsi objektif kedua yang digunakan pada penelitian ini. Oleh karena itu dapat disimpulkan bahwa dari sisi nilai simpangan baku, solusi objektif jamak memiliki solusi yang lebih optimal. Selain itu juga membuktikan bahwa algoritma objektif jamak Hiperheuristik mampu menghasilkan solusi VRP yang lebih seimbang jarak antar rutenya.

Gambar 5. 12 Grafik Perbandingan Single Objektif dan Objektif jamak berdasarkan Simpangan baku menggunakan algoritma GD

BAB VI
KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dan saran terkait pengerjaan penelitian ini. Dalam bagian kesimpulan diberikan beberapa poin penting kesimpulan dari hasil penelitian, sedangkan dalam bagian saran diberikan beberapa hal yang berguna untuk kemungkinan pengembangan penelitian lebih lanjut.

6.1 Kesimpulan

Berdasarkan hasil uji coba dan analisis hasil yang telah dilakukan, dapat diperoleh beberapa kesimpulan sebagai berikut:

a. Dalam penelitian ini telah berhasil dilakukan pengembangan algoritma hiperheuristik untuk menyelesaikan permasalahan *Vehicle Routing Problem* (VRP) dari yang semula hanya mempunyai objektif tunggal (meminimumkan total jarak rute kendaraan) menjadi penyelsaian permasalahan dengan objektif jamak. Pengembangan objektif jamak dilakukan dengan menambahkan fungsi objektif yang mempertimbangkan keseimbangan jarak antar rute kendaraan yang didasarkan pada pengukuran simpangan baku terhadap jarak antar rute yang diperoleh. Namun pada penelitian ini masih belum menerapkan batasan jumlah rute dalam setiap solusi, sehingga setiap solusi yang dihasilkan memiliki jumlah rute yang berbeda-beda dan tidak terbatas. Hal tersebut mempengaruhi nilai total jarak semua rute dari setiap solusi VRP dengan objektif jamak.
b. Dalam menentukan algoritma yang digunakan pada tahapan perbandingan solusi VRP dengan objektif jamak (VRPOJ) terhadap solusi VRP dengan objektif tunggal (VRPOT) didasarkan pada hasil uji coba kinerja kombinasi algoritma Hiperheuristik Hill Climbing (HC) dan algoritma Hiperheuristik Great Deluge (GD). Berdasarkan hasil uji coba tersebut maka dapat ditarik kesimpulan berikut:
    1) Berdasarkan indikator jumlah solusi pareto, GD memberikan hasil yang lebih baik dengan rerata jumlah solusi pareto yang lebih banyak (sebesar 24 solusi) dibandingkan dengan HC (sebesar 12 solusi).

2) Berdasarkan indikator nilai *coverage*, HC menghasilkan rerata nilai coverage yang lebih kecil (sebesar 0,06) daripada GD (sebesar 0.8). Oleh karena itu, berdasarkan indikator ini HC menghasilkan solusi-solusi yang lebih optimal daripada GD.
3) Berdasarkan nilai *hypervolume*, HC menghasilkan rerata nilai hypervolume yang lebih besar (sebesar 0.56) daripada GD (sebesar 0.1) sehingga HC memiliki cakupan volume daerah solusi yang lebih luas dibandingkan dengan GD.
4) Hasil uji coba secara umum memberikan indikasi bahwa HC merupakan kombinasi dengan kinerja yang lebih baik apabila digunakan untuk menyelesaikan permasalahan VRP objektif jamak dibandingkan dengan GD sehingga kombinasi algoritma Hiperheuristik Hill Climbing (HC) akan dipakai pada uji coba perbandingan antara solusi VRP dengan objektif jamak (VRPOJ) terhadap solusi VRP dengan objektif tunggal (VRPOT).

c. Berdasarkan hasil uji coba perbandingan antara solusi VRP dengan objektif jamak (VRPOJ) terhadap solusi VRP dengan objektif tunggal (VRPOT) menggunakan set data Solomon (yang dijalankan sebanyak 25 kali) menunjukkan bahwa rerata simpangan baku jarak antar rute untuk VRPOJ (sebesar 678) cukup jauh lebih rendah dibandingkan rerata simpangan baku antar rute VRPOT (sebesar 1.053), walaupun rerata total jarak minimum yang dihasilkan oleh VRPOJ (sebesar 99.590) lebih besar dibandingkan dengan yang dihasilkan oleh VRPOT (sebesar 94.650). Hal ini menunjukkan bahwa tambahan fungsi objektif untuk menyeimbangkan jarak antar rute kendaraan dari solusi VRP yang dihasilkan sesuai dengan tujuan penelitian.

6.2 Saran

Saran dari penelitian ini terdiri dari dua macam yaitu saran untuk penelitian selanjutnya dan saran untuk manajemen calon pengguna.

6.2.1 Saran untuk Penelitian Selanjutnya

Berdasarkan hasil penelitian, seperti dijelaskan pada kesimpulan poin a bahwa pengembangan permasalahan VRP dengan objektif jamak pada penelitian ini masih memperbolehkan batasan jumlah rute untuk dilanggar. Padahal batasan ini dapat mempengaruhi nilai total jarak rute pada setiap solusi yang dibuktikan pada kesimpulan poin c. Solusi yang dihasilkan oleh VRP dengan objektif jamak memiliki nilai simpangan baku yang lebih kecil tetapi nilai total jarak semua rutenya lebih besar dibandingkan dengan solusi VRP dengan objektif tunggal. Berdasarkan hasil analisis, hal tersebut terjadi akibat jumlah rute dari solusi VRP dengan objektif jamak yang lebih banyak sehingga membutuhkan jarak rute yang lebih panjang karena setiap rute membutuhkan jarak tempuh dari dan ke depot yang pada akhirnya mempengaruhi total jarak rutenya. Oleh karena itu, kemungkinan pengembangan perbaikan yang dapat dilakukan pada penelitian selanjutnya adalah menambahkan batasan jumlah rute agar solusi VRP dengan objektif jamak yang dihasilkan dapat lebih optimal pada kedua fungsi objektif.

6.2.2 Saran untuk Manajemen Calon Pengguna

Selain saran bagi penelitian selanjutnya, juga terdapat saran untuk implementasi praktis. Bagi perusahaan logistik dalam menentukan rute pengiriman perlu mempertimbangkan keseimbangan jarak antar rute yang harus ditempuh oleh petugas pengiriman. Hal tersebut berkaitan dengan keadilan gaji dan kesejahteraan petugas pengiriman atau sopir. Pada studi kasus apabila gaji dibayarkan berdasarkan hari kerja, maka sopir yang mendapatkan jatah rute yang panjang maupun yang pendek dalam waktu sehari akan mendapatkan gaji yang sama. Padahal usaha yang dilakukan berbeda sehingga tidak adil bagi para sopir yang mendapatkan tugas rute yang panjang. Di sisi lain, apabila gaji diberikan berdasarkan jarak rute yang ditugaskan, maka akan menyebabkan ketidakadilan bagi para sopir yang mendapatkan tugas jarak rute yang pendek. Oleh karena itu keseimbangan jarak rute akan menjadi salah satu alternatif untuk menciptakan keadilan terhadap para petugas pengiriman atau sopir. Hasil dari penelitian ini dapat digunakan acuan oleh praktisi perusahaan terutama

dalam bidang logistik untuk pengambilan keputusan terhadap rute beberapa kendaraan yang lebih optimal dengan GAP beban kerja yang lebih minimal antar kendaraan sehingga didapatkan pembagian rute yang lebih adil.

DAFTAR PUSTAKA

Baños, R., Ortega, J., Gil, C., Fernández, A., de Toro, F., 2013a. A Simulated Annealing-based parallel multi-objective approach to vehicle routing problems with time windows. Expert Syst. Appl. 40, 1696–1707. https://doi.org/10.1016/j.eswa.2012.09.012

Baños, R., Ortega, J., Gil, C., Fernández, A., de Toro, F., 2013b. A Simulated Annealing-based parallel multi-objective approach to vehicle routing problems with time windows. Expert Syst. Appl. 40, 1696–1707. https://doi.org/10.1016/j.eswa.2012.09.012

Baños, R., Ortega, J., Gil, C., Márquez, A.L., de Toro, F., 2013c. A hybrid meta-heuristic for multi-objective vehicle routing problems with time windows. Comput. Ind. Eng. 65, 286–296. https://doi.org/10.1016/j.cie.2013.01.007

Baños, R., Ortega, J., Gil, C., Márquez, A.L., de Toro, F., 2013d. A hybrid meta-heuristic for multi-objective vehicle routing problems with time windows. Comput. Ind. Eng. 65, 286–296. https://doi.org/10.1016/j.cie.2013.01.007

Baran, B., Schaerer, M., 2003. A Multiobjective Ant Colony System for Vehicle Routing Problem with Time Windows. Appl. Inform. 97–102.

Borgulya, I., 2008. An algorithm for the capacitated vehicle routing problem with route balancing. Cent. Eur. J. Oper. Res. 16, 331–343. https://doi.org/10.1007/s10100-008-0062-2

Braekers, K., Ramaekers, K., Van Nieuwenhuyse, I., 2016. The vehicle routing problem: State of the art classification and review. Comput. Ind. Eng. 99, 300–313. https://doi.org/10.1016/j.cie.2015.12.007

Burke, E., Curtois, T., Hyde, M., Kendall, G., Ochoa, G., Petrovic, S., Vázquez-Rodríguez, J.A., Gendreau, M., 2010. Iterated local search vs. hyper-heuristics: Towards general-purpose search algorithms, in: Evolutionary Computation (CEC), 2010 IEEE Congress On. IEEE, pp. 1–8.

Burke, E.K., Curtois, T., Hyde, M., Kendall, G., Ochoa, G., Petrovic, S., Vázquez-Rodríguez, J.A., 2009. Hyflex: A flexible framework for the design and analysis of hyper-heuristics, in: Multidisciplinary International Scheduling Conference (MISTA 2009), Dublin, Ireland. pp. 790–797.

Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Qu, R., 2013. Hyper-heuristics: a survey of the state of the art. J. Oper. Res. Soc. 64, 1695–1724. https://doi.org/10.1057/jors.2013.71

Chiang, T.-C., Hsu, W.-H., 2014. A knowledge-based evolutionary algorithm for the multiobjective vehicle routing problem with time windows. Comput. Oper. Res. 45, 25–37. https://doi.org/10.1016/j.cor.2013.11.014

Dabia, S., Talbi, E.-G., van Woensel, T., De Kok, T., 2013. Approximating multi-objective scheduling problems. Comput. Oper. Res. 40, 1165–1175. https://doi.org/10.1016/j.cor.2012.12.001

Demeester, P., Bilgin, B., De Causmaecker, P., Vanden Berghe, G., 2012. A hyperheuristic approach to examination timetabling problems: benchmarks and a new problem from practice. J. Sched. 15, 83–103. https://doi.org/10.1007/s10951-011-0258-5

Eksioglu, B., Vural, A.V., Reisman, A., 2009. The vehicle routing problem: A taxonomic review. Comput. Ind. Eng. 57, 1472–1483. https://doi.org/10.1016/j.cie.2009.05.009

El-Sherbeny, N., 2001. Resolution of a vehicle routing problem with multi-objective simulated annealing method. Faculté Polytechnique de Mons.

El-Sherbeny, N.A., 2010. Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. J. King Saud Univ. - Sci. 22, 123–131. https://doi.org/10.1016/j.jksus.2010.03.002

Garcia-Najera, A., Bullinaria, J.A., 2011. An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows. Comput. Oper. Res. 38, 287–300. https://doi.org/10.1016/j.cor.2010.05.004

Geiger, M., 2001. A genetic algorithm applied to multiple objective vehicle routing problems. Citeseer.

Glover, F., Kochenberger, G., 2005. Handbook of Metaheuristics (International Series in Operations Research and Management Science). J.-Oper. Res. Soc. 56, 614–614.

Jensen, M.T., 2003. Guiding single-objective optimization using multi-objective methods, in: Workshops on Applications of Evolutionary Computation. Springer, pp. 268–279.

Jozefowiez, N., Semet, F., Talbi, E.-G., 2008. Multi-objective vehicle routing problems. Eur. J. Oper. Res. 189, 293–309. https://doi.org/10.1016/j.ejor.2007.05.055

Jozefowiez, N., Semet, F., Talbi, E.-G., 2007. Target aiming Pareto search and its application to the vehicle routing problem with route balancing. J. Heuristics 13, 455–469. https://doi.org/10.1007/s10732-007-9022-6

Jozefowiez, N., Semet, F., Talbi, E.-G., 2005. Enhancements of NSGA II and its application to the vehicle routing problem with route balancing, in: International Conference on Artificial Evolution (Evolution Artificielle). Springer, pp. 131–142.

Jozefowiez, N., Semet, F., Talbi, E.-G., 2002. Parallel and hybrid models for multi-objective optimization: Application to the vehicle routing problem, in: International Conference on Parallel Problem Solving from Nature. Springer, pp. 271–280.

Kendall, G., Hussin, N.M., 2005. An investigation of a tabu-search-based hyper-heuristic for examination timetabling, in: Multidisciplinary Scheduling: Theory and Applications. Springer, pp. 309–328.

Knowles, J.D., Watson, R.A., Corne, D.W., 2001. Reducing local optima in single-objective problems by multi-objectivization, in: International Conference on Evolutionary Multi-Criterion Optimization. Springer, pp. 269–283.

Laporte, G., 1992. The vehicle routing problem: An overview of exact and approximate algorithms. Eur. J. Oper. Res. 59, 345–358.

Lee, C.-Y., Lee, Z.-J., Lin, S.-W., Ying, K.-C., 2010. An enhanced ant colony optimization (EACO) applied to capacitated vehicle routing problem. Appl. Intell. 32, 88–95. https://doi.org/10.1007/s10489-008-0136-9

Lee, T.-R., Ueng, J.-H., 1999. A study of vehicle routing problems with load-balancing. Int. J. Phys. Distrib. Logist. Manag. 29, 646–657.

Lenstra, J.K., Kan, A.H.G., 1981. Complexity of vehicle routing and scheduling problems. Networks 11, 221–227.
Li, W., Özcan, E., John, R., 2017. Multi-objective evolutionary algorithms and hyper-heuristics for wind farm layout optimisation. Renew. Energy 105, 473–482. https://doi.org/10.1016/j.renene.2016.12.022
Maashi, M., Kendall, G., Özcan, E., 2015. Choice function based hyper-heuristics for multi-objective optimization. Appl. Soft Comput. 28, 312–326. https://doi.org/10.1016/j.asoc.2014.12.012
Maashi, M., Özcan, E., Kendall, G., 2014. A multi-objective hyper-heuristic based on choice function. Expert Syst. Appl. 41, 4475–4493. https://doi.org/10.1016/j.eswa.2013.12.050
Matl, P., Hartl, R.F., Vidal, T., 2017. Workload Equity in Vehicle Routing Problems: A Survey and Analysis. Transp. Sci. https://doi.org/10.1287/trsc.2017.0744
Melián-Batista, B., De Santiago, A., AngelBello, F., Alvarez, A., 2014. A bi-objective vehicle routing problem with time windows: A real case in Tenerife. Appl. Soft Comput. 17, 140–152. https://doi.org/10.1016/j.asoc.2013.12.012
Mourgaya Virapatrin, M., 2004. The periodic vehicle routing problem: planning before routing. (Doctoral dissertation, Bordeaux 1.
Muklason, A., 2017. Hyper-heuristics and fairness in examination timetabling problems. Philosophy.
Nahum, O.E., Hadas, Y., Spiegel, U., 2014. Multi-objective vehicle routing problems with time windows: A vector evaluated artificial bee colony approach. Int J Comput Inf Technol 3, 41–47.
Ombuki, B., Ross, B.J., Hanshar, F., 2006. Multi-objective genetic algorithms for vehicle routing problem with time windows. Appl. Intell. 24, 17–30.
Pasia, J.M., Doerner, K.F., Hartl, R.F., Reimann, M., 2007. A population-based local search for solving a bi-objective vehicle routing problem, in: European Conference on Evolutionary Computation in Combinatorial Optimization. Springer, pp. 166–175.
Qi, Y., Hou, Z., Li, H., Huang, J., Li, X., 2015. A decomposition based memetic algorithm for multi-objective vehicle routing problem with time windows. Comput. Oper. Res. 62, 61–77. https://doi.org/10.1016/j.cor.2015.04.009
Ribeiro, R., Lourenc¸o, H.R., 2001. A multi-objective model for a multi-period distribution management problem. Presented at the Metaheuristic International Conference 2001 (MIC'2001), pp. 91–102.
Tan, K.C., Chew, Y.H., Lee, L.H., 2006. A Hybrid Multiobjective Evolutionary Algorithm for Solving Vehicle Routing Problem with Time Windows. Comput. Optim. Appl. 34, 115–151. https://doi.org/10.1007/s10589-005-3070-3
Walker, J.D., 2015a. Design of vehicle routing problem domains for a hyper-heuristic framework. University of Nottingham.
Walker, J.D., 2015b. Design of vehicle routing problem domains for a hyper-heuristic framework (PhD Thesis). University of Nottingham.
Wang, J., Zhou, Y., Wang, Y., Zhang, J., Chen, C.L.P., Zheng, Z., 2016. Multiobjective Vehicle Routing Problems With Simultaneous Delivery and

Pickup and Time Windows: Formulation, Instances, and Algorithms. IEEE Trans. Cybern. 46, 582–594. https://doi.org/10.1109/TCYB.2015.2409837

LAMPIRAN A KODE PROGRAM PYTHON

A.1 Kode Program untuk menampilkan Boxplot

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  import pandas as pd
4  import seaborn as sns
5  import scipy
6
7  dataA = pd.read_csv("data\hv_720.csv")
8  sns.boxplot( x="Hyperheuristic", y="hv",data=dataA, palette="Set3")
9  #plt.boxplot(dataA['HC_520'])
10 plt.title('Boxplot of Hypervolume 7_20')
11 ##plt.legend()
12 plt.show()
```

A.2 Kode Program untuk menampilkan Swarmplot

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  import pandas as pd
4  import seaborn as sns
5  import scipy
6
7  dataA = pd.read_csv("data\hv_780.csv")
8  sns.swarmplot( x="Hyperheuristic", y="hv",data=dataA)
9  plt.title('Swarmplot of Hypervolume 7_80')
10 plt.show()
```

A.3 Kode Program untuk menampilkan Histogram

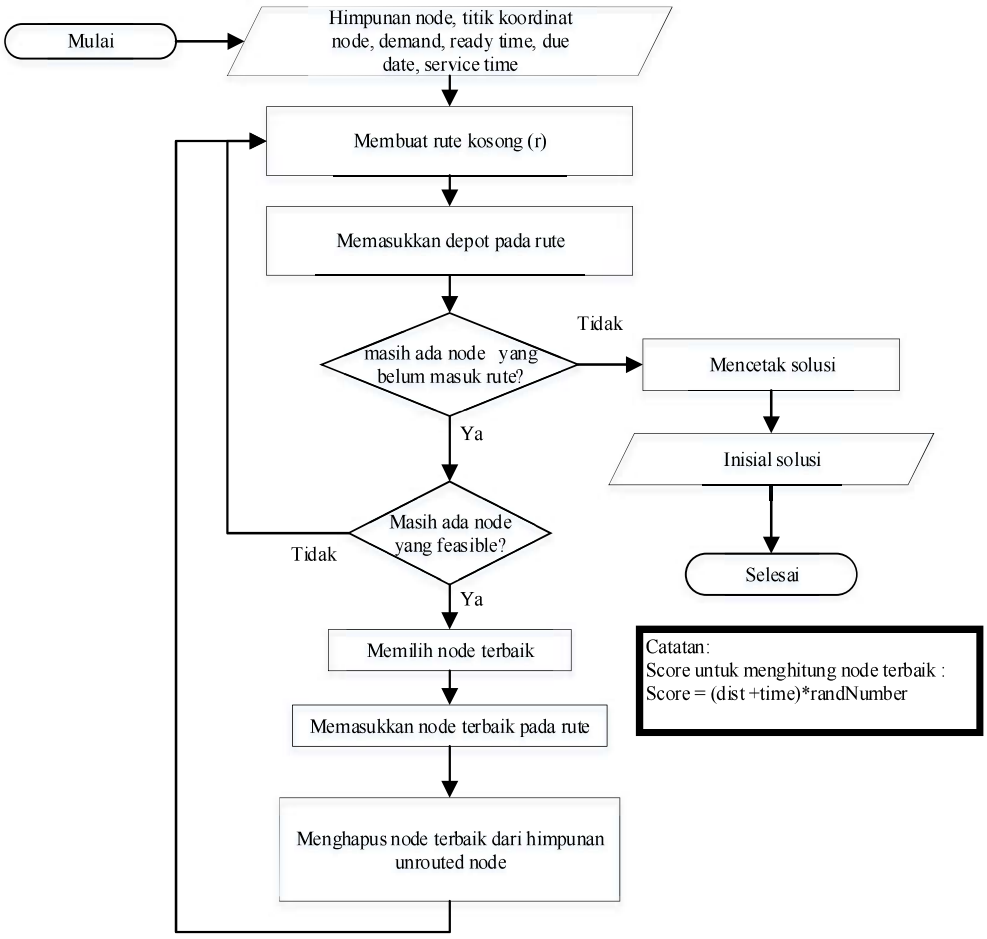
```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  import pandas as pd
4  import seaborn as sns
5  import scipy
6  dataA = pd.read_csv("data\hv.csv")
7  plt.hist([dataA.HC_720,dataA.GD_720], bins=[0,0.25,0.5,0.75,1],
   rwidth=1, color=['blue','yellow'], label=['HC','GD'])
8  plt.ylabel('number of data')
9  plt.xlabel('hypervolume')
10 plt.title('Histogram of Hypervolume 7_20')
11 plt.legend()
12 plt.show()
```

A.4 Kode Program untuk menampilkan Scatter Plot

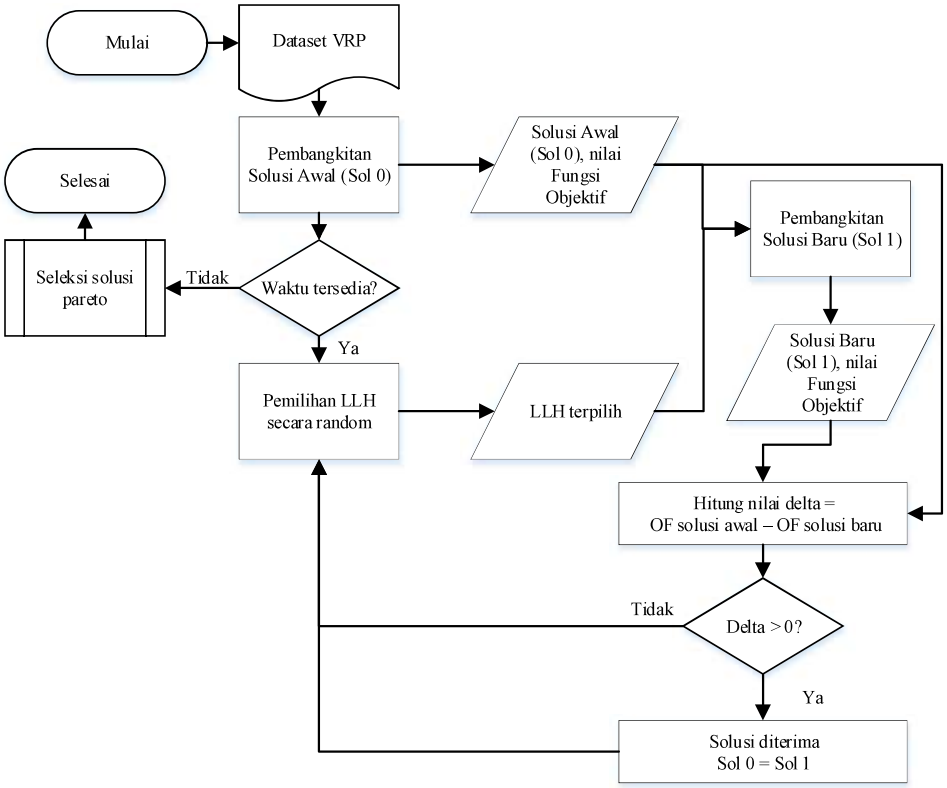
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
#def Union(dataA, dataB):
#    final_list = list(set(dataA) | set(dataB))
#    return final_list
#import csv
#data = list(csv.reader(open(datafile)))
#print(data[1][4])
dataA = pd.read_csv("datagd\hom519_80.csv")
dataB = pd.read_csv("datahc\hom519_80.csv")
A_of1 = dataA.OF1
A_of2 = dataA.OF2
B_of1 = dataB.OF1
B_of2 = dataB.OF2

#df.plot()  # plots all columns against index
plt.figure(figsize=(10,7))
plt.plot(A_of1,A_of2,'ko',label='GD') # scatter plot
plt.plot(B_of1,B_of2, 'r+', label='HC')  # blue
#plt.plot([0.8],[0.8], 'r*', label='Ref_point')
plt.title('Plot of Nondominated Solutions 5_80')
plt.xlabel('OF1'); plt.ylabel('OF2')
plt.xlim(0, 1.1); plt.ylim(0, 1.1)
plt.legend(loc='best')

plt.show()
#print(dataA)
#print(dataB)
```

LAMPIRAN B FLOWCHART ALGORITMA

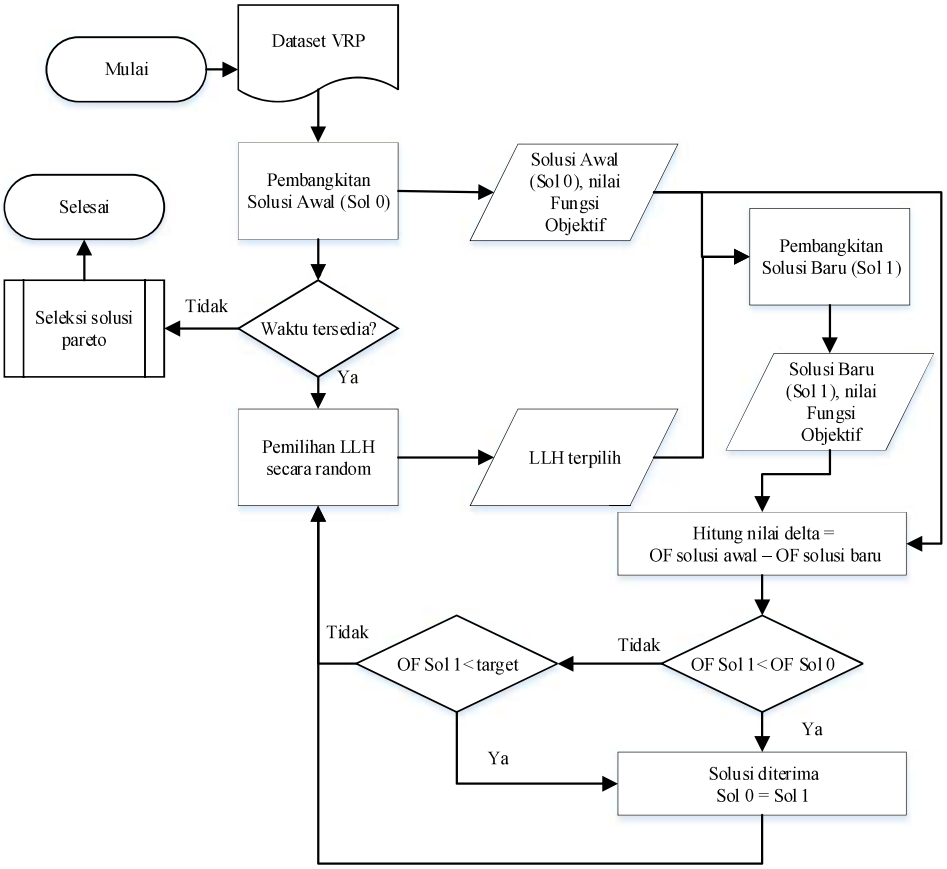
B.1 Flowchart Algoritma Inisialisasi Solusi



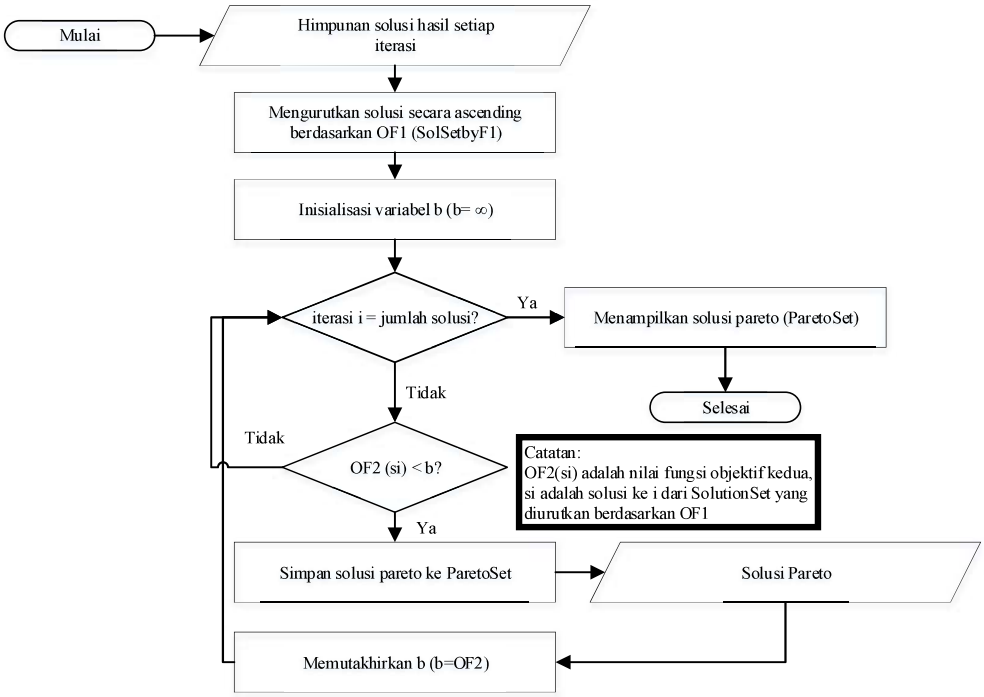
B.2 Flowchart Algoritma Hiperheuristik Hill Climbing



B.3 Flowchart Algoritma Hiperheuristik Great Deluge



B.4 Flowchart Algoritma Seleksi Solusi Pareto



LAMPIRAN C REKAPITULASI NILAI COVERAGE

C.1 Nilai Coverage Instance 0

Running ke-	0 (instance)					
	20 (running time (s))		40		80	
	HC	GD	HC	GD	HC	GD
1	0	1	0	1	0	1
2	0	1	0	1	0	1
3	0	1	0	1	0	1
4	0	0.923076923	0	1	0	1
5	0	1	0	1	0	1
6	0	1	0	1	0	1
7	0	1	0	1	0	1
8	0	1	0	0.810811	0	0.897436
9	0	1	0	1	0	1
10	0	1	0	1	0	1
11	0	1	0	1	0	1
12	0	1	0	1	0	1
13	0	1	0	1	0	1
14	0	1	0	1	0	1
15	0	1	0	1	0	1
16	0	1	0	0.090909	0	1
17	0	1	0	1	0	1
18	0	1	0	1	0	1
19	0	1	0	1	0	1
20	0	1	0	1	0	1
21	0	1	0	1	0	1
22	0	1	0	1	0	1
23	0	1	0	1	0	1
24	0	1	0	1	0	1
25	0	1	0	1	0	1

C.2 Nilai Coverage Instance 1

Running ke-	1 (instance)					
	20 (running time (s))		40		80	
	HC	GD	HC	GD	HC	GD
1	0	1	0	1	0	1
2	0	1	0	1	0	1
3	0	1	0	1	0	1
4	0	1	0	1	0	1
5	0	1	0	1	0	1
6	0	1	0	1	0	1
7	0	1	0	1	0	1
8	0	1	0	1	0	1
9	0	1	0	1	0	1
10	0	1	0	1	0	1
11	0	1	0	1	0	1
12	0	1	0	1	0	1
13	0	1	0	1	0	1
14	0	1	0	1	0	1
15	0	1	0	1	0	1
16	0	1	0	1	0	1
17	0	1	0	1	0	1
18	0	1	0	1	0	1
19	0.233333	0.681818	0	1	0	1
20	0	1	0	1	0	1
21	0	1	0	1	0	1
22	0	1	0	1	0	1
23	0	1	0	1	0	1
24	0	1	0	1	0	1
25	0	1	0	1	0	1

C.3 Nilai Coverage Instance 5

Running ke-	5 (instance)					
	20 (running time (s))		40		80	
	HC	GD	HC	GD	HC	GD
1	0.428571	0.461538	0	1	0.533333	0.5
2	0	1	0	1	0	1
3	0	0.923077	0	1	0	0.833333
4	0	1	0	1	0	1
5	0.047619	0.444444	0.666667	0.090909	0	1
6	0	1	0.62069	0	0	0.95
7	0	1	0	1	0	1
8	0	0.777778	0	1	0	1
9	0	0.75	0.695652	0.5	0	1
10	0	1	0	1	0	1
11	0	1	0	1	0	0.857143
12	0	0.6875	0.208333	0	0.142857	0.75
13	0	0.75	0	1	0	1
14	0	1	0	1	0	0.933333
15	0	0.947368	0.111111	0.705882	0	1
16	0	1	0	1	0	1
17	0	1	0	1	0	1
18	0.727273	0.5	0	1	0.333333	0.684211
19	0	1	0	1	0	1
20	0	1	0.416667	0.5	0.666667	0
21	0	1	0.45	0.25	0.214286	0.7
22	0	1	0	1	0	1
23	0.277778	0.3	0	1	0.15	0.538462
24	0.3	0.875	0	1	0	1
25	0.214286	0.461538	0.740741	0	0.555556	0.210526

C.4 Nilai Coverage Instance 6

Running ke-	6 (instance)					
	20 (running time (s))		40		80	
	HC	GD	HC	GD	HC	GD
1	0.047619	0.647059	0.555556	0.533333	0.5625	0
2	0.125	0.444444	0	1	0.1	0
3	0.047619	0.52381	0.125	0	0.142857	0
4	0.166667	0	0.157895	0	0.133333	0.227273
5	0.352941	0.142857	0.272727	0	0.043478	0.47619
6	0.388889	0.142857	0.357143	0	0	1
7	0.423077	0	0.333333	0	0.6	0.166667
8	0.111111	1	0.230769	0	0	0.833333
9	0.285714	0.230769	0.0625	0	0.083333	0.230769
10	0.083333	0.266667	0.347826	0.9	0.076923	0.434783
11	0.117647	0.272727	0.1875	0.083333	0	1
12	0.041667	0.428571	1	0	0	1
13	0	1	0.2	0.333333	0.111111	0
14	0.1	1	0.733333	0	0.1	0.074074
15	0.1	0.166667	0.111111	1	0.4	0
16	0.35	0	1	0	0.666667	0
17	0.076923	0.5	0.111111	0	0.05	0.285714
18	0.083333	0	0.1	0.588235	0	0.6875
19	0.105263	0	0	1	0	1
20	0.777778	0.25	0.090909	1	0.153846	0.230769
21	0.076923	0	0.565217	0	0.142857	0
22	0.090909	0	0.090909	0.071429	0.153846	0.368421
23	0.25	0	0.636364	1	0.416667	0.25
24	0.142857	0	0.083333	0	0.052632	0.222222
25	0	1	0	1	0.263158	0

C.5 Nilai Coverage Instance 7

Running ke-	7 (instance)					
	20 (running time (s))		40		80	
	HC	GD	HC	GD	HC	GD
1	0	1	0	1	0	0.636364
2	0	1	0	0.846154	0	0.275862
3	0	0.642857	0	1	0	1
4	0	1	0	0.9	0.111111	0.470588
5	0	0.8125	0	0.9	0	1
6	0	1	0	0.470588	0	1
7	0	1	0	0.863636	0	0.9
8	0	0.235294	0	1	0	1
9	0.052632	0.26087	0	1	0	1
10	0	1	0	1	0	0.666667
11	0	0.521739	0.166667	0.5	0	0.818182
12	0	1	0	1	0	0.923077
13	0	0.952381	0	0.214286	0	1
14	0	1	0	0.913043	0	1
15	0	1	0	0.764706	0	1
16	0	0.588235	0	0.444444	0	1
17	0	0.636364	0	1	0	0.7
18	0.047619	0.411765	0	1	0	1
19	0	0.875	0	1	0	0.846154
20	0	0.625	0	0.703704	0	1
21	0	1	0	0.611111	0	1
22	0	0.666667	0	0.5	0	0.869565
23	0	1	0	0.571429	0	1
24	0	1	0	1	0	1
25	0	1	0	0.642857	0	1

C.6 Nilai Coverage Instance 13

Running ke-	13 (instance)					
	20 (running time (s))		40		80	
	HC	GD	HC	GD	HC	GD
1	0	0.714286	0	1	0	1
2	0	0.692308	0	1	0	0.6
3	0	1	0	0.941176	0	0.792453
4	0	0.766667	0	1	0	1
5	0	1	0	1	0	1
6	0	0.785714	0	1	0	1
7	0	1	0	0.878049	0	1
8	0	1	0	1	0	1
9	0	0.807692	0	1	0	1
10	0	1	0	0.913043	0	0.4375
11	0	0.782609	0	1	0	1
12	0	0.85	0	1	0	1
13	0	1	0	1	0	0.878049
14	0	0.666667	0	1	0	0.906977
15	0	1	0	1	0	1
16	0	0.678571	0	1	0	1
17	1	0.633663	0	1	0	1
18	0	0.740741	0	1	0	1
19	0	1	0	0.5	0	1
20	0	1	0	0.5	0	1
21	0	1	0	0.882353	0	1
22	0	0.695652	0	0.87234	0	1
23	0	1	0	1	0	1
24	0	1	0	1	0	1
25	0	1	0	1	0	0.733333

LAMPIRAN D REKAPITULASI JUMLAH PARETO FRONT

D.1 Jumlah Pareto Front Instance 0

Running ke-	0 (instance)					
	20 (running time (s))		40		80	
	HC	GD	HC	GD	HC	GD
1	13	21	8	32	11	34
2	13	23	10	29	12	46
3	9	38	4	20	5	44
4	4	26	5	36	11	24
5	9	42	11	27	10	28
6	11	31	8	30	4	18
7	9	30	10	43	11	25
8	7	9	6	37	11	39
9	7	36	9	26	6	30
10	8	19	19	25	8	34
11	12	15	8	16	3	24
12	5	22	6	35	7	29
13	6	20	8	49	8	28
14	8	34	10	23	11	39
15	15	17	6	32	6	30
16	5	30	2	22	10	36
17	6	23	7	21	7	21
18	9	16	8	15	11	32
19	6	46	5	12	9	11
20	5	32	6	23	7	22
21	9	31	16	27	10	21
22	7	44	13	55	25	35
23	6	32	9	30	10	37
24	9	20	11	18	12	26
25	10	53	7	13	11	19

D.2 Jumlah Pareto Front Instance 1

Running ke-	1 (instance)					
	20 (running time (s))		40		80	
	HC	GD	HC	GD	HC	GD
1	1	22	20	32	33	40
2	17	11	14	37	1	54
3	1	21	1	36	7	50
4	1	17	1	24	12	32
5	1	29	2	57	15	31
6	20	33	1	25	1	44
7	14	43	15	44	1	39
8	17	21	18	37	10	57
9	1	23	1	50	26	52
10	1	44	1	25	1	49
11	33	14	20	24	11	61
12	11	15	27	39	11	49
13	19	28	12	38	2	18
14	1	16	10	47	19	33
15	1	29	22	42	1	50
16	39	29	13	31	31	34
17	15	22	21	26	1	40
18	25	23	1	22	32	22
19	30	22	1	34	1	48
20	24	18	9	29	4	53
21	19	40	27	24	1	68
22	17	28	23	22	24	32
23	9	27	18	42	1	46
24	29	32	17	20	16	43
25	1	27	1	32	25	26

D.3 Jumlah Pareto Front Instance 5

Running ke-	5 (instance)					
	20 (running time (s))		40		80	
	HC	GD	HC	GD	HC	GD
1	7	13	16	15	15	8
2	14	16	24	29	34	27
3	14	13	14	28	8	12
4	17	30	10	23	14	13
5	21	18	21	22	14	21
6	10	7	29	22	20	20
7	11	33	18	13	24	17
8	2	9	21	20	14	24
9	16	16	23	4	16	17
10	21	12	19	26	20	18
11	23	12	10	20	24	7
12	13	16	24	3	14	16
13	18	8	12	8	19	12
14	7	20	16	20	15	15
15	20	19	18	17	16	11
16	19	19	16	19	24	16
17	15	32	18	27	13	16
18	11	4	14	21	12	19
19	14	9	19	14	19	30
20	15	14	12	12	15	11
21	13	23	20	12	14	20
22	12	17	15	31	13	24
23	18	20	20	14	20	13
24	10	8	26	9	9	24
25	14	13	27	16	18	19

D.4 Jumlah Pareto Front Instance 6

Running ke-	6 (instance)					
	20 (running time (s))		40		80	
	HC	GD	HC	GD	HC	GD
1	21	17	9	15	16	7
2	8	18	14	8	10	10
3	21	21	8	7	14	21
4	12	17	19	7	15	22
5	17	14	11	20	23	21
6	18	7	14	12	15	19
7	26	4	18	9	15	6
8	9	8	26	7	14	6
9	7	13	16	24	12	26
10	12	15	23	20	13	23
11	17	11	16	12	17	10
12	24	14	9	3	12	16
13	19	13	15	15	9	19
14	10	10	15	12	10	27
15	10	6	9	11	5	15
16	20	19	12	9	15	7
17	13	18	9	13	20	21
18	12	6	10	17	16	16
19	19	11	10	8	15	14
20	9	4	11	20	13	13
21	13	20	23	7	7	16
22	11	8	11	14	13	19
23	12	11	11	10	12	16
24	7	9	12	20	19	18
25	8	15	11	12	19	10

D.5 Jumlah Pareto Front Instance 7

Running ke-	7 (instance)					
	20 (running time (s))		40		80	
	HC	GD	HC	GD	HC	GD
1	14	21	19	23	28	33
2	18	21	11	13	18	29
3	19	14	12	14	28	32
4	20	21	17	20	27	17
5	22	16	21	10	23	14
6	17	25	23	17	23	28
7	18	26	16	22	23	20
8	22	17	11	18	19	20
9	19	23	16	22	16	15
10	22	20	22	15	19	12
11	26	23	18	22	18	22
12	12	22	19	19	13	13
13	15	21	14	28	21	29
14	13	17	17	23	21	19
15	14	11	20	34	20	29
16	7	17	12	18	14	17
17	14	11	9	19	21	20
18	21	17	18	21	19	22
19	22	8	9	17	16	13
20	12	16	17	27	21	21
21	5	17	19	18	19	22
22	21	15	13	14	23	23
23	19	17	7	21	13	28
24	17	17	15	14	20	27
25	17	21	12	28	13	22

D.6 Jumlah Pareto Front Instance 13

Running ke-	13 (instance)					
	20 (running time (s))		40		80	
	HC	GD	HC	GD	HC	GD
1	8	7	3	31	6	39
2	3	26	3	33	5	30
3	5	21	22	17	1	53
4	1	30	5	33	6	27
5	5	36	4	22	2	26
6	4	28	1	19	1	31
7	9	17	2	41	9	20
8	1	15	6	23	9	34
9	4	26	2	37	8	35
10	6	26	2	69	2	32
11	10	23	1	26	4	18
12	3	40	6	40	8	24
13	6	23	10	27	7	41
14	3	33	7	27	3	43
15	2	16	3	14	3	24
16	3	28	1	37	11	40
17	7	101	4	28	13	26
18	4	27	6	17	5	25
19	7	35	4	30	2	25
20	1	23	3	32	2	15
21	16	28	5	34	6	26
22	10	23	8	47	11	14
23	12	16	3	29	5	25
24	1	21	2	33	7	15
25	16	42	4	28	10	30

LAMPIRAN E REKAPITULASI NILAI HYPERVOLUME

E.1 Nilai Hypervolume Instance 0

Running ke-	0 (instance)					
	20 (running time (s))		40		80	
	HC	GD	HC	GD	HC	GD
1	0	0	0	0	0.431877	0
2	0.477743	0	0	0	0	0
3	0	0	0	0	0.372194	0
4	0	0	0	0	0.505442	0
5	0	0	0	0	0.521656	0
6	0.645478	0.148989	0	0	0	0
7	0.453881	0	0	0	0	0
8	0.6583	0	0	0	0.360947	0
9	0	0	0	0	0.466779	0.091436
10	0	0	0	0	0	0
11	0.431607	0	0.470794	0	0.351578	0
12	0.696558	0	0.464629	0	0.351578	0
13	0.381891	0	0.243529	0	0	0
14	0	0	0.329588	0	0.358926	0
15	0	0	0.260623	0	0	0
16	0.586157	0	0.480516	0	0.376153	0
17	0.307097	0	0.554497	0.095773	0	0
18	0.672229	0.20404	0	0	0.580801	0
19		0.239077	0	0	0	0
20	0.517698	0	0.414256	0	0.032117	0
21	0.717648	0	0.452285	0	0	0
22	0.394433	0	0.531527	0	0.631208	0.163538
23	0.333219	0	0.400854	0.074269	0	0
24	0.576725	0	0.566992	0	0.648748	0
25	0	0	0	0	0.619887	0

E.2 Nilai Hypervolume Instance 1

Running ke-	1 (instance)					
	20 (running time (s))		40		80	
	HC	GD	HC	GD	HC	GD
1	0.770602	0	0.800213	0	0.92283	0
2	0.908251	0	0.694221	0	0.865374	0
3	0.812936	0	0.91441	0	0.697265	0
4			0.952942	0	0.885798	0
5	0.921832	0	0.864864	0	0.912032	0
6	0.788365	0	0.955426	0	0.494163	0
7	0.898048	0	0.928565	0	0.818862	0
8	0.877558	0				
9	0.476216	0	0.835198	0	0.692091	0
10	0.871724	0	0.917923	0	0.409547	0
11	0.733509	0	0.806919	0	0.061497	0
12	0.433862	0	0.937584	0	0.669352	0
13	0.97833	0	0.897796	0	0.797398	0
14	0.932723	0	0.587862	0	0.877236	0
15	0.861111	0	0.96053	0	0.421904	0
16	0.683376	0			0.96459	0
17	0.481076	0	0.873511	0	0.917313	0
18	0.514254	0	0.795233	0	0.917802	0
19	0.825687	0	0.63248	0	0.848247	0
20	0.501368	0	0.968373	0	0.707891	0
21	0.803105	0	0.63583	0	0.859604	0
22	0.813763	0	0.82396	0	0.793215	0
23	0.83055	0	0.8974	0	0.9043	0
24	0.805279	0	0.419444	0	0.96667	0
25	0.849753	0	0.950127	0	0.853329	0

E.3 Nilai Hypervolume Instance 5

Running ke-	5 (instance)					
	20 (running time (s))		40		80	
	HC	GD	HC	GD	HC	GD
1	0.63889	0.580121	0.494743	0.034153	0.644792	0.675513
2	0.633635	0.00448	0.619565	0.011544	0.698594	0.130839
3	0.608272	0.10803	0.586014	0.010635	0.218729	0
4	0.687647	0.132339	0.629934	0.082584	0.60038	0.230885
5	0.824587	0.591903	0.805133	0.900653	0.484515	0.259607
6	0.343134	0	0.614332	0.633755	0.64915	0.277313
7	0.604278	0.022209	0.51691	0	0.185989	0
8	0	0	0.731431	0.204467	0.484515	0.130025
9	0.698602	0.210702	0.72121	0.767382	0.193147	0
10	0.608265	0.212067	0.537898	0.016069	0.547316	0.309691
11	0.552831	0.038678	0.617978	0	0.33123	0.14963
12	0.497582	0.038861	0.727476	0.569178	0.610856	0.581453
13	0.767786	0.450857	0.797616	0.158376	0.447141	0.005177
14	0	0	0.547685	0.075838	0.686325	0.412337
15	0.742047	0.572551	0.679487	0.642024	0.472877	0.136512
16	0.720303	0.188312	0.603303	0.016555	0.667369	0.211963
17	0.595008	0	0.79795	0.368295	0.593222	0
18	0.60782	0.766094	0.53828	0.008694	0.864466	0.619661
19	0.481928	0.100518	0.595291	0.084844	0.434516	0
20	0.595008	0.053511	0.749728	0.639941	0.7585	0.88939
21	0.481557	0.007628	0.573448	0.556072	0.74815	0.47561
22	0.536177	0	0.497422	0	0.567171	0.060788
23	0.681453	0.712496	0.453907	0.13435	0.800409	0.547167
24	0.728922	0.480661	0.613724	0.042235	0.735433	0
25	0.874754	0.482141	0.676747	0.804803	0.694065	0.701594

E.4 Hypervolume Instance 6

Running ke-	6 (instance)					
	20 (running time (s))		40		80	
	HC	GD	HC	GD	HC	GD
1	0.793067	0.447161	0.738544	0.441898	0.751195	0.961051
2	0.843438	0.442025	0.666544	0.440098	0.718505	0.833882
3	0.894902	0.500459	0.815684	0.748009	0.811278	0.635536
4	0.466698	0.544277	0.792982	0.760208	0.612032	0.43375
5	0.590126	0.45603	0.491992	0.505929	0.845351	0.48268
6	0.686442	0.77451	0.732827	0.716584	0.624307	0.094098
7	0.568767	0.549923	0.671884	0.692752	0.618271	0.912969
8	0.852839	0.368897	0.617345	0.829919	0.614924	0.327226
9	0.712177	0.626731	0.578921	0.564825	0.774552	0.657659
10	0.767745	0.614992	0.59839	0.507462	0.809174	0.563575
11	0.647228	0.851097	0.784908	0.927637	0.489454	0.360392
12	0.88529	0.480183	0	0	0.121158	0
13	0.807876	0.50941	0.687172	0.451348	0.613836	0.705735
14	0.562054	0.490881	0.611573	0.802898	0.751391	0.586646
15	0.853624	0.785354	0.706055	0.376201	0.478015	0.557048
16	0.607032	0.440507	0	0	0.633781	0.983257
17	0.780559	0.449767	0.535597	0.607302	0.806598	0.550222
18	0.620123	0.791257	0.527371	0.748986	0.567301	0.070552
19	0.579762	0.741175	0.645181	0.226294	0.483385	0.265365
20	0.465908	0.788386	0.411691	0.135265	0.853866	0.55343
21	0.401531	0.473739	0.703792	0.896238	0.562917	0.606011
22	0.805577	0.806849	0.631695	0.538947	0.789346	0.560596
23	0.553275	0.614021	0.631198	0.717282	0.562671	0.545277
24	0.502336	0.833209	0.55521	0.626364	0.752579	0.605607
25	0.756378	0.382286	0.552111	0.398973	0.590684	0.729295

E.5 Hypervolume Instance 7

Running ke-	7 (instance)					
	20 (running time (s))		40		80	
	HC	GD	HC	GD	HC	GD
1	0.374172	0	0.381016	0	0.83156	0.013397
2	0.570969	0	0.74948	0	0.395143	0
3	0.480276	0	0.719302	0.244647	0.642095	0
4	0.670975	0	0.609018	0	0.854015	0.325031
5	0.75753	0	0.673497	0	0.46048	0
6	0.456717	0	0.782056	0	0.258595	0
7	0.774793	0.243794	0.631673	0	0.504268	0
8	0.663199	0	0.711139	0	0.603689	0
9	0.813458	0.245137	0.465284	0	0.615962	0
10	0.599532	0	0.589722	0	0.668934	0
11	0.763143	0.049188	0.793782	0.228235	0.658634	0
12	0.212529	1	0.647274	0	0.521994	0
13	0.555624	0	0.755778	0	0.60908	0
14	0.356297	0	0.668329	0	0.649544	0
15	0.560646	0	0.728132	0	0.506073	0
16	0.606737	0	0.793692	0	0.5087	0
17	0.684283	0	0.626463	0	0.473116	0
18	0.913154	0.48128	0.661412	0.009466	0.432495	0
19	0.632826	0	0.574465	0	0.484325	0
20	0.59961	0	0.697763	0	0.553286	0
21	0.127457	0	0.653151	0	0.684699	0.03916
22	0.568498	0	0.482092	0	0.608373	0
23	0.719173	0	0.367002	0	0.473623	0
24	0.668659	0	0.488223	0	0.666876	0
25	0.517748	0	0.423368	0	0.560965	0

E.6 **Nilai Hypervolume Instance 13**

Running ke-	13 (instance)					
	20 (running time (s))		40		80	
	HC	GD	HC	GD	HC	GD
1	0.302808	0	0.858045	0	0.533425	0
2	0.735422	0	0.950561	0	0.809246	0
3	0.678724	0	0.508441	0	0	0
4	0	0	0.704877	0	0.318206	0
5	0.843123	0	0.717812	0	0	0
6	0.410776	0	0	0	0	0
7	0.663669	0	0	0	0.548548	0
8	0	0	0.192579	0	0.651497	0
9	0.214069	0	0	0	0.384236	0
10	0.034958	0	0	0	0	0
11	0.471883	0	0	0	0.783888	0
12	0.268519	0	0.729509	0	0.248682	0
13	0.55963	0	0.643361	0	0.707967	0
14	0.706821	0	0.088339	0	0.823073	0
15	0	0	0.889773	0	0.556025	0
16	0.228674	0	0	0	0.342535	0
17	0.412273	0	0.045004	0	0.425899	0
18	0.165621	0	0.693971	0	0.182076	0
19	0.55914	0	0.208638	0	0	0
20	0	0	0.713809	0	0	0
21	0.821037	0	0.8707	0	0.481813	0
22	0.602869	0	0.595064	0	0.501202	0
23	0.151688	0	0.793766	0	0.081752	0
24	0	0	0	0	0.578427	0
25	0.430616	0	0.407744	0	0.797814	0

LAMPIRAN F HASIL RUNNING VALIDASI

F.1 Hasil running ke 1 sampai 5

TIME_LIMIT (mili second)	ALGORITMA	RUNNING KE									
		1		2		3		4		5	
		TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV
20000	SOVRP_HC	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52
	MOVRP_HC (MIN_TOTAL_DISTANCE)	103629.2718	1232.25872	105772.5453	1154.22414	106811.8235	1007.77316	117297.24	1135.77012	116203.3965	696.01834
	MOVRP_HC (MIN_STDEV)	212053.0211	477.9449	212053.0211	477.9449	206332.5494	462.54354	206332.5494	462.54354	212053.0211	477.9449
	SOVRP_GD	64036.76	446.00	68278.68	558.93	68135.10	575.00	69101.65	589.49	68892.40	599.13
	MOVRP_GD (MIN_TOTAL_DISTANCE)	159020.6448	847.89968	176621.12	718.5528	176621.12	718.5528	176621.12	718.5528	172196.2565	709.87036
	MOVRP_GD (MIN_STDEV)	212053.0211	477.9449	199225.1045	448.15462	199225.1045	448.15462	199225.1045	448.15462	191539.0138	434.8395
40000	SOVRP_HC	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52
	MOVRP_HC (MIN_TOTAL_DISTANCE)	104830.977	1493.10008	99578.75152	1214.54102	108013.5286	1426.29438	108437.3613	850.53816	103242.239	739.66064
	MOVRP_HC (MIN_STDEV)	192766.0981	471.17996	212053.0211	477.9449	212053.0211	477.9449	212053.0211	477.9449	212053.0211	477.9449
	SOVRP_GD	59568.49	483.75	59568.49	483.75	59568.49	483.75	59568.49	483.75	59568.49	483.75
	MOVRP_GD (MIN_TOTAL_DISTANCE)	183233.6706	733.64736	159520.615	901.0988	154423.2027	899.68752	169598.6954	706.5109	172632.7787	742.29912
	MOVRP_GD (MIN_STDEV)	205775.476	460.11982	212053.0211	477.9449	212053.0211	477.9449	188351.3862	447.28024	202337.8634	449.09036

F.1 Hasil running ke 1 sampai 5 (Lanjutan)

TIME_LIMIT (mili second)	ALGORITMA	RUNNING KE									
		1		2		3		4		5	
		TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV
80000	SOVRP_HC	94417.39	1188.96	94417.39	1188.96	94417.39	1188.96	94417.39	1188.96	94417.39	1188.96
	MOVRP_HC (MIN_TOTAL_DISTANCE)	99813.50912	1214.83248	97314.92688	1280.9172	98284.41232	1059.03944	102444.0632	893.13734	98873.20976	1052.15178
	MOVRP_HC (MIN_STDEV)	212053.0211	477.9449	212053.0211	477.9449	212053.0211	477.9449	206332.5494	462.54354	212053.0211	477.9449
	SOVRP_GD	46403.74	482.89	47863.45	475.91	47670.86	493.03	48911.54	468.86	47534.79	481.58
	MOVRP_GD (MIN_TOTAL_DISTANCE)	154522.1816	777.73452	160924.0848	882.64478	154921.904	810.5161	155509.4325	887.5229	183925.2538	723.3082
	MOVRP_GD (MIN_STDEV)	178251.7336	439.1347	212053.0211	477.9449	212053.0211	477.9449	212053.0211	477.9449	205594.0147	446.31382

F.2 Hasil running ke 6 sampai 10

TIME_LIMIT (mili second)	ALGORITMA	RUNNING KE									
		6		7		8		9		10	
		TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV
20000	SOVRP_HC	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52
	MOVRP_HC (MIN_TOTAL_DISTANCE)	116119.6451	1268.06228	116249.079	1048.4395	106794.0581	1455.0262	122468.252	1625.3002	122468.252	1625.3002
	MOVRP_HC (MIN_STDEV)	212053.0211	477.9449	212053.0211	477.9449	212053.0211	477.9449	212053.0211	477.9449	212053.0211	477.9449
	SOVRP_GD	69387.00	538.62	69101.65	589.49	68875.55	598.53	69101.65	589.49	69389.97	538.58
	MOVRP_GD (MIN_TOTAL_DISTANCE)	148484.4699	1179.94932	160443.149	820.08826	178491.567	770.2486	178725.0557	743.67972	161481.1582	954.14452
	MOVRP_GD (MIN_STDEV)	212053.0211	477.9449	212053.0211	477.9449	201948.2926	438.56712	195301.4802	468.31138	212053.0211	477.9449
40000	SOVRP_HC	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52
	MOVRP_HC (MIN_TOTAL_DISTANCE)	107820.6467	1225.75456	99479.77264	1091.49888	101894.6035	1043.86818	108068.0939	1468.372	101926.3275	1000.6554
	MOVRP_HC (MIN_STDEV)	212053.0211	477.9449	196532.3714	459.42952	212053.0211	477.9449	212053.0211	477.9449	212053.0211	477.9449
	SOVRP_GD	59568.49	483.75	59568.49	483.75	59568.49	483.75	59568.49	483.75	59569.07	485.39
	MOVRP_GD (MIN_TOTAL_DISTANCE)	127776.3117	1182.1276	177161.697	738.38742	176000.5986	706.8944	176482.8034	705.86662	149664.6027	922.91228
	MOVRP_GD (MIN_STDEV)	212053.0211	477.9449	202288.3739	442.63222	201500.3498	449.36648	201170.4202	445.6542	212053.0211	477.9449

F.2 Hasil running ke 6 sampai 10 (Lanjutan)

TIME_LIMIT (mili second)	ALGORITMA	RUNNING KE									
		6		7		8		9		10	
		TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV
80000	SOVRP_HC	94417.39	1188.96	94653.66	1053.52	94417.39	1188.96	94417.39	1188.96	94417.39	1188.96
	MOVRP_HC (MIN_TOTAL_DISTANCE)	101381.9437	1017.09988	101516.4534	1162.01686	102737.193	1090.90062	102314.6293	992.84734	101502.4949	1186.22338
	MOVRP_HC (MIN_STDEV)	212053.0211	477.9449	212053.0211	477.9449	212053.0211	477.9449	202226.1949	470.78112	212053.0211	477.9449
	SOVRP_GD	47784.94	475.05	48615.19	458.26	46855.15	488.64	50622.42	473.39	50622.42	473.39
	MOVRP_GD (MIN_TOTAL_DISTANCE)	143544.4086	786.0488	166267.6754	710.51464	176972.6219	730.47198	158868.3696	908.70744	164838.8264	685.37238
	MOVRP_GD (MIN_STDEV)	212053.0211	477.9449	192827.0082	470.45898	202594.1933	447.31092	212053.0211	477.9449	184242.4938	447.7251

F.3 Hasil running ke 11 sampai 15

TIME_LIMIT (mili second)	ALGORITMA	RUNNING KE									
		11		12		13		14		15	
		TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV
20000	SOVRP_HC	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52
	MOVRP_HC (MIN_TOTAL_DISTANCE)	106794.0581	1455.0262	103128.0326	1122.96122	107003.4365	931.18054	131897.8938	678.08588	110902.9506	1217.3329
	MOVRP_HC (MIN_STDEV)	212053.0211	477.9449	212053.0211	477.9449	212053.0211	477.9449	199409.1037	465.39678	212053.0211	477.9449
	SOVRP_GD	63383.44	447.78	63793.18	455.66	63301.45	444.51	70980.67	572.16	70819.45	567.63
	MOVRP_GD (MIN_TOTAL_DISTANCE)	167597.5454	890.88236	157787.2157	1038.86734	159620.8629	879.94494	178725.0557	743.67972	182844.0998	663.14472
	MOVRP_GD (MIN_STDEV)	212053.0211	477.9449	212053.0211	477.9449	212053.0211	477.9449	199633.7096	472.7293	201894.9963	453.38556
40000	SOVRP_HC	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52
	MOVRP_HC (MIN_TOTAL_DISTANCE)	101900.9483	1104.24642	102596.3384	1218.22262	99305.92512	1293.4193	102771.4549	804.65622	104488.3578	1210.46058
	MOVRP_HC (MIN_STDEV)	212053.0211	477.9449	189192.7067	475.81264	206332.5494	462.54354	212053.0211	477.9449	212053.0211	477.9449
	SOVRP_GD	59568.49	483.75	59568.49	483.75	59498.43	484.32	59568.49	483.75	59568.49	483.75
	MOVRP_GD (MIN_TOTAL_DISTANCE)	162547.0846	879.0859	152083.2405	842.40796	168928.6845	716.20578	156551.2486	884.20946	157685.6989	796.18854
	MOVRP_GD (MIN_STDEV)	212053.0211	477.9449	212053.0211	477.9449	192697.5742	440.31588	212053.0211	477.9449	212053.0211	477.9449

F.3 Hasil running ke 11 sampai 15 (Lanjutan)

TIME_LIMIT (mili second)	ALGORITMA	RUNNING KE									
		11		12		13		14		15	
		TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV
80000	SOVRP_HC	94653.66	1053.52	94653.66	1053.52	94417.39	1188.96	94653.66	1053.52	94653.66	1053.52
	MOVRP_HC (MIN_TOTAL_DISTANCE)	96756.58448	1307.0719	99020.40912	1545.839	100739.8499	1150.94138	96467.2616	1284.0619	100790.6083	1235.92498
	MOVRP_HC (MIN_STDEV)	212053.0211	477.9449	202392.4286	462.2981	212053.0211	477.9449	212053.0211	477.9449	212053.0211	477.9449
	SOVRP_GD	50760.35	486.36	50760.35	486.36	50760.35	486.36	50622.42	473.39	50506.87	468.12
	MOVRP_GD (MIN_TOTAL_DISTANCE)	176399.052	700.26752	170976.7859	717.87784	174668.1906	755.2154	180323.9453	768.868	149178.591	826.66912
	MOVRP_GD (MIN_STDEV)	205284.3885	456.97512	192187.4523	438.7512	199818.9778	430.52896	203700.7264	438.9813	181703.3048	403.49988

F.4 Hasil running ke 16 sampai 20

TIME_LIMIT (mili second)	ALGORITMA	RUNNING KE									
		16		17		18		19		20	
		TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV
20000	SOVRP_HC	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52
	MOVRP_HC (MIN_TOTAL_DISTANCE)	109928.3893	883.10498	120692.977	1195.71884	109084.5309	1059.2542	99590.17216	1005.7943	119331.3829	901.46696
	MOVRP_HC (MIN_STDEV)	212053.0211	477.9449	212053.0211	477.9449	202392.4286	462.2981	212053.0211	477.9449	195659.3269	477.39266
	SOVRP_GD	70145.86	577.97	70145.86	577.97	69950.03	573.21	69389.97	538.58	69387.00	538.62
	MOVRP_GD (MIN_TOTAL_DISTANCE)	169157.0973	777.12092	178725.0557	743.67972	158614.5776	896.19	178624.8078	734.07688	184469.6376	699.4545
	MOVRP_GD (MIN_STDEV)	200329.0997	471.79356	200329.0997	471.79356	212053.0211	477.9449	201915.2997	444.10486	205690.4557	445.70022
40000	SOVRP_HC	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52
	MOVRP_HC (MIN_TOTAL_DISTANCE)	106639.245	1062.36822	106207.7986	1428.90218	105640.5734	1061.9387	110331.9186	1258.9043	105102.5344	1214.15752
	MOVRP_HC (MIN_STDEV)	212053.0211	477.9449	212053.0211	477.9449	212053.0211	477.9449	212053.0211	477.9449	212053.0211	477.9449
	SOVRP_GD	59569.07	485.39	59568.49	483.75	59568.49	483.75	59569.07	485.39	59568.49	483.75
	MOVRP_GD (MIN_TOTAL_DISTANCE)	173528.6645	752.62294	179694.5411	756.71872	159634.8214	914.41392	162983.6069	909.27502	157391.3002	875.34294
	MOVRP_GD (MIN_STDEV)	194777.3997	450.01076	205594.0147	446.31382	212053.0211	477.9449	212053.0211	477.9449	212053.0211	477.9449

F.4 Hasil running ke 16 sampai 20 (Lanjutan)

TIME_LIMIT (mili second)	ALGORITMA	RUNNING KE									
		16		17		18		19		20	
		TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV
80000	SOVRP_HC	94417.39	1188.96	94417.39	1188.96	94417.39	1188.96	94653.66	1053.52	94653.66	1053.52
	MOVRP_HC (MIN_TOTAL_DISTANCE)	103419.8934	1064.59252	103518.8723	1049.8968	101355.2955	1177.2188	101845.1141	1156.97	100829.9461	998.0476
	MOVRP_HC (MIN_STDEV)	212053.0211	477.9449	202392.4286	462.2981	212053.0211	477.9449	206332.5494	462.54354	212053.0211	477.9449
	SOVRP_GD	50506.87	468.12	50637.84	473.81	50760.35	486.36	50760.35	486.36	48503.41	447.77
	MOVRP_GD (MIN_TOTAL_DISTANCE)	151037.6174	765.89204	156562.6693	929.5545	173984.2211	717.67842	166214.379	744.46206	168808.1333	729.87372
	MOVRP_GD (MIN_STDEV)	212053.0211	477.9449	212053.0211	477.9449	200315.1411	471.45608	184939.1528	461.92994	188639.4402	439.11936

F.5 Hasil running ke 21 sampai 25

TIME_LIMIT (mili second)	ALGORITMA	RUNNING KE									
		21		22		23		24		25	
		TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV
20000	SOVRP_HC	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52
	MOVRP_HC (MIN_TOTAL_DISTANCE)	107368.897	1239.806	111009.5432	1596.55304	113495.4358	1151.1408	99756.40592	1326.35428	106533.9213	1052.62732
	MOVRP_HC (MIN_STDEV)	212053.0211	477.9449	202392.4286	462.2981	206332.5494	462.54354	212053.0211	477.9449	187885.6779	470.84248
	SOVRP_GD	69101.65	589.49	69389.97	538.58	70145.86	577.97	69387.00	538.62	69101.65	589.49
	MOVRP_GD (MIN_TOTAL_DISTANCE)	155608.4114	794.1023	165215.7075	923.84802	162908.7382	863.7459	175249.3742	786.17152	173555.3126	701.32598
	MOVRP_GD (MIN_STDEV)	212053.0211	477.9449	212053.0211	477.9449	212053.0211	477.9449	198985.271	457.14386	197640.1734	436.1434
40000	SOVRP_HC	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52
	MOVRP_HC (MIN_TOTAL_DISTANCE)	104813.2115	1359.94888	104329.7378	1191.57704	100496.2096	998.59984	102002.4651	1225.9233	101643.3494	1206.24208
	MOVRP_HC (MIN_STDEV)	212053.0211	477.9449	212053.0211	477.9449	212053.0211	477.9449	212053.0211	477.9449	212053.0211	477.9449
	SOVRP_GD	59568.49	483.75	59568.49	483.75	59568.49	483.75	59568.49	483.75	52237.04	427.12
	MOVRP_GD (MIN_TOTAL_DISTANCE)	174050.207	716.81938	175461.2906	676.07634	173587.0366	693.99346	176141.4531	700.57432	168435.059	816.8055
	MOVRP_GD (MIN_STDEV)	196001.9461	446.25246	196307.7654	437.49332	195187.2738	451.16126	196031.1322	432.52316	186588.8008	433.93444

F.5 Hasil running ke 21 sampai 25

TIME_LIMIT (mili second)	ALGORITMA	RUNNING KE									
		21		22		23		24		25	
		TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV
80000	SOVRP_HC	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52	94653.66	1053.52
	MOVRP_HC (MIN_TOTAL_DISTANCE)	101360.3714	1062.96648	101049.4762	1229.54354	99741.1784	1251.97062	100328.7069	883.39644	100887.0493	1246.5556
	MOVRP_HC (MIN_STDEV)	212053.0211	477.9449	212053.0211	477.9449	212053.0211	477.9449	212053.0211	477.9449	212053.0211	477.9449
	SOVRP_GD	48896.01	469.30	46342.59	483.13	50760.35	486.36	48751.42	471.09	48764.51	473.90
	MOVRP_GD (MIN_TOTAL_DISTANCE)	176373.6728	718.18464	156099.4989	927.39156	161191.8354	881.70904	171474.2182	696.34048	155265.7922	857.70194
	MOVRP_GD (MIN_STDEV)	203252.7835	444.47302	212053.0211	477.9449	212053.0211	477.9449	195929.6154	452.03564	175268.4086	441.08288

F.6 Deskriptif Statistik Hasil Running Validasi

TIME_LIMIT (mili second)	ALGORITMA	MIN_BASED_0N_TOTAL_DISTANCE		MAX_BASED_0N_TOTAL_DISTANCE		MIN_BASED_ON_STDEV		MAX_BASED_ON_STDEV	
		TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV
20000	SOVRP_HC	94653.66	1053.52	94653.66	1053.52	94653.66	**1053.52**	94653.66	**1053.52**
	MOVRP_HC (MIN_TOTAL_DISTANCE)	**99590.17**	678.09	**131897.89**	1625.30	99590.17	**678.09**	131897.89	**1625.30**
	MOVRP_HC (MIN_STDEV)	**187885.68**	462.30	**212053.02**	477.94	187885.68	462.30	212053.02	477.94
	SOVRP_GD	63301.45	444.51	70980.67	572.16	63301.45	**444.51**	68892.40	**599.13**
	MOVRP_GD (MIN_TOTAL_DISTANCE)	**148484.47**	663.14	**184469.64**	1179.95	148484.47	**663.14**	184469.64	**1179.95**
	MOVRP_GD (MIN_STDEV)	**191539.01**	434.84	**212053.02**	477.94	191539.01	434.84	212053.02	477.94
40000	SOVRP_HC	94653.66	1053.52	94653.66	1053.52	94653.66	**1053.52**	94653.66	**1053.52**
	MOVRP_HC (MIN_TOTAL_DISTANCE)	**99305.93**	739.66	**110331.92**	1493.10	99305.93	**739.66**	110331.92	**1493.10**
	MOVRP_HC (MIN_STDEV)	**189192.71**	459.43	**212053.02**	477.94	189192.71	459.43	212053.02	477.94
	SOVRP_GD	52237.04	427.12	**59569.07**	485.39	52237.04	**427.12**	59569.07	**485.39**
	MOVRP_GD (MIN_TOTAL_DISTANCE)	**127776.31**	676.08	**183233.67**	1182.13	127776.31	**676.08**	183233.67	**1182.13**
	MOVRP_GD (MIN_STDEV)	**186588.80**	432.52	**212053.02**	477.94	186588.80	432.52	212053.02	477.94

F.6 Deskriptif Statistik Hasil Running Validasi (Lanjutan)

TIME_LIMIT (mili second)	ALGORITMA	MIN_BASED_0N_TOTAL_DISTANCE		MAX_BASED_0N_TOTAL_DISTANCE		MIN_BASED_ON_STDEV		MAX_BASED_ON_STDEV	
		TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV	TOT_DISTANCE	STDEV
80000	SOVRP_HC	94417.39	1188.96	94653.66	1053.51544	94653.66	**1053.52**	94417.39	**1188.96**
	MOVRP_HC (MIN_TOTAL_DISTANCE)	**96467.26**	883.40	**103518.87**	1545.84	96467.26	**883.40**	103518.87	**1545.84**
	MOVRP_HC (MIN_STDEV)	**202226.19**	462.30	**212053.02**	477.94	202226.19	462.30	212053.02	477.94
80000	SOVRP_GD	46342.59	483.13	50760.35	486.36	48503.41	**447.77**	47670.86	**493.03**
	MOVRP_GD (MIN_TOTAL_DISTANCE)	**143544.41**	685.37	**183925.25**	929.55	143544.41	**685.37**	183925.25	**929.55**
	MOVRP_GD (MIN_STDEV)	**175268.41**	403.50	**212053.02**	477.94	175268.41	403.50	212053.02	477.94

LAMPIRAN G HASIL VERIFIKASI

G.1 Perhitungan Manual

RUTE KE-	JARAK RUTE	TOTAL JARAK RUTE	STANDARD DEVIASI
0	202.8263567	1409.306205	57.3295708
1	362.6174847		
2	291.7038746		
3	317.9216523		
4	234.2368364		

RUTE				JARAK
Route	0			202.8263567
Location0	visited	at	0	
Location92	visited	at	14.76482	14.76482306
Location95	visited	at	31.47303	6.708203932
Location62	visited	at	50.69257	9.219544457
Location72	visited	at	90.75916	30.06659276
Location40	visited	at	233	20.22374842
Location39	visited	at	248	5
Location38	visited	at	260	2
Location41	visited	at	276.4031	6.403124237
Location42	visited	at	292.2341	5.830951895
Location44	visited	at	304.2341	2
Location61	visited	at	333.9572	19.72308292
Location100	visited	at	358.0993	14.14213562
Location90	visited	at	388.9799	20.88061302
Location56	visited	at	416.6717	17.69180601
Location64	visited	at	433.7428	7.071067812
Location66	visited	at	451.805	8.062257748
Location0	visited	at	0	13.03840481

G.1 Perhitungan Manual (Lanjutan 1)

RUTE				JARAK
Route	1			362.6174847
Location0	visited	at	0	
Location82	visited	at	85	14.76482306
Location65	visited	at	103.544	8.544003745
Location99	visited	at	170	10.29563014
Location52	visited	at	185.099	5.099019514
Location83	visited	at	207.1406	12.04159458
Location8	visited	at	272	53.71219601
Location7	visited	at	287	5
Location6	visited	at	300	3
Location2	visited	at	314	4
Location5	visited	at	334.198	10.19803903
Location79	visited	at	366.2208	22.02271555
Location78	visited	at	388.3863	12.16552506
Location73	visited	at	405.5974	7.211102551
Location53	visited	at	436.1886	20.59126028
Location55	visited	at	460.3308	14.14213562
Location88	visited	at	476.6553	6.32455532
Location3	visited	at	513.7293	27.07397274
Location46	visited	at	530.1324	6.403124237
Location4	visited	at	542.1324	2
Location45	visited	at	554.1324	2
Location1	visited	at	591	5.830951895
Location70	visited	at	619.868	18.86796226
Location37	visited	at	662.5635	32.69556545
Location35	visited	at	676.1691	3.605551275
Location36	visited	at	688.1691	2
Location43	visited	at	708.1691	10
Location54	visited	at	743.1691	25
Location0	visited	at	0	18.02775638

G.1 Perhitungan Manual (Lanjutan 2)

RUTE				JARAK
Route	2			291.7038746
Location0	visited	at	0	
Location69	visited	at	9.219544	9.219544457
Location98	visited	at	109	5
Location9	visited	at	236	23.34523506
Location16	visited	at	257.1803	11.18033989
Location15	visited	at	269.1803	2
Location14	visited	at	285.0113	5.830951895
Location47	visited	at	298.0113	3
Location17	visited	at	359	2
Location12	visited	at	377	8
Location11	visited	at	392	5
Location13	visited	at	407.831	5.830951895
Location87	visited	at	430.8694	13.03840481
Location59	visited	at	445.3415	4.472135955
Location86	visited	at	467.0462	11.70469991
Location57	visited	at	486.1016	9.055385138
Location58	visited	at	517.3148	21.21320344
Location77	visited	at	537.9449	10.63014581
Location49	visited	at	568.9687	21.02379604
Location20	visited	at	581.9687	3
Location22	visited	at	593.9687	2
Location74	visited	at	624.5843	20.61552813
Location97	visited	at	650.7088	16.1245155
Location10	visited	at	683.5123	22.8035085
Location60	visited	at	714.1278	20.61552813
Location68	visited	at	749.1278	25
Location0	visited	at	0	10

G.1 Perhitungan Manual (Lanjutan 3)

RUTE				JARAK
Route	3			317.9216523
Location0	visited	at	0	
Location29	visited	at	131	52.20153254
Location31	visited	at	143	2
Location30	visited	at	158	5
Location28	visited	at	172	4
Location33	visited	at	190.6023	8.602325267
Location63	visited	at	221.2179	20.61552813
Location76	visited	at	240.6518	9.433981132
Location51	visited	at	281	9.433981132
Location84	visited	at	304	9.219544457
Location85	visited	at	322.4853	8.485281374
Location81	visited	at	370.1814	37.69615365
Location71	visited	at	396.4603	16.2788206
Location93	visited	at	411.4603	5
Location94	visited	at	427.1171	5.656854249
Location67	visited	at	446.3367	9.219544457
Location27	visited	at	488.1172	31.78049716
Location26	visited	at	509	5
Location32	visited	at	564	8
Location34	visited	at	579.3852	5.385164807
Location50	visited	at	602.3852	13
Location96	visited	at	637.8803	25.49509757
Location91	visited	at	661.2967	13.41640786
Location80	visited	at	676.6818	5.385164807
Location0	visited	at	0	7.615773106
Route	4			234.2368364
Location0	visited	at	0	
Location21	visited	at	158	45
Location23	visited	at	170	2
Location19	visited	at	205	6.403124237
Location75	visited	at	252.3363	37.33630941
Location18	visited	at	338	39
Location48	visited	at	636	2
Location89	visited	at	671	25
Location25	visited	at	713	32
Location24	visited	at	733.4403	10.44030651
Location0	visited	at	0	35.05709629

G.2 Output Kode Program

```
run:
best solution:1409.3062046960397
best solution:
Route 0
Location0 visited at 0.0
Location92 visited at 14.7648230602334
Location95 visited at 31.47302699273277
Location62 visited at 50.69257145002565
Location72 visited at 90.75916420677147
Location40 visited at 233.0
Location39 visited at 248.0
Location38 visited at 260.0
Location41 visited at 276.40312423743285
Location42 visited at 292.2340761322782
Location44 visited at 304.2340761322782
Location61 visited at 333.9571590555942
Location100 visited at 358.09929467932517
Location90 visited at 388.97990769714625
Location56 visited at 416.6717137101004
Location64 visited at 433.74278152196587
Location66 visited at 451.8050392702644
Location0 visited at 0.0
Route 1
Location0 visited at 0.0
Location82 visited at 84.99999999999997
Location65 visited at 103.5440037453175
Location99 visited at 170.0
Location52 visited at 185.0990195135928
Location83 visited at 207.14061409238508
Location8 visited at 272.0
Location7 visited at 287.0
Location6 visited at 300.0
Location2 visited at 314.0
Location5 visited at 334.1980390271856
Location79 visited at 366.2207545727308
Location78 visited at 388.38627963332726
Location73 visited at 405.59738218425525
Location53 visited at 436.1886424662292
Location55 visited at 460.3307780899602
Location88 visited at 476.6553334102969
Location3 visited at 513.7293061516586
Location46 visited at 530.1324303890915
```