



INSTITUT TEKNOLOGI BANDUNG

**ALOKASI PETUGAS PENGUMPULAN DATA LAPANGAN SECARA
DINAMIS BERBASIS *CONTEXT***

THESIS

**ARIS PRAWISUDATAMA
23215131**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
PROGRAM STUDI MAGISTER TEKNIK ELEKTRO
BANDUNG
JANUARI 2016**



INSTITUT TEKNOLOGI BANDUNG

**ALOKASI PETUGAS PENGUMPULAN DATA LAPANGAN SECARA
DINAMIS BERBASIS *CONTEXT***

THESIS

**Diajukan sebagai salah satu syarat untuk memperoleh gelar
Master**

ARIS PRAWISUDATAMA

23215131

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
PROGRAM STUDI MAGISTER TEKNIK ELEKTRO
BANDUNG
JANUARI 2016**

HALAMAN PERSETUJUAN

Judul : Alokasi Petugas Pengumpulan Data Lapangan Secara Dinamis
Berbasis *Context*
Nama : Aris Prawisudatama
NIM : 23215131

Laporan Thesis ini telah diperiksa dan disetujui.

XX Januari 2016

Dr. I Gusti Bagus Baskara Nugraha
Pembimbing Thesis

HALAMAN PERNYATAAN ORISINALITAS

**Thesis ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar.**

Nama : Aris Prawisudatama
NIM : 23215131
Tanda Tangan :

Tanggal : XX Januari 2016

HALAMAN PENGESAHAN

Thesis ini diajukan oleh :
Nama : Aris Prawisudatama
NPM : 23215131
Program Studi : Magister Teknik Elektro
Judul Thesis : Alokasi Petugas Pengumpulan Data Lapangan Secara
Dinamis Berbasis *Context*

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Master pada Program Studi Magister Teknik Elektro, Fakultas Sekolah Teknik Elektro dan Informatika, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing : Dr. I Gusti Bagus Baskara Nugraha ()

Penguji : Prof. XXX ()

Penguji : Prof. XXXX ()

Penguji : Prof. XXXXXX ()

@todo

Jangan lupa mengisi nama para penguji.

Ditetapkan di : Bandung

Tanggal : XX Januari 2016

KATA PENGANTAR

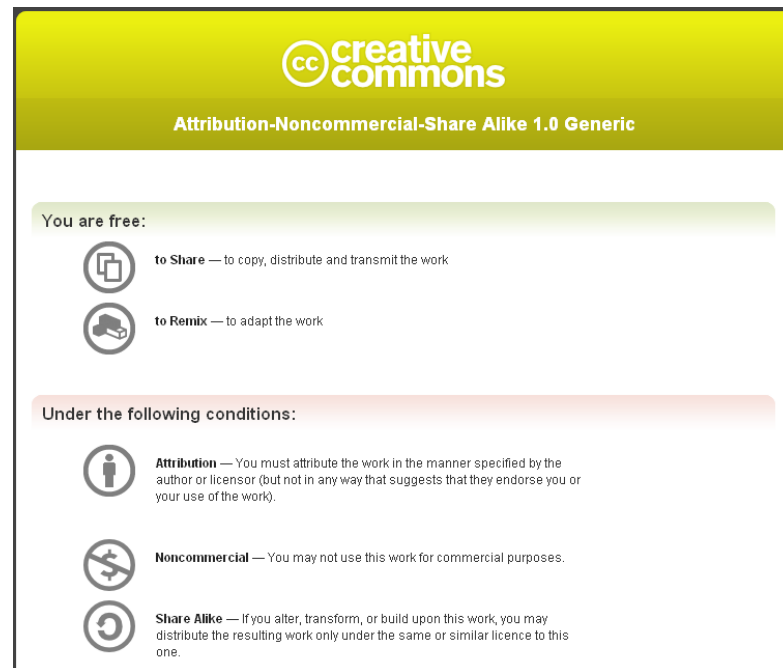
Template ini disediakan untuk orang-orang yang berencana menggunakan \LaTeX untuk membuat dokumen tugas akhirnya. Mengapa \LaTeX ? Ada banyak hal mengapa menggunakan \LaTeX , diantaranya:

1. \LaTeX membuat kita jadi lebih fokus terhadap isi dokumen, bukan tampilan atau halaman.
2. \LaTeX memudahkan dalam penulisan persamaan matematis.
3. Adanya otomatis dalam penomoran caption, bab, subbab, subsubbab, referensi, dan rumus.
4. Adanya otomatisasi dalam pembuatan daftar isi, daftar gambar, dan daftar tabel.
5. Adanya kemudahan dalam memberikan referensi dalam tulisan dengan menggunakan label. Cara ini dapat meminimalkan kesalahan pemberian referensi.

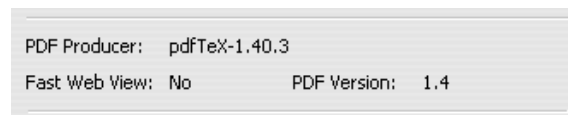
Template ini bebas digunakan dan didistribusikan sesuai dengan aturan *Creative Common License 1.0 Generic*, yang secara sederhana berisi:

Gambar 1 diambil dari http://creativecommons.org/licenses/by-nc-sa/1.0/deed.en_CA. Jika ingin mengetahui lebih lengkap mengenai *Creative Common License 1.0 Generic*, silahkan buka <http://creativecommons.org/licenses/by-nc-sa/1.0/legalcode>. Seluruh dokumen yang dibuat dengan menggunakan template ini sepenuhnya menjadi hak milik pembuat dokumen dan bebas didistribusikan sesuai dengan keperluan masing-masing. Lisensi hanya berlaku jika ada orang yang membuat template baru dengan menggunakan template ini sebagai dasarnya.

Dokumen ini dibuat dengan \LaTeX juga. Untuk meyakinkan Anda, coba lihat properti dari dokumen ini dan Anda akan menemukan bagian seperti Gambar 2. Dokumen ini dimaksudkan untuk memberikan gambaran kepada Anda seperti apa mudahnya menggunakan \LaTeX dan juga memperlihatkan betapa bagus dokumen yang dihasilkan. Seluruh url yang Anda temukan dapat Anda klik. Seluruh referensi yang ada juga dapat diklik. Untuk mengerti template yang disediakan, Anda tetap harus membuka kode \LaTeX dan bermain-main dengannya. Penjelasan dalam PDF ini masih bersifat gambaran dan tidak begitu mendetail, dapat dianggap sebagai



Gambar 1: *Creative Common License 1.0 Generic*



Gambar 2: Dokumen Dibuat dengan PDFLatex

pengantar singkat. Jika Anda merasa kesulitan dengan template ini, mungkin ada baiknya Anda belajar sedikit dasar-dasar \LaTeX .

Semoga template ini dapat membantu orang-orang yang ingin mencoba menggunakan \LaTeX . Semoga template ini juga tidak berhenti disini dengan ada kontribusi dari para penggunanya. Kami juga ingin berterima kasih kepada Andreas Febrian, Lia Sadita, Fahrurrozi Rahman, Andre Tampubolon, dan Erik Dominikus atas kontribusinya dalam template ini.

Bandung, XX Januari 2016

Aris Prawisudatama

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

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

Nama : Aris Prawisudatama
NPM : 23215131
Program Studi : Magister Teknik Elektro
Fakultas : Sekolah Teknik Elektro dan Informatika
Jenis Karya : Thesis

demikian pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (Non-exclusive Royalty Free Right)** atas karya ilmiah saya yang berjudul:

Alokasi Petugas Pengumpulan Data Lapangan Secara Dinamis Berbasis *Context*

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

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Bandung
Pada tanggal : XX Januari 2016
Yang menyatakan

(Aris Prawisudatama)

ABSTRAK

Nama : Aris Prawisudatama
Program Studi : Magister Teknik Elektro
Judul : Alokasi Petugas Pengumpulan Data Lapangan Secara Dinamis Berbasis *Context*

@todo

Tuliskan abstrak laporan disini.

Kata Kunci:

@todo

Tuliskan kata kunci yang berhubungan dengan laporan disini

ABSTRACT

Name : Aris Prawisudatama
Program : Magister Teknik Elektro
Title : Context-aware Dynamic Enumerator Allocation

@todo

Write your abstract here.

Keywords:

@todo

Write up keywords about your report here.

DAFTAR ISI

HALAMAN JUDUL	i
LEMBAR PERSETUJUAN	ii
LEMBAR PERNYATAAN ORISINALITAS	iii
LEMBAR PENGESAHAN	iv
KATA PENGANTAR	v
LEMBAR PERSETUJUAN PUBLIKASI ILMIAH	vii
ABSTRAK	viii
ABSTRACT	ix
DAFTAR ISI	x
DAFTAR GAMBAR	xiii
DAFTAR TABEL	xv
DAFTAR ALGORITMA	xvi
DAFTAR KODE	xvii
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	5
1.3 Tujuan Penelitian	5
1.4 Batasan Masalah	6
1.5 Manfaat dan Kontribusi	6
1.6 Sistematika Penulisan	6
2 STUDI LITERATUR	8
2.1 <i>Literature Map</i>	8
2.2 Badan Pusat Statistik	9

2.3	<i>Context-aware Computing</i>	10
2.3.1	Definisi <i>Context</i>	10
2.3.2	Kategori <i>Context</i>	10
2.3.2.1	<i>Individuality Context</i>	10
2.3.2.2	<i>Time Context</i>	11
2.3.2.3	<i>Location Context</i>	11
2.3.2.4	<i>Activity Context</i>	12
2.3.2.5	<i>Relations Context</i>	12
2.3.3	<i>Context-aware Computing</i>	12
2.4	<i>Location Routing</i>	13
2.4.1	<i>Vehicle Routing Problem</i>	15
2.4.2	<i>Multi-Depot Vehicle Routing Problem</i>	17
2.4.3	<i>Cooperative Evolution Strategy</i>	20
2.5	Messaging Solution	23
2.5.1	Mekanisme <i>Publish/Subscribe</i>	24
3	METODOLOGI	27
3.1	<i>Awareness of Problem</i>	28
3.2	<i>Suggestion</i>	28
3.3	<i>Development</i>	28
3.4	<i>Evaluation</i>	28
3.5	<i>Conclusion</i>	28
4	ANALISIS DAN PERANCANGAN	29
4.1	Analisis	29
4.1.1	Dataset	30
4.1.1.1	Lokasi Pencacahan	30
4.1.1.2	Petugas Pencacahan	30
4.1.1.3	Jarak dan Waktu Tempuh	31
4.1.2	Algoritma dan Implementasi	33
4.1.2.1	Definisi Masalah	33
4.1.2.2	Definisi Algoritma	33
4.1.2.3	Pencarian Solusi dan Penentuan Solusi Terbaik	35
4.1.3	Hasil dan Analisis	35
4.2	Perancangan Solusi	39
4.2.1	Garis Besar Sistem Usulan	40
4.2.2	<i>Recommendation Publisher</i>	41
4.2.3	<i>VRP Solver</i>	45

4.2.4	<i>Message Broker</i>	47
5	IMPLEMENTASI DAN PENGUJIAN	48
5.1	Implementasi	48
5.1.1	<i>VRP Solver</i>	48
5.1.2	<i>Publisher</i>	48
5.2	Pengujian	49
5.2.1	Lingkungan Pengujian	49
5.2.2	<i>Dataset dan Metric</i>	49
5.2.2.1	<i>Dataset</i>	49
5.2.2.2	<i>Metric</i>	51
5.2.3	Skenario dan Hasil Pengujian	52
5.2.3.1	<i>Message Broker Setup</i>	52
5.2.3.2	<i>Publisher Setup</i>	53
5.2.3.3	<i>Subscriber Setup</i>	53
5.2.3.4	Pengujian Tanpa <i>Service Time</i>	55
5.2.3.5	Pengujian Kondisi Normal dengan <i>Service Time</i> .	58
5.2.3.6	Pengujian Kondisi <i>Delay</i> pada Jaringan dengan <i>Service Time</i>	65
5.2.3.7	Pengujian Kondisi <i>Packet Loss</i> pada Jaringan de- ngan <i>Service Time</i>	70
6	KESIMPULAN DAN SARAN	75
6.1	Kesimpulan	75
6.2	Saran	75
	DAFTAR PUSTAKA	77
	DAFTAR PUSTAKA	77
	LAMPIRAN	1
	LAMPIRAN 1	1

DAFTAR GAMBAR

1	<i>Creative Common License 1.0 Generic</i>	vi
2	Dokumen Dibuat dengan PDFLatex	vi
1.1	Pembagian Blok Sensus dalam Desa/Kelurahan	1
1.2	Ilustrasi <i>Multi Depot Vehicle Routing Problem</i>	2
1.3	<i>Timeline</i> Sebelum dan Setelah Dikunjungi	4
1.4	<i>Timeline</i> Solusi Bertahap	4
2.1	<i>Literature Map</i> Penelitian	8
2.2	Kategori dari <i>Context</i>	11
2.3	Ilustrasi <i>Vehicle Routing Problem (VRP)</i>	16
2.4	Hierarki variasi dari VRP (Weise et al., 2009)	17
2.5	Ilustrasi <i>Multi Depot VRP</i>	17
2.6	<i>Lifecycle</i> pada <i>Cooperative Coevolution</i> (de Oliveira et al., 2016) . .	23
2.7	Arsitektur dasar pada Pub/Sub (Eugster et al., 2003)	25
2.8	Pemisahan informasi pada Pub/Sub (Eugster et al., 2003)	26
3.1	Tahapan <i>Design Science Research Methods and Patterns</i>	27
4.1	Google Direction API Response	32
4.2	Grafik Rekomendasi dengan MDVRP	36
4.3	Penerapan MDVRP pada Permasalahan Alokasi Petugas Pencacahan	38
4.4	Penerapan <i>real time</i> MDVRP pada Permasalahan Alokasi Petugas Pencacahan	40
4.5	Garis Besar Sistem Usulan	41
4.6	<i>Flowchart Topic Watcher</i>	42
4.7	<i>Algoritma Global Best Solution</i>	44
4.8	<i>VRP Worker</i>	45
4.9	Arsitektur <i>Publish-Subscribe</i> terdistribusi	47
5.1	Ilustrasi Rute yang Dihasilkan	52
5.2	<i>Flowchart</i> Subsistem pada Pengujian	56
5.3	Perbandingan Waktu Total dari 10 <i>Instance</i> Cordeau	57
5.4	Perbandingan Standar Deviasi Waktu dari 10 <i>Instance</i> Cordeau . . .	58

5.5	Perbandingan Rute Hasil Pengujian Data Cordeau P01 Dengan <i>Service Time</i>	60
5.6	Rute Hasil Pengujian Data Lapangan Pada MDVRP berbasis CoEAs	63
5.7	Rute Hasil Pengujian Data Lapangan Pada MDVRP berbasis CoEAs dengan Publish/Subscribe	64
5.8	Flowchart Setup Pumba Net Emulator	65
5.9	Rute Hasil Pengujian Kondisi <i>Network Delay</i> Data Lapangan Pada MDVRP berbasis CoEAs	68
5.10	Rute Hasil Pengujian Kondisi <i>Network Delay</i> Data Lapangan Pada MDVRP berbasis CoEAs dengan Publish/Subscribe	69
5.11	Flowchart Setup Pumba Net Emulator	70
5.12	Rute Hasil Pengujian Kondisi <i>Packet Loss</i> Data Lapangan Pada MDVRP berbasis CoEAs	73
5.13	Rute Hasil Pengujian Kondisi <i>Packet Loss</i> Data Lapangan Pada MDVRP berbasis CoEAs dengan Publish/Subscribe	74
1	Perbandingan Rute Hasil Pengujian Data Cordeau P01 Tanpa <i>Service Time</i>	1
2	Perbandingan Rute Hasil Pengujian Data Cordeau P02 Tanpa <i>Service Time</i>	2
3	Perbandingan Rute Hasil Pengujian Data Cordeau P03 Tanpa <i>Service Time</i>	3
4	Perbandingan Rute Hasil Pengujian Data Cordeau P04 Tanpa <i>Service Time</i>	4
5	Perbandingan Rute Hasil Pengujian Data Cordeau P05 Tanpa <i>Service Time</i>	5
6	Perbandingan Rute Hasil Pengujian Data Cordeau P06 Tanpa <i>Service Time</i>	6
7	Perbandingan Rute Hasil Pengujian Data Cordeau P07 Tanpa <i>Service Time</i>	7
8	Perbandingan Rute Hasil Pengujian Data Cordeau P08 Tanpa <i>Service Time</i>	8
9	Perbandingan Rute Hasil Pengujian Data Cordeau P09 Tanpa <i>Service Time</i>	9
10	Perbandingan Rute Hasil Pengujian Data Cordeau P10 Tanpa <i>Service Time</i>	10

DAFTAR TABEL

4.1	Lokasi Pencacahan	30
4.2	Pencacah	31
4.3	Data Jarak dan Waktu Tempuh	32
4.4	Total Waktu Setiap Pencacah dengan MDVRP	36
5.1	Hasil Pengujian Tanpa <i>Service Time</i> pada Data Cordeau	56
5.2	Hasil Pengujian Kondisi Normal (Cordeau), MDVRP berbasis CoEAs .	59
5.3	Hasil Pengujian Kondisi Normal (Cordeau), MDVRP berbasis Co- EAs dan Publish/Subscribe	59
5.4	Komparasi Hasil Pengujian Kondisi Normal Dengan Data Cordeau .	59
5.5	Hasil Pengujian Kondisi Normal (Lapangan), MDVRP berbasis Co- EAs	61
5.6	Hasil Pengujian Kondisi Normal (Lapangan), MDVRP berbasis Co- EAs dan Publish/Subscribe	62
5.7	Komparasi Hasil Pengujian Kondisi Normal Dengan Data Lapangan	62
5.8	Hasil Pengujian Kondisi Network Delay, MDVRP berbasis CoEAs .	66
5.9	Hasil Pengujian Kondisi Network Delay, MDVRP berbasis CoEAs dan Publish/Subscribe	67
5.10	Komparasi Hasil Pengujian Kondisi Network Delay Dengan Data Lapangan	67
5.11	Hasil Pengujian Kondisi <i>Packet Loss</i> , MDVRP berbasis CoEAs . . .	71
5.12	Hasil Pengujian Kondisi <i>Packet Loss</i> , MDVRP berbasis CoEAs dan Publish/Subscribe	72
5.13	Komparasi Hasil Pengujian Kondisi <i>Packet Loss</i> Dengan Data La- pangan	72

DAFTAR ALGORITMA

2.1	Algoritma Evolution Strategy	22
4.1	TopicWatcher	42
4.2	VRPSolver Procedure	44

DAFTAR KODE

4.1	Google Direction API Request	32
4.2	Definisi Lokasi Pencacahan	33
4.3	Definisi Pencacah dari File .csv	34
4.4	Definisi Penimbang Jarak dan Waktu Tempuh dari File .csv	34
4.5	Build Problem	34
4.6	Penentuan Algoritma	35
4.7	Pencarian Solusi	35
4.8	Rekomendasi dengan MDVRP	37
5.1	Konfigurasi Redis Cluster	53
5.2	Pembuatan Redis Cluster	53
5.3	Respon Pembuatan Redis Cluster	54
5.4	Format penggunaan Publisher	55

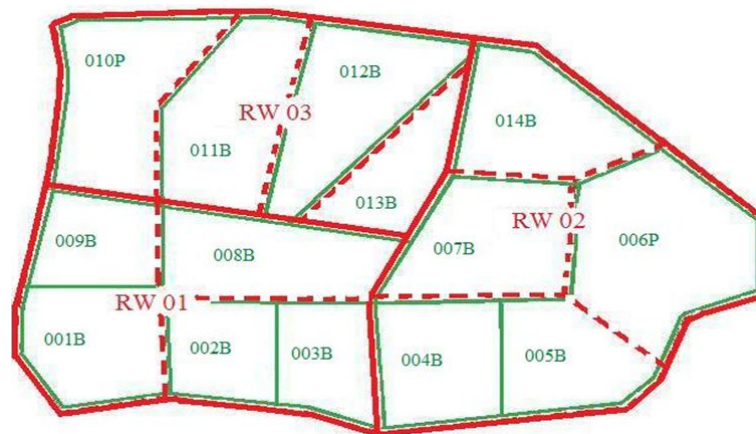
BAB 1

PENDAHULUAN

1.1 Latar Belakang

Badan Pusat Statistik (BPS) merupakan suatu lembaga pemerintah non-departemen yang bertanggung jawab dalam penyediaan statistik dasar (BPS, 2016a). Dalam peranannya sebagai penyedia data, BPS melakukan pengumpulan data dengan 2 (dua) metode, yaitu primer dan sekunder. Pengumpulan data primer adalah pengumpulan data dengan menggunakan metode wawancara langsung dengan responden, baik responden individu, rumah tangga, maupun perusahaan. Sementara pengumpulan data sekunder adalah pengumpulan data dengan memanfaatkan data yang telah dikumpulkan oleh pihak lain.

Pada pengumpulan data primer oleh BPS, selanjutnya disebut dengan pencacahan, suatu wilayah administratif dibagi dalam beberapa Blok Sensus (BS), dimana setiap Blok Sensus merupakan wilayah kerja dari seorang pencacah (BPS, 2016b). Setiap pencacah dapat memiliki wilayah kerja lebih dari satu Blok Sensus. Setiap Blok Sensus akan dikunjungi oleh seorang pencacah yang akan melakukan pencacahan terhadap rumah tangga didalamnya yang menjadi sampel pencacahan.



Gambar 1.1: Pembagian Blok Sensus dalam Desa/Kelurahan

Pembagian wilayah kerja pada pencacah terkang bisa menjadi sesuatu yang tidak mudah. Subject matter dalam melakukan pembagian wilayah kerja harus mempertimbangkan berbagai hal, antara lain:

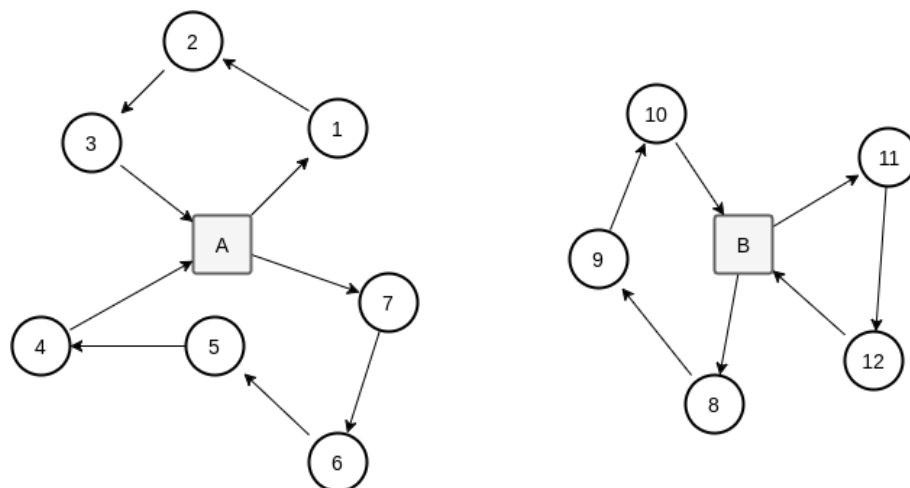
1. Total waktu dan total biaya harus minimal,

2. Setiap pencacah harus dialokasikan ke wilayah yang relatif dekat dengan domisili atau kantor.

Kesalahan dalam pengalokasian wilayah kerja dapat berakibat pada total waktu penyelesaian untuk masing-masing pencacah menjadi tidak merata. Kondisi ini dapat berakibat pada keterlambatan kegiatan pencacahan secara keseluruhan.

Terdapat 2 (dua) faktor yang menyusun total waktu pencacahan, yaitu lama tempuh antar wilayah kerja dan lama pencacahan dari seluruh wilayah kerja. Lama pencacahan dalam sebuah wilayah kerja tersusun atas lama perpindahan antar rumah tangga dan lama wawancara pada seluruh rumah tangga di wilayah kerja. Berdasarkan (Sudman, 1965), waktu yang dihabiskan oleh seorang pencacah tersusun atas: 21 persen untuk berpindah antar wilayah kerja, 15 persen untuk berpindah antar rumah tangga, 15 persen untuk keseluruhan wawancara, dan sisanya untuk kegiatan lain seperti pengenalan wilayah dan perbaikan data.

Kondisi yang terjadi dalam permasalahan pembagian wilayah kerja pencacahan memiliki kesamaan dengan permasalahan VRP (Vehicle Routing Problem) atau lebih spesifiknya MDVRP (Multi Depot Vehicle Routing Problem). VRP merupakan suatu permasalahan yang bertujuan untuk menentukan rute terbaik yang harus ditempuh oleh beberapa kendaraan kurir untuk mengirimkan barang kepada konsumen (Dantzig and Ramser, 1959). Sementara MDVRP adalah variasi dari VRP dimana terdapat lebih dari satu depot yang digunakan, dimana depot merupakan lokasi dimana kendaraan memulai dan mengakhiri perjalanan.



Gambar 1.2: Ilustrasi *Multi Depot Vehicle Routing Problem*

Penyelesaian permasalahan MDVRP bertujuan untuk mendapatkan solusi dengan total biaya terkecil. Solusi yang dihasilkan terdiri dari sejumlah rute, dimana setiap rute terdiri dari sejumlah konsumen yang harus dikunjungi oleh sebuah

kendaraan. Penyelesaian permasalahan MDVRP harus mengikuti beberapa aturan, yaitu:

1. Setiap konsumen dikunjungi hanya sekali dan oleh satu kendaraan,
2. Semua kendaraan memulai dan mengakhiri perjalanan pada sebuah depot yang sama,
3. Total *demand* dari seluruh konsumen pada setiap rute tidak melebihi kapasitas dari kendaraan,

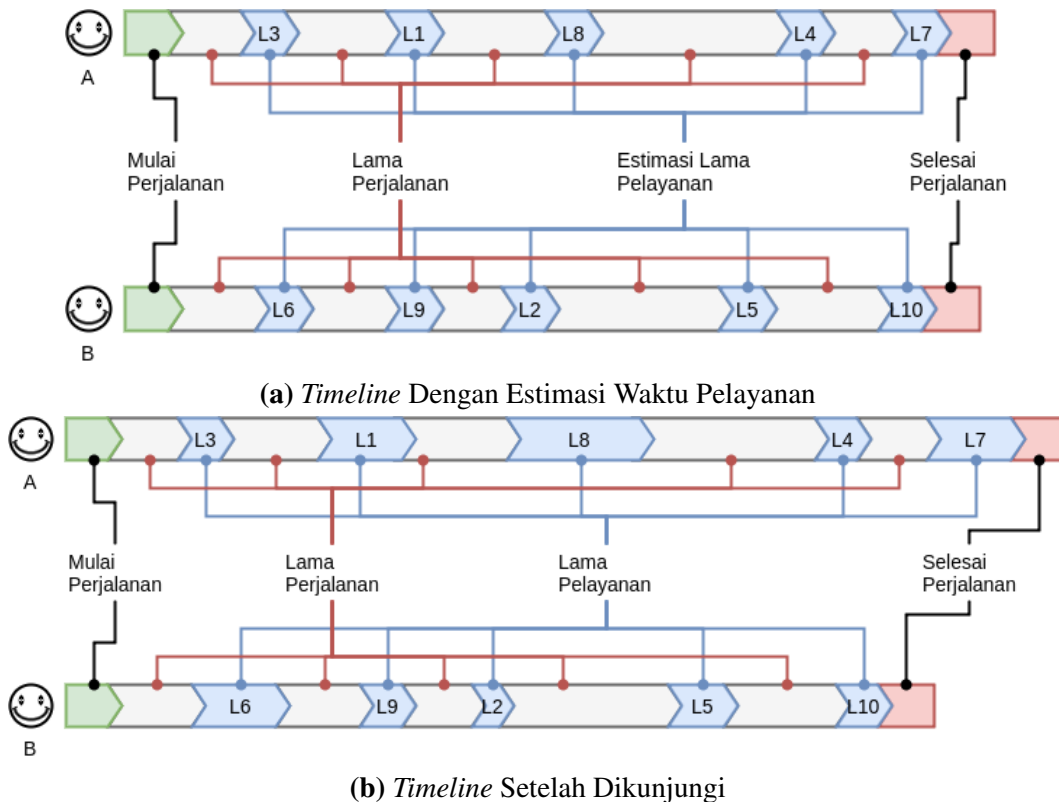
Karena mempunyai kesamaan tujuan dan aturan, permasalahan pembagian wilayah kerja dapat dianggap sebagai permasalahan terkait MDVRP. Pada permasalahan pembagian wilayah kerja, pencacah dapat dianalogikan dengan kendaraan, wilayah kerja dapat dianalogikan dengan konsumen, dan lokasi dimana pencacahan memulai perjalanan (rumah atau kantor) dapat dianalogikan dengan depot.

Pada permasalahan MDVRP terdapat 2 (dua) variabel yang dapat mempengaruhi solusi atau rute yang dihasilkan, yaitu biaya tempuh antar konsumen dan lama pelayanan pada setiap konsumen. Biaya tempuh biasanya didefinisikan dalam satuan waktu, sehingga biaya tempuh dapat didekati dengan lama tempuh. Kedua variabel ini identik dengan lama tempuh antar wilayah kerja dan lama pencacahan dari setiap wilayah kerja pada permasalahan pembagian wilayah kerja.

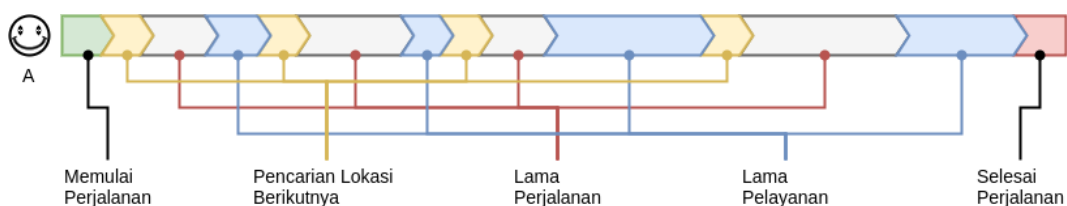
Terdapat beberapa pendekatan yang dapat digunakan untuk mengestimasi lama tempuh antar wilayah kerja, seperti melalui perkiraan, survei, dan menggunakan service seperti Google Direction API. Akan tetapi, lama pelayanan pada setiap wilayah kerja tidak dapat diketahui, sampai dengan wilayah kerja tersebut selesai dicacah. Hal ini berpotensi menyebabkan rute-rute yang dihasilkan memiliki standar deviasi yang tinggi. Figure 4.3 memberikan ilustrasi potensi masalah ini:

1. Solusi yang diperoleh dengan menggunakan estimasi lama pelayanan pada setiap wilayah kerja menghasilkan rute terbaik yang relatif *equal* dari segi total waktu (Figure 1.3a),
2. Setelah semua wilayah kerja dikunjungi, ternyata lama pelayanan pada masing-masing wilayah kerja bervariasi, sehingga variasi total waktu pencacahan menjadi besar (Figure 1.3b).

Agar rute yang dihasilkan lebih *equal* dari sisi total waktu pencacahan, solusi yang ditawarkan adalah dengan menggunakan mekanisme pencarian solusi secara bertahap, seperti digambarkan pada Figure 1.4. Setiap tahap pencarian solusi tidak



Gambar 1.3: Timeline Sebelum dan Setelah Dikunjungi



dimaksudkan untuk mendapatkan rute secara keseluruhan, tetapi hanya digunakan untuk menentukan wilayah kerja yang akan dikunjungi berikutnya.

Agar 'wilayah kerja berikutnya' yang dihasilkan lebih akurat, maka pencarian solusi harus dilakukan secara tersentral dengan melibatkan *context* dari seluruh pencacah. Dalam sistem komputer tidak terdapat definisi *context* yang tunggal. Sebagian besar definisi sepakat bahwasannya *context* adalah sesuatu yang harus dilakukan terkait interaksi pengguna dan sistem komputer (Chen et al., 2000). (Schilit et al., 1994) dan (Schmidt et al., 1999), misalnya, mendefinisikan *context* sebagai pengetahuan tentang *state* dari *user* dan *IT device*, termasuk lingkungan, situasi, dan lokasi. Sementara (Abowd et al., 1999) mendefinisikan *context* sebagai segala informasi yang dapat digunakan untuk mengkarakterisasi kondisi dari suatu entitas.

Pencarian solusi secara tersentral membutuhkan mekanisme komunikasi antara

client yang merupakan pencacah yang melakukan *request* 'wilayah kerja berikutnya' dan *server* dimana *request* diproses dan 'wilayah kerja berikutnya' dicari. Terdapat berbagai alternatif mekanisme komunikasi *client-server* yang dapat digunakan, seperti *Web service*, *Request/Reply*, dan *Publish/Subscribe*. Mekanisme yang tepat diperlukan agar komunikasi antara *client* dan *server* dapat dilakukan dalam berbagai kondisi jaringan komunikasi.

Dengan adanya permasalahan yang telah dijabarkan sebelumnya, maka penelitian ini akan merancang sebuah sistem yang dapat digunakan untuk membuat rekomendasi lokasi pencacahan. Peneliti mengusulkan penggunaan mekanisme pencarian solusi secara bertahap dengan melibatkan *context* dari pencacah.

1.2 Rumusan Masalah

Berdasarkan latar belakang permasalahan yang telah diuraikan sebelumnya, maka dapat dirumuskan masalah dalam penelitian ini adalah bagaimana merancang sebuah sistem rekomendasi lokasi pencacahan sesuai dengan *context* dari pencacah.

Adapun detail dari permasalahan yang akan dikaji adalah sebagai berikut:

- Bagaimana menentukan *context* dari pencacah yang sesuai digunakan dalam kasus rekomendasi lokasi pencacahan?
- Bagaimana menyusun algoritma pencarian rekomendasi lokasi dengan memanfaatkan *context* dari pencacah?
- Bagaimana menyusun mekanisme *conflict resolution*, agar system tidak merekomendasikan lokasi yang sama pada dua atau lebih pencacah?
- Apa mekanisme *real-time* yang sesuai digunakan dalam berbagai kondisi jaringan komunikasi yang bervariasi?

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah diatas, maka dapat ditentukan tujuan utama dari penelitian ini adalah untuk merancang sebuah arsitektur sistem rekomendasi lokasi pencacahan secara dinamis dan *real-time*.

Adapun tujuan khusus dari penelitian ini adalah:

- Menentukan *context* yang sesuai dengan kasus rekomendasi lokasi pencacahan.

- Menyusun algoritma rekomendasi lokasi sehingga sistem memberikan rekomendasi terbaik secara global.
- Menyusun algoritma *location conflict* sehingga sistem tidak memberikan duplikasi rekomendasi.
- Menganalisis dan mengimplementasikan mekanisme komunikasi yang tepat digunakan pada sistem.

1.4 Batasan Masalah

Masalah dalam penelitian ini memiliki batasan sebagai berikut:

- Lokasi pencacahan yang menjadi rujukan adalah Blok Sensus (BS) yang dikeluarkan oleh Badan Pusat Statistik (BPS).
- Penelitian tidak berfokus pada algoritma MDVRP, sehingga pengujian hanya dilakukan dengan membandingkan dengan salah satu algoritma MDVRP.

1.5 Manfaat dan Kontribusi

Hasil penelitian ini diharapkan dapat memberikan kontribusi kepada BPS khususnya, yaitu memudahkan *subject matter* dalam melakukan alokasi petugas pencacahan. Ketepatan alokasi petugas memiliki implikasi pada meratanya beban setiap pencacah serta meratanya waktu penyelesaian pencacahan.

Selain itu, penelitian ini juga memberikan implikasi dalam bidang akademis, yaitu sistem tidak hanya dapat digunakan secara spesifik untuk kasus pengumpulan data, tetapi juga dapat digunakan pada kasus yang terkait dengan *Vehicle Routing Problem*.

1.6 Sistematika Penulisan

Sistematika penulisan tesis ini terdiri atas enam bab dengan perincian sebagai berikut:

BAB I PENDAHULUAN. Pada bab ini dijelaskan fenomena dan masalah yang diangkat dalam penelitian ini. Selain itu dipaparkan pula mengenai rumusan masalah, tujuan penelitian, dan kontribusi penelitian.

BAB II STUDI LITERATUR. Bab ini menjelaskan landasan teori yang digunakan dalam penelitian, teknologi pendukung, serta penelitian terkait dengan penelitian yang dilakukan.

BAB III METODOLOGI PENELITIAN. Pada bagian ini dijelaskan metode penelitian yaitu alur dan langkah-langkah dalam melakukan penelitian.

BAB IV ANALISIS DAN PERANCANGAN. Bab ini memaparkan analisis terhadap permasalahan yang dihadapi, serta kriteria yang dibutuhkan pada algoritma yang dirancang. Dari analisis permasalahan, diuraikan pendekatan yang dibuat yang merupakan solusi atas permasalahan yang dikemukakan.

BAB V IMPLEMENTASI DAN PENGUJIAN. Proses dan hasil implementasi rancangan dijelaskan dalam bab ini. Proses pengujian diuraikan mulai dari skenario pengujian hingga hasil pengujian.

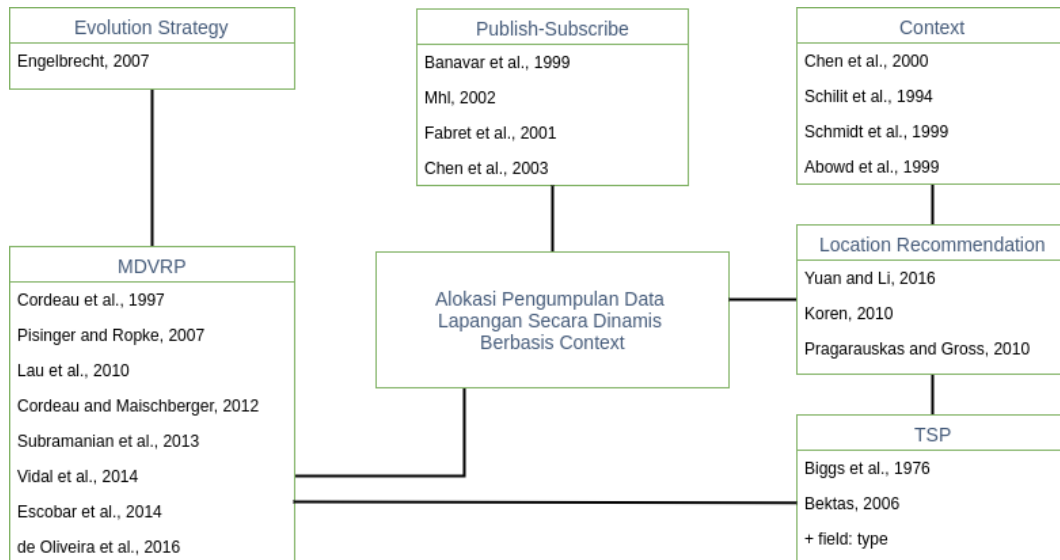
BAB VI KESIMPULAN DAN SARAN. Bab ini memaparkan kesimpulan yang diperoleh dari penelitian ini, serta saran untuk pihak terkait dan penelitian selanjutnya yang berkaitan dengan topik penelitian ini

BAB 2

STUDI LITERATUR

2.1 *Literature Map*

Literature Map merupakan sebuah visualisasi yang memuat *summary* penelitian terkait yang telah dilakukan sebelumnya. *Literature Map* biasanya direpresentasikan dalam sebuah gambar, misalnya menggunakan struktur hierarki *top-down* dalam mempresentasikan literatur (Creswell, 2013). Ide utama dari penggambaran *literature map* adalah untuk membuat visualisasi dari penelitian terdahulu tentang sebuah topik.



Gambar 2.1: *Literature Map* Penelitian

Berdasarkan review dari literatur yang terkait dengan penelitian ini, arsitektur dari rekomendasi lokasi pada pengumpulan data lapangan dapat dibagi menjadi 3 (tiga) kelompok: *context-aware computing*, *location routing*, dan mekanisme komunikasi.

Context-aware computing akan membahas beberapa literatur yang membahas definisi dari konteks itu sendiri dan pemanfaatannya dalam komputasi. *Location routing* akan membahas literatur terkait beberapa algoritma yang biasa digunakan dalam penyelesaian masalah *location routing*. Sementara pada mekanisme komunikasi antara *client* dan *server* akan dibahas beberapa mekanisme komunikasi yang dapat digunakan.

2.2 Badan Pusat Statistik

Badan Pusat Statistik (BPS) adalah Lembaga Pemerintah Non-Departemen yang bertanggung jawab dan memiliki peranan dalam penyediaan data. Berdasarkan Rencana Strategis BPS 2015-2019, BPS memiliki misi yang harus dijalankan yaitu:

1. Menyediakan data statistik berkualitas melalui kegiatan statistik yang terintegrasi dan berstandar nasional maupun internasional.
Dalam rangka menyediakan data berkualitas, data yang dihasilkan BPS harus memenuhi dimensi kualitas yakni relevan, akurat, disajikan tepat waktu, koheren, dapat diakses, dan dapat diinterpretasikan.
2. Memperkuat Sistem Statistik Nasional yang berkesinambungan melalui pembinaan dan koordinasi di bidang statistik.
3. Membangun insan statistik yang profesional, berintegritas dan amanah untuk kemajuan perstatistikan. Dalam menyelenggarakan kegiatan statistik, insan statistik harus memiliki kapasitas dan kapabilitas yang diperlukan untuk menghasilkan data statistik yang berkualitas.

Di dalam menjalankan peranan sebagai penyedia data, BPS menyelenggarakan statistik dasar dengan cara sensus, survei, dan kompilasi administrasi untuk mendapatkan data. Data yang berkualitas hanya dapat diperoleh melalui proses yang berkualitas pula. Terdapat berbagai variabel yang menjadi parameter proses yang berkualitas, salah satunya adalah ketepatan waktu dalam proses pengumpulan data.

Pengumpulan data, pada prosesnya merupakan suatu kondisi yang sangat fleksibel. Terdapat banyak faktor yang dapat mempengaruhi ketepatan waktu pengumpulan data, antara lain: kualitas petugas pengumpulan data, kualitas responden, jumlah anggota rumah tangga yang didata, jarak antar lokasi pencacahan, jarak antar rumah tangga dalam satu segmen, kondisi alam, dan lain sebagainya. Alokasi petugas pengumpulan data secara statis tidak dapat mengakomodir faktor-faktor tersebut. Akibatnya, waktu penyelesaian pengumpulan data bisa bervariasi antara petugas satu dengan petugas lainnya. Oleh karena itu, alokasi petugas secara fleksibel sesuai dengan konteks diperlukan sebagai antisipasi terhadap kondisi yang tidak diperkirakan.

2.3 Context-aware Computing

2.3.1 Definisi Context

Semenjak *Context-aware computing* pertama kali diperkenalkan oleh (Schilit et al., 1994), berbagai definisi mengenai *context* dan *context-awareness* telah berkembang luas. Sebagian besar definisi yang ada saat ini, menurut (Zimmermann et al., 2007), dapat dikelompokkan menjadi dua: definisi menurut sinonim dan definisi menurut contoh. *Context* mengalami berbagai pengkarakteristikan menggunakan sinonim seperti *application environment* (Hull et al., 1997) atau situasi (Brown, 1995). Sementara beberapa authors, seperti (Brown et al., 1997), (Gross and Specht, 2001), dan (Ryan et al., 1999), mendefinisikan *context by example* dan *context element* seperti lokasi, identitas, waktu, suhu, kebisingan sama dengan kepercayaan, keinginan, komitmen, dan hubungan dengan manusia (Chen et al., 2003b).

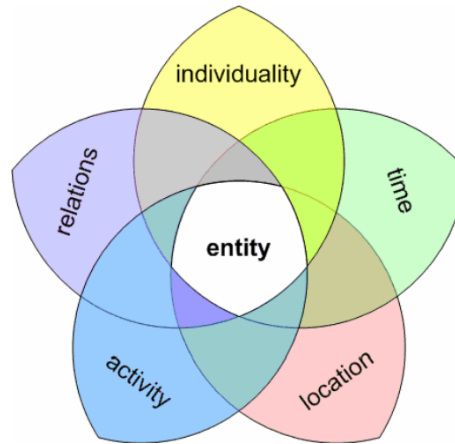
Secara umum, dapat dikatakan bahwasanya *context* adalah informasi yang dapat digunakan dalam menjelaskan situasi dari sebuah entitas. Entitas dapat berupa manusia, lokasi, atau obyek yang relevan dengan interaksi antara seorang pengguna dan aplikasi, termasuk pengguna dan aplikasi itu sendiri (Dey, 2001). (Schilit et al., 1994) menyebutkan bahwasannya yang termasuk ke dalam *context* adalah: lokasi dari penggunaan, kumpulan orang-orang sekitar, *hosts*, dan *accessible devices*, serta perubahan hal-hal tersebut dari waktu ke waktu. (Dey, 2001) mengembangkan definisi dari (Schilit et al., 1994) dengan menyatakan "Context adalah lokasi, identitas dan status dari individu, kelompok, dan obyek komputasi".

2.3.2 Kategori Context

Dari berbagai informasi yang menjelaskan entitas dari *context*, semuanya mengarah ke 5 (lima) kategori, yaitu: *individuality*, aktivitas, lokasi, waktu, dan relasi, seperti pada Gambar 2.2 (Zimmermann et al., 2007). Kategori *individuality* memuat properti dan atribut yang menjelaskan entitas itu sendiri. Kategori aktivitas mencakup seluruh kegiatan yang dilakukan entitas tersebut. Kategori lokasi dan waktu menyediakan koordinat *spatio-temporal* dari entitas. Sementara kategori relasi merepresentasikan informasi tentang hubungan antara sebuah entitas dengan entitas yang lain.

2.3.2.1 Individuality Context

Kategori ini memberi akses kepada informasi yang bersifat kontekstual tentang entitas terkait. Informasi ini dapat berupa apa saja terkait entitas tersebut, tetapi umum-



Gambar 2.2: Kategori dari *Context*

nya adalah *state* dari entitas. Entitas dapat berupa entitas individu atau sekelompok entitas yang saling berbagi *context* yang terkait. Entitas dapat berupa entitas yang *real* (eksis secara nyata), maupun virtual (hanya terdapat di lingkup informasi). Selain itu, terdapat juga entitas yang bersifat *mobile*, *movable*, maupun *fixed*. kategori *individuality context* dikelompokkan menjadi empat, yaitu: *natural*, *human*, *artificial*, dan *group entities*.

2.3.2.2 *Time Context*

Waktu merupakan aspek yang vital dalam klasifikasi *context*, karena sebagian besar pernyataan sangat terkait dengan dimensi *temporal*. Kategori ini mengkalkulasi informasi seperti *time zone* dari *client*, dan *current time* atau *virtual time*. Contoh representasi dari waktu adalah *time zone*, misalnya format *Central European Time (CET)*, yang menyediakan fasilitas kalkulasi secara matematis dan komparasi waktu. Model dimensi waktu yang sering diimplementasikan dalam *context-aware computing* misalnya jam kerja atau hari libur.

Context yang disimpan secara persisten dapat membentuk sebuah *data pool* yang berisi histori dari informasi terkait *context* tersebut. Histori tersebut membentuk basis data untuk *context information* dari event yang telah lampau. Basis data yang dipersistenkan dapat digunakan sebagai bahan analisis dari kebiasaan dari pengguna dan prediksi.

2.3.2.3 *Location Context*

Dengan semakin berkembangnya *portable/mobile devices*, lokasi menjadi parameter yang penting dalam *context-aware system*. Obyek-obyek fisik dan perangkat tersusun secara spasial, dipengaruhi oleh pergerakan pengguna. Kategori ini berisi

model lokasi yang terklasifikasi secara fisik maupun virtual, misalnya *IP address* sebagai posisi komputer dalam sebuah jaringan, koordinat, maupun informasi spasial terkait seperti kecepatan dan orientasi. Lokasi dapat berupa lokasi absolut atau lokasi relatif, yaitu lokasi yang diperhitungkan dari objek yang lain. Model dapat juga terdiri dari lokasi kuantitatif (geometrik) dan kualitatif (simbol).

Lokasi yang bersifat kuantitatif merujuk kepada koordinat dengan dua dimensi, tiga dimensi, atau lebih. Sebagai contoh, koordinat dua dimensi secara geografis melambangkan setiap lokasi di bumi secara lintang dan bujur. Informasi terkait koordinat geografis biasanya diperoleh melalui *Global Positioning System* dengan menggunakan satelit. Selain itu, lokasi juga dapat diklasifikasikan menjadi dalam dan luar ruangan, sinyal radio atau cahaya, dll. Sementara itu, lokasi yang bersifat kualitatif dan berupa bangunan, ruangan, jalan, negara, dan sebagainya.

2.3.2.4 Activity Context

Context aktifitas meliputi aktifitas yang sedang dikerjakan oleh sebuah entitas. Kategori ini dapat diterjemahkan menjadi: tujuan, kegiatan, dan aksi. Sebuah *task* merupakan aktifitas yang berorientasi pada capaian. Capaian atau *goals* dapat bersifat *low-level*, dimana *goal* dapat sering berganti, dan *high-level*, dimana *goal* bersifat konsisten.

2.3.2.5 Relations Context

Kategori ini mencakup relasi dari sebuah entitas yang berhubungan dengan entitas yang lain. Entitas yang lain tersebut dapat berupa manusia, benda, layanan, atau informasi (misalnya teks, gambar, film, suara). Karakteristik dari *environment* biasanya dibangun secara spasial dan temporal. Individu dalam kelompok saling mempengaruhi kelompok dalam satu relasi, misalnya seluruh manusia dengan umur yang sama.

2.3.3 Context-aware Computing

(Dey, 2001) menjelaskan, sebuah sistem dikatakan *context-aware* jika dia menggunakan *context* untuk menyediakan informasi atau layanan yang relevan kepada pengguna, dimana relevansinya tergantung dari apa yang pengguna kerjakan.

Terdapat berbagai contoh sistem yang bersifat *context-aware*. (Tsai et al., 2016) misalnya, menggunakan *context* untuk menciptakan *smart home environment*. (Magara et al., 2016) menggabungkan beberapa *context* seperti: lokasi dan aktivitas pengguna, preferensi, label, dan *tags* untuk membuat rekomendasi musik yang akan

diputar. Sementara (Said et al., 2013) menciptakan rekomendasi film berdasarkan *time, mood, dan social recommendation*.

Pada aplikasi yang bersifat *mobile, context* biasanya diperoleh dengan menggunakan sensor yang tersemat dalam *mobile device* tersebut. Sekarang ini, telah banyak ditemui *mobile device*, terutama *programmable smartphone*, yang telah dilengkapi dengan berbagai sensor (Cao et al., 2015). (Do and Gatica-Perez, 2011), misalnya mem-*propose* GroupUs, framework yang mengelompokkan pengguna berdasarkan aktifitas sehari-hari yang dikumpulkan dengan menggunakan *proximity sensor*. (Dai et al., 2010a) menciptakan *drunk driving detection* dengan memanfaatkan sensor *accelerometer* yang tersemat dalam *smartphone*. Sementara (Zou et al., 2016), memanfaatkan sensor *Global Positioning System* (GPS) dan *Micro-Electro-Mechanical System* (MEMS) untuk membuat rekomendasi transportasi. Begitu pula sensor-sensor yang lain juga telah dimanfaatkan dalam berbagai penelitian (Dai et al., 2010b; Lu et al., 2009; Bao and Roy Choudhury, 2010; Rubel et al., 2005; Atzmueller and Hilgenberg, 2013).

Pada kasus yang dijadikan dasar dari penelitian ini, yaitu terkait pengumpulan data lapangan, *context* yang akan digunakan adalah *spatio-temporal context*, yang meliputi lokasi dan waktu dari setiap pencacah.

2.4 Location Routing

Location routing problem, menurut (Nagy and Salhi, 2007), secara umum dapat dikatakan sebagai sebuah pendekatan dalam pemodelan dan penyelesaian permasalahan terkait lokasi. Definisi ini mengikuti definisi (Bruns, 1998), yaitu perencanaan lokasi yang mempertimbangkan perencanaan *tour*. Definisi ini juga senada dengan (Balakrishnan et al., 1987), yang menyatakan bahwa *location routing problem* merupakan keputusan strategis yang fokus terhadap lokasi fasilitas.

Lokasi dari fasilitas dan *vehicle routing* merupakan area yang saling terkait. (Maranzana, 1964) menyatakan bahwa "*the location of factories, warehouses and supply points in general...is often influenced by transport costs*". Tetapi beberapa peneliti, sebagaimana (Nagy and Salhi, 2007), menolak korelasi tersebut dengan beberapa alasan:

1. Terdapat beberapa kondisi dimana *location problem* tidak memiliki aspek *routing*,
2. Permasalahan lokasi bersifat strategis, sementara permasalahan *routing* bersifat taktis. Rute dapat dikalkulasi dan diputuskan secara berkala (bahkan terkadang harian).

Location Routing Problem meliputi topik bahasan yang luas. Menurut (Nagy and Salhi, 2007), LRP setidaknya dapat dikelompokkan dalam 4 (empat) klasifikasi:

1. Berdasarkan struktur hierarkinya.

Sebagian besar LRP terdiri dari beberapa fasilitas yang melayani sejumlah *customer*, yang terhubung dengan masing-masing depot berdasarkan *tour*. Akan tetapi, ada beberapa struktur LRP yang tidak standar, antara lain:

- (a) *transportation-location problem* tidak melibatkan perencanaan *tour*,
- (b) *many-to-many routing problem*, yang selain melibatkan perencanaan *tour* antara fasilitas-*customer*, juga melibatkan rute antara fasilitas,
- (c) *vehicle routing-allocation problem* menyertakan rencana *tour* antar fasilitas, tetapi tidak antara fasilitas dan *customer*,
- (d) *multi-level location-routing problem* mengikutsertakan rencana *tour* untuk kedua *layer*, dan bahkan mungkin lebih dari satu level fasilitas.

2. Tipe input data.

Terdapat dua macam tipe input data: *deterministic* dan *stochastic*. Sebagian besar kasus pada LRP bersifat *deterministic*. Adapun pada kasus LRP yang bersifat *stochastic*, variabel *stochastic* biasanya hanya terdapat pada *demand*.

3. Periode perencanaan.

Berdasarkan periode perencanaan, terdapat *single-period* dan *multi-period*. Permasalahan ini juga sering disebut dengan *static* dan *dynamic*. Umumnya, mayoritas penelitian lebih berfokus pada permasalahan *static* LRP.

4. Metode solusi.

Berdasarkan metode solusi, terdapat metode *exact* dan *heuristic*. Sebagian besar peneliti menggunakan metode *exact*, walaupun untuk sebagian kasus, penggunaan metode *exact* lebih sukses.

Sementara itu, berdasarkan strukturnya, LRP juga setidaknya dapat dipecah menjadi lima jenis:

1. Fungsi obyektif.

Obyektif yang paling umum digunakan dalam permasalahan *location routing* adalah minimal keseluruhan *cost*, dimana *cost* dapat dipecah menjadi *depot cost* dan *vehicle cost*. Hanya terdapat sedikit penelitian yang menggunakan fungsi obyektif selain *total overall cost* atau *multiobjective*, antara lain (Averbakh and Berman, 1994) dan (Averbakh and Berman, 1995).

2. *Solution space.*

Solution space dapat berupa diskrit, kontinyu, atau *network*. Sebagian besar literatur dalam permasalahan LRP menggunakan lokasi yang bersifat diskrit. Akan tetapi ada beberapa permasalahan *round-trip location* yang terbatas pada *path* atau *tree network*, salah satunya (Simchi-Levi, 1991).

3. Jumlah *depot*.

Berdasarkan jumlah *depot* yang digunakan, terdapat *single depot* dan *multi depot* LRP. Sebagian besar penelitian menggunakan *multi depot*, meskipun ada beberapa kasus yang terbatas hanya pada *single depot*, antara lain (Laporte and Nobert, 1981), (Averbakh and Berman, 1994), dan (Simchi-Levi, 1991).

4. Jumlah dan tipe kendaraan.

Pada sebagian besar penelitian LRP, jumlah kendaraan yang digunakan tidak tetap dan tipe kendaraan bersifat homogen. Meskipun begitu, terdapat juga penelitian yang menggunakan kendaraan yang bertipe heterogen, seperti (Bookbinder and Reece, 1988) dan (Salhi and Fraser, 1996). Selain itu, terdapat kasus khusus dimana sebuah depot hanya terdapat satu kendaraan, misalnya (Branco and Coelho, 1990).

5. Struktur rute.

Struktur rute yang biasa dipakai adalah dengan memulai dari sebuah *depot*, kemudian mengunjungi sejumlah *customer*, dan kembali lagi ke *depot* asal. Terdapat pula struktur rute yang memungkinkan adanya *multiple trip*, dan *pickup-delivery*.

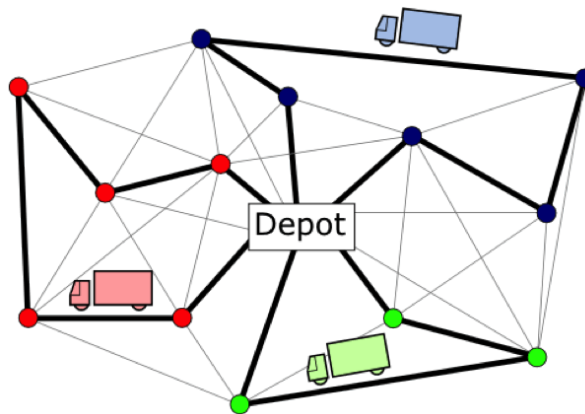
2.4.1 *Vehicle Routing Problem*

Vehicle Routing Problem (VRP) merupakan salah satu permasalahan optimasi kombinatorial yang diusulkan pertama kali oleh (Dantzig and Ramser, 1959). VRP dapat dideskripsikan sebagai permasalahan dalam menentukan desain pengiriman yang optimal atau koleksi rute dari satu atau lebih *depot* ke sejumlah kota atau pelanggan *customer* yang tersebar secara geografis (Laporte, 1992). VRP memiliki peran yang penting dalam distribusi logistik. Pada VRP, kunjungan dimulai dengan kendaraan meninggalkan *depot*, melayani sejumlah *customer*, dan kembali lagi ke depot asal, dimana masing-masing *customer* ditandai dengan *demand*. VRP klasik merupakan generalisasi dari *Traveling Salesman Problem* (TSP) dan *Bin Packaging Problem* (BPP) (Garey and Johnson, 2002).

VRP dapat didefinisikan sebagai berikut, $G = (V, A)$ adalah sebuah *graph* dimana $V = 1, \dots, n$ adalah sebuah set dari simpul(*vertex*) yang merepresentasikan kota dengan *depot* berada pada vertex 1. Sementara A merupakan sebuah set dari busur(*arc*), dimana setiap $arc(i, j)$ $i \neq j$ diasosiasikan dengan matrik jarak *non-negative* $C = (c_{ij})$. Dalam beberapa konteks, c_{ij} dapat diinterpretasikan sebagai *travel cost* atau *travel time*. Ketika C bersifat simetris, maka seringkali A diganti dengan satu set E dari *undirected edges*. Sebagai tambahan, jika diasumsikan terdapat m kendaraan yang berada pada depot, dimana $m_L < m < m_U$, maka ketika $m_L = m_U$, maka m dikatakan *fixed*, dan ketika $m_L = 1$ dan $m_U = n - 1$, maka m dikatakan *free*. Jika m tidak *fixed*, maka seringkali diasosiasikan *fixed cost* f untuk setiap kendaraan. Pada VRP klasik, asumsi bahwasannya setiap kendaraan identik dan memiliki kapasitas yang sama, yaitu D .

VRP terdiri dari satu set rute dari kendaraan yang memiliki *cost* terkecil, dengan ketentuan:

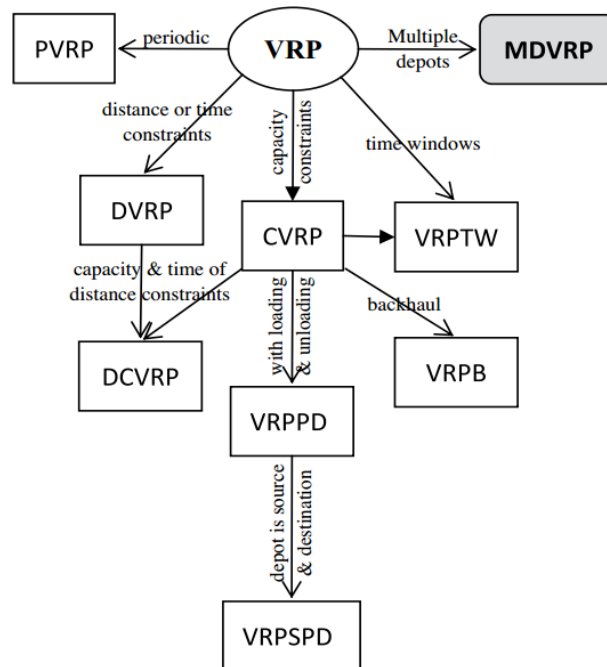
1. Setiap *customer* pada V dikunjungi hanya sekali dan oleh satu kendaraan,
2. Semua kendaraan memulai dan mengakhiri perjalanan pada satu *depot*,
3. Total *demand* dari seluruh *customer* pada setiap rute tidak melebihi Q ,
4. Durasi total dari rute tidak melebihi D



Gambar 2.3: Ilustrasi *Vehicle Routing Problem* (VRP)

Terdapat sejumlah variasi dari VRP klasik yang telah diteliti, antara lain: *Capacitated VRP (CVRP)* yang diteliti oleh (Baldacci et al., 2010), (Cordeau et al., 2007), dan (Toth and Vigo, 2002); *VRP with Time Windows (VRPTW)*, *VRP with Pickup and Delivery (VRPPD)*, dan *Periodic VRP (PVRP)* oleh (Solomon and Desrosiers, 1988); *Dynamic VRP (DVRP)* oleh (Psaraftis, 1995), dan berbagai varian lainnya, dimana keseluruhannya hanya mempertimbangkan satu buah depot.

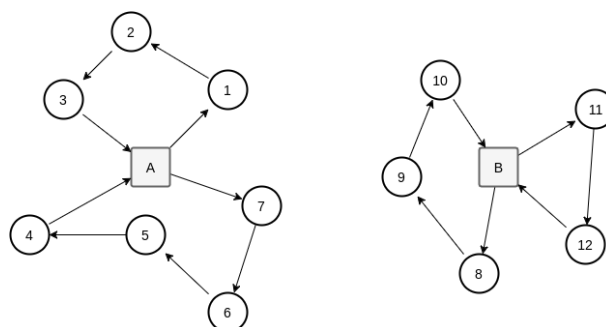
Sementara itu, *Multi-Depot Vehicle Routing Problem (MDVRP)* merupakan salah satu varian dari VRP klasik dimana terdapat lebih dari satu *depot* yang digunakan. Gambar 2.4 memberikan ilustrasi hierarki variasi dari VRP.



Gambar 2.4: Hierarki variasi dari VRP (Weise et al., 2009)

2.4.2 Multi-Depot Vehicle Routing Problem

Multi-Depot Vehicle Routing Problem (MDVRP) merupakan salah satu varian dari VRP klasik dimana terdapat lebih dari satu *depot* yang digunakan. Pada variasi ini, setiap *customer* akan dikunjungi oleh kendaraan yang berasal dari salah satu dari beberapa *depot*. Pada MDVRP standar, kendaraan harus memulai dan mengakhiri rute pada depot yang sama. Gambar 2.5 memberikan ilustrasi tentang MDVRP.



Gambar 2.5: Ilustrasi *Multi Depot VRP*

Berdasarkan (Renaud et al., 1996a), MDVRP dapat diformulasikan sebagai

berikut, $G = (V, A)$ adalah sebuah *graph*, dimana V adalah sebuah set dari simpul(*vertex* atau *node*) dan A adalah sebuah set dari busur(*arcs*). *Node* dibagi menjadi 2 (dua) *subset*: *customer* yang akan dilayani $V_C = \{1, \dots, N\}$, dan satu set depot $V_D = \{N+1, \dots, N+M\}$, dimana $V_C \cup V_D = V$ dan $V_C \cap V_D = \emptyset$. Terdapat biaya *non-negative* c_{ij} yang diasosiasikan untuk setiap *arc* $(i, j) \in A$. *Demand* untuk masing-masing *customer* adalah d_i , dan tidak ada *demand* pada *depot node*. Terdapat juga sejumlah K kendaraan yang identik, yang masing-masing memiliki kapasitas Q . *Service time* untuk masing-masing *customer* i adalah t_i , sementara waktu durasi maksimum untuk masing-masing rute adalah T . Faktor konversi w_{ij} mungkin dibutuhkan untuk mengkonversi *cost* c_{ij} menjadi unit waktu. Pada MDVRP klasik, *cost* sama dengan waktu dan unit jarak, sehingga $w_{ij} = 1$.

Dalam formula matematika, berdasarkan (Kulkarni and Bhave, 1985), variabel biner x_{ijk} sama dengan 1 ketika kendaraan k mengunjungi *node* j tepat setelah *node* i . Variabel tambahan y_i juga digunakan untuk mengeliminasi *subtour*. Persamaan 2.1 meminimalisasi *total cost*. Persamaan 2.2 dan 2.3 menjamin bahwa setiap *customer* hanya dilayani tepat satu kali oleh satu kendaraan. Alur kendaraan dijamin dengan persamaan 2.4. Kapasitas kendaraan dan batasan durasi untuk setiap rute terdapat pada persamaan 2.5 and 2.6. Persamaan 2.7 dan 2.8 mengecek keberadaan kendaraan. Eliminasi *subtour* terdapat pada persamaan 2.9. Terakhir, persamaan 2.10 dan 2.11 mendefinisikan x dan y sebagai variabel biner.

$$\sum_{i=1}^{N+M} \sum_{j=1}^{N+M} \sum_{k=1}^K c_{ij} x_{ijk}; \quad (2.1)$$

$$\sum_{i=1}^{N+M} \sum_{k=1}^K x_{ijk} = 1 (j = 1, \dots, N); \quad (2.2)$$

$$\sum_{j=1}^{N+M} \sum_{k=1}^K x_{ijk} = 1 (j = 1, \dots, N); \quad (2.3)$$

$$\sum_{i=1}^{N+M} x_{ihk} - \sum_{j=1}^{N+M} x_{hjk} = 0 \quad (2.4)$$

$$(k = 1, \dots, K; h = 1, \dots, N+M);$$

$$\sum_{i=1}^{N+M} \sum_{j=1}^{N+M} d_i x_{ijk} \leq Q(k = 1, \dots, K); \quad (2.5)$$

$$\sum_{i=1}^{N+M} \sum_{j=1}^{N+M} (c_{ij} w_{ij} + t_i) x_{ijk} \leq T(k = 1, \dots, K); \quad (2.6)$$

$$\sum_{i=N+1}^{N+M} \sum_{j=1}^N x_{ijk} \leq 1(k = 1, \dots, K); \quad (2.7)$$

$$\sum_{j=N+1}^{N+M} \sum_{i=1}^N x_{ijk} \leq 1(k = 1, \dots, K); \quad (2.8)$$

$$y_i - y_j + (M + N)x_{ijk} \leq N + M - 1; \quad (2.9)$$

for $1 \leq i \neq j \leq N$ and $1 \leq k \leq K$;

$$x_{ijk} \in \{0, 1\} \forall i, j, k; \quad (2.10)$$

$$y_i \in \{0, 1\} \forall i; \quad (2.11)$$

Terdapat beberapa algoritma yang dapat digunakan untuk menyelesaikan permasalahan MDVRP. (Laporte et al., 1984) dan (Laporte et al., 1988) mengembangkan algoritma *branch-and-bound*, tetapi, algoritma ini hanya dapat digunakan untuk sejumlah kecil *instance*. Sejumlah algoritma *heuristic* juga telah diteliti untuk MDVRP. Algoritma *heuristic* yang relatif awal, yang berbasis pada prosedur *simple construction and improvement* diteliti oleh (Tillman, 1969), (Tillman and Cain, 1972), (Tillman and Hering, 1971), (Wren and Holliday, 1972), (Gillett and Johnson, 1976), (Golden et al., 1977), dan (Raft, 1982). Penelitian yang lebih baru, (Chao et al., 1993) mengusulkan sebuah prosedur pencarian yang mengkombinasikan metode lokal *record-to-record* (Dueck, 1993) untuk mengalokasikan ulang

pelanggan ke rute kendaraan yang berbeda, yang dilanjutkan dengan prosedur 2-opt (Lin, 1965) untuk meningkatkan rute individual.

Penelitian yang lain, (Renaud et al., 1996a) menggunakan *tabu search heuristic* dimana solusi awal disusun dengan meng-assign setiap *customer* dengan depot terdekatnya. Algoritma *petal* yang dirancang oleh penulis yang sama, (Renaud et al., 1996b), kemudian digunakan untuk mencari solusi dari setiap VRP yang diasosiasikan pada masing-masing depot. Terakhir terdapat fase *improvement*, baik dengan menggunakan subset dari pertukaran 4-opt untuk meningkatkan rute individual, menukar *customer* antar rute dari depot yang sama maupun berbeda, atau menukar *customer* antara 3 (tiga) rute.

Penggunaan tabu search juga dilakukan oleh (Cordeau et al., 1997), dimana solusi awal diperoleh dengan meng-assign setiap *customer* dengan depot terdekat dan solusi digenerate untuk masing-masing depot dengan menggunakan algoritma sweep. *Improvement* dilakukan dengan memindahkan *customer* antara dua rute kepada depot yang sama, atau dengan merelokasi *customer* pada rute ke depot yang lain. *Reinsertion* dilakukan dengan menggunakan *GENI heuristic* (Gendreau et al., 1992).

Selain tabu search, terdapat juga beberapa algoritma yang dapat digunakan untuk menyelesaikan permasalahan MDVRP, yaitu: adaptive large neighborhood search (ALNS) (Pisinger and Ropke, 2007), fuzzy logic guided genetic algorithm (FLGA) (Lau et al., 2010), paralel iterated tabu search (ITS) (Cordeau and Maischberger, 2012), hybrid algorithm combining Iterated Local Search and Set Partitioning (ILS-RVND-SP) (Subramanian et al., 2013), hybrid genetic algorithm with adaptive diversity control (HGSADC+) (Vidal et al., 2014), hybrid Granular Tabu Search (ELTG) (Escobar et al., 2014), dan evolution algorithms (EAs) (Weise et al., 2009).

2.4.3 Cooperative Evolution Strategy

Evolution Strategies (ESs) merupakan salah satu *family* dari algoritma optimasi yang terinspirasi dari alam. ESs pertama kali diteliti oleh (Rechenberg, 1965) dan (Huning et al., 1976), yang kemudian diteliti lebih lanjut oleh (Schwefel, 1975). Pada ESs, setiap individu direpresentasikan oleh *genetic building blocks* dan parameter strategi yang memodelkan perilaku individu pada lingkungannya. Evolusi kemudian terjadi, yang terdiri dari perkembangan dari karakteristik genetik dan parameter strategi, dimana evolusi dari karakteristik genetik dikontrol oleh parameter strategi.

ESs merepresentasikan individu sebagai sebuah *tuple*, yang terdiri dari *decision*

vector x yang akan dioptimalisasi, dan vector parameter strategi σ yang merepresentasikan jumlah mutasi dari setiap dimensi.

$$\chi(t) = (x(t), \sigma(t)) \quad (2.12)$$

Berdasarkan observasi biologi, keturunan (*offspring*) harus mempunyai kesamaan dengan induknya,

$$\chi'(t) = (x'(t), \sigma'(t)) \quad (2.13)$$

Operator seleksi kemudian akan menentukan yang terbaik antara induk dan keturunannya. Asumsi yang digunakan adalah,

$$x(t+1) = \begin{cases} x'(t) & f(x'(t)) < f(x(t)) \\ x(t) & \text{otherwise} \end{cases} \quad (2.14)$$

dan

$$\sigma(t+1) = \begin{cases} \sigma'(t) & f(x'(t)) < f(x(t)) \\ \sigma(t) & \text{otherwise} \end{cases} \quad (2.15)$$

Algoritma 2.1 merupakan framework generik pada implementasi ES. Parameter μ and λ mengindikasikan jumlah induk dan jumlah keturunannya. Komponen utama dari algoritma ES adalah:

1. **Initialization:** Untuk setiap individu, *genotype*-nya diinisialisasi sesuai dengan konstrain dari masalah. Parameter dari strategi juga diinisialisasi.
2. **Recombination:** Keturunan diproduksi dengan melalui operator *crossover* pada dua atau lebih induk.
3. **Mutation:** Keturunan kemudian bermutasi, dimana jumlah mutasi diukur dari parameter strategi adaptasi.
4. **Evaluation:** *Absolute fitness function* digunakan untuk mengukur kualitas dari solusi yang direpresentasikan dengan *genotype* dari individu.

5. **Selection:** Operator seleksi digunakan untuk dua tujuan: memilih induk untuk rekomendasi, dan menentukan individu yang masih bertahan.

Algoritma 2.1 Algoritma Evolution Strategy

```

1: Set the generation counter,  $t = 0$ ;
2: Initialize the strategy parameters;
3: Create and initialize the population,  $C(0)$ , of  $\mu$  individuals;
4: for each individual,  $\chi_i(t) \in C(t)$  do
5:   Evaluate the fitness,  $f(\chi_i(t))$ ;
6: end for
7: while stopping condition(s) not true do
8:   for  $i = 1, \dots, \lambda$  do
9:     Choose  $\rho \geq 2$  parents at random;
10:    Create offspring through application of crossover operator on parent
        genotypes and strategy parameters;
11:    Mutate offspring strategy parameters and genotype;
12:    Evaluate the fitness of the offspring;
13:   end for
14:   Select the new population,  $C(t + 1)$ ;
15:    $t = t + 1$ ;
16: end while

```

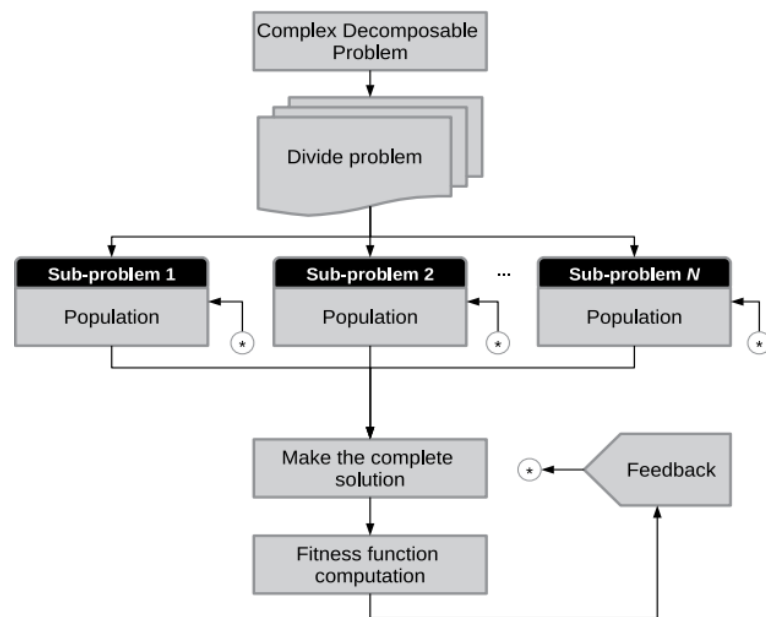
Jika pada algoritma evolusi (EA), evolusi biasanya dipandang sebagai usaha populasi untuk beradaptasi dengan lingkungan yang tetap, terdapat algoritma terkait yang lebih bersifat natural, yaitu algoritma *coevolution* (CoEA). Algoritma *coevolution* adalah algoritma yang bersifat komplemen (saling melengkapi) antara spesies terkait. Ilustrasi *coevolution* antara dua spesies adalah seperti dicontohkan oleh (Holland and others, 1990), yaitu interaksi antara tanaman dan serangga. Agar tetap dapat bertahan, tanaman membutuhkan mekanisme evolusi untuk mempertahankan diri dari serangga, sementara serangga membutuhkan tanaman sebagai sumber makanan. Keduanya, tanaman dan serangga, masing-masing berevolusi untuk mendapatkan karakteristik yang membuat mereka bertahan hidup.

Perbedaan antara algoritma CoEA dengan EA, adalah CoEA tidak hanya terkait antar populasi, tetapi juga merespon perubahan lingkungan yang disebabkan oleh populasi yang lain. Perbedaan lain, EA mendefinisikan solusi optimal melalui fungsi *fitness* yang bersifat absolut, yang mana fungsi *fitness* tersebut mendorong terjadinya evolusi. Sebaliknya, CoEA tidak mendefinisikan solusi optimal menggunakan fungsi *fitness*, sehingga evolusi terus terjadi dimana optimal solusi diperoleh dengan mengalahkan lawan.

Terdapat dua macam pendekatan berbasis *coevolution*, yaitu *competitive coevolution* dan *cooperative coevolution*. *Competitive coevolution* dibagi menjadi dua:

1) **Competition**, dimana satu atau lebih populasi saling menghambat. Keberhasilan satu populasi membuat populasi lain gagal. 2) **Amensalism**, dimana terhambatnya satu populasi tidak berpengaruh terhadap populasi lain. Sementara *cooperative co-evolution* juga dibagi menjadi tiga: 1) **Mutualism**, dimana satu atau lebih populasi saling menguntungkan. 2) **Commensalism**, dimana hanya sebagian populasi diuntungkan, sementara populasi yang lain tidak terpengaruh. 3) **Parasitism**, dimana sebagian populasi mengambil keuntungan dari populasi yang lain.

Penelitian yang menggunakan CoEA, dalam kaitannya dengan penyelesaian masalah MDVRP, salah satunya dilakukan oleh (de Oliveira et al., 2016). (de Oliveira et al., 2016) membagi masalah menjadi beberapa sub-masalah, dimana masing-masing sub-masalah akan menjadi sebuah populasi yang akan berinteraksi dengan populasi yang lain. Setiap populasi kemudian akan berevolusi, dimana sebagaimana pada teori *coevolution*, evolusi dari satu populasi akan mempengaruhi populasi yang lain. Pada penelitian (de Oliveira et al., 2016), hubungan yang terjadi pada setiap populasi bersifat *cooperative*, dimana setiap populasi akan bekerja sama membentuk solusi terbaik.



Gambar 2.6: Lifecycle pada Cooperative Coevolution (de Oliveira et al., 2016)

2.5 Messaging Solution

Ketika dua buah aplikasi ingin saling bertukar data, mereka akan melakukannya dengan mengirimkan data melalui *channel* yang akan menghubungkan keduanya. Yang menjadi tantangan adalah memilih *channel* yang tepat dalam pengiriman pe-

san.

Terdapat banyak cara interaksi antara *client* dengan *server*, dan yang paling banyak diadopsi adalah dengan melalui media web. Begitu juga dengan mekanisme yang dipakai dalam implementasi *location routing problem*, sebagian besar menggunakan media web, seperti (Weise et al., 2009), (Sengoku and Yoshihara, 1998), (Sarmenta et al., 2002), dan (Diaz, 2012). Akan tetapi, teknologi web bekerja secara *synchronous*, dimana setiap *request* dan *reply* akan diproses secara berurutan, sehingga tidak sesuai diimplementasikan pada sistem yang bersifat *information driven* (Mhl, 2002). Selain itu, komunikasi *point-to-point* dan *synchronous* membuat pengembangan menjadi tidak fleksibel (Eugster et al., 2003).

2.5.1 Mekanisme *Publish/Subscribe*

Publish/subscribe interaction merupakan salah satu alternatif untuk komunikasi antara *client* dan *server*. Paradigma interaksi pada *publish/subscribe* adalah adanya *subscriber* yang memiliki ketertarikan pada suatu *event* atau *pattern of event*, agar dapat dikirimkan notifikasi tentang sebuah *event* oleh *publisher* yang sesuai dengan *interest-nya*.

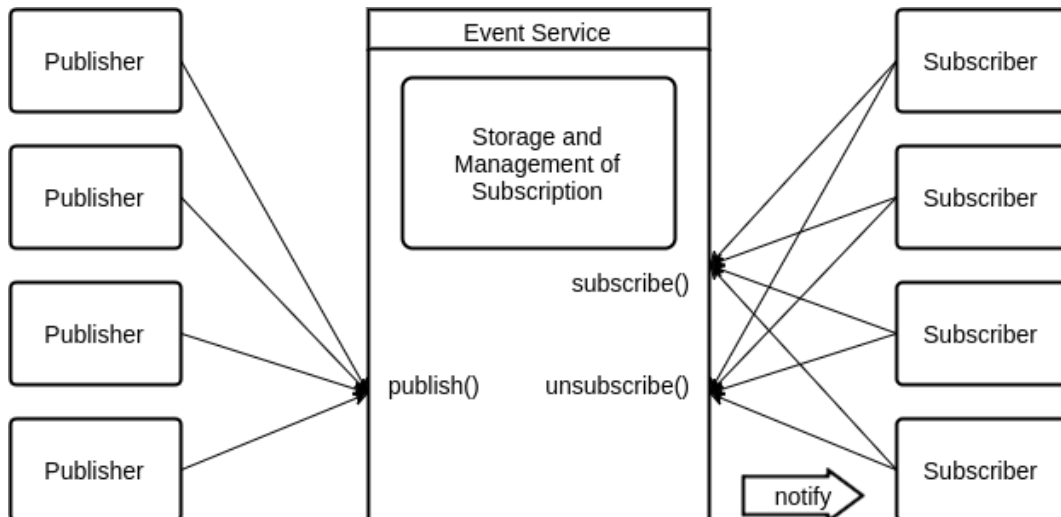
Model dasar dari sistem berbasis *publish/subscribe*, seperti Gambar 2.7, bergantung pada *event notification service* yang menyediakan penyimpanan dan manajemen dari *subscription*. *Event service* tersebut berperan sebagai mediator antara *publisher* yang berperan sebagai produsen *event* dan *subscriber* yang berperan sebagai konsumen dari *event*.

Subscriber mendaftarkan ketertarikannya pada sebuah *event* tanpa perlu mengetahui sumber dari *event* tersebut, dengan memanggil operasi *subscribe()* pada *event service*. Informasi *subscription* ini tetap tersimpan didalam penyimpanan pada *event service*. Untuk menggenerate *event*, *publisher* memanggil operasi *publish()*. *Event service* kemudian meneruskan *event* yang diproduksi tersebut kepada *subscriber* dengan ketertarikan yang sesuai.

Paradigma *publish/subscribe* memiliki sifat *loose coupling*. Pemisahan (*decoupling*) informasi yang terjadi antara *subscriber* dan *publisher* dapat dipisahkan dalam 3 (tiga) dimensi. Sebagaimana diilustrasikan pada Gambar 2.8, *decoupling* informasi tersebut adalah sebagai berikut:

1. *Space decoupling*.

Interaksi antara *publisher* dan *subscriber* tidak perlu saling mengetahui satu dengan yang lain. *Publisher* mengirimkan *event*, dan *subscriber* menerima *event* secara tidak langsung melalui *event service*. *Publisher* biasanya tidak



Gambar 2.7: Arsitektur dasar pada Pub/Sub (Eugster et al., 2003)

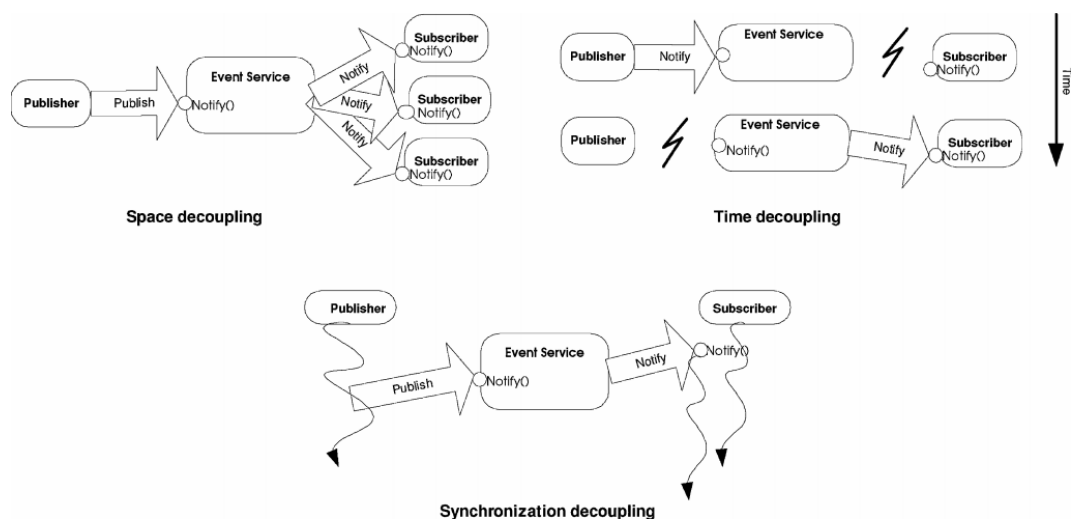
memegang referensi terhadap *subscriber*. Begitu juga sebaliknya, *subscriber* biasanya tidak memegang referensi terhadap *publisher*.

2. *Time decoupling.*

Pihak-pihak yang berinteraksi, tidak harus berinteraksi pada waktu yang bersamaan. *Publisher* dapat mengirimkan *event* pada saat *subscriber* dalam kondisi terputus. Begitu juga sebaliknya, *subscriber* tetap dapat menerima *event*, meskipun *original publisher* dalam kondisi terputus.

3. *Synchronizing decoupling.*

Publisher tidak diblok ketika memproduksi *event*, dan juga *subscriber* tetap dapat memperoleh informasi meskipun sedang mengerjakan tugas yang lain. *Publisher* dan *subscriber* tidak berada dalam *main flow*, sehingga tidak berinteraksi secara *synchronous*.

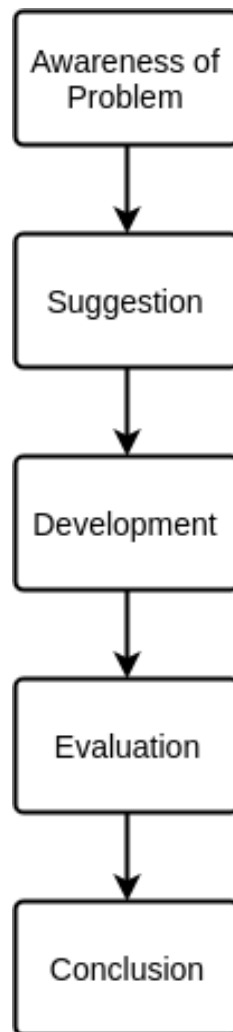


Gambar 2.8: Pemisahan informasi pada Pub/Sub (Eugster et al., 2003)

BAB 3

METODOLOGI

Metodologi penelitian yang akan dilakukan di dalam penelitian ini adalah *Design Science Research Methods and Patterns* (?). Metodologi penelitian terdiri dari 5 (lima) tahapan seperti digambarkan pada Gambar 3.1 yaitu: *Awareness of Problem*, *Suggestion*, *Development*, *Evaluation*, dan *Conclusion*.



Gambar 3.1: Tahapan *Design Science Research Methods and Patterns*

3.1 *Awareness of Problem*

Langkah pertama dalam *Design Science Research Methods and Patterns* adalah *awareness of problem*. Langkah ini merupakan proses identifikasi dan definisi masalah. Permasalahan yang diidentifikasi dapat tersusun dari permasalahan nyata (*real problem*) maupun permasalahan penelitian (*research problem*). *Real problem* diidentifikasi melalui analisis dan wawancara dengan *subject matter*. Sementara *research problem* diidentifikasi dari riset-riset yang meneliti permasalahan yang terkait dengan *real problem*.

3.2 *Suggestion*

Setelah masalah terdefinisikan, maka tahapan berikutnya adalah menemukan solusi dari masalah tersebut. Serangkaian analisis dan *preliminary research* dilakukan untuk menemukan kandidat solusi dari permasalahan. Pada tahapan ini, akan diperoleh keluaran berupa *tentative design* atau *design overview*.

3.3 *Development*

Tentative design yang diperoleh pada tahap sebelumnya kemudian dikembangkan dan diimplementasikan pada tahap ini. Elaborasi dari *tentative design* memerlukan kreatifitas. Complete design diimplementasikan dalam sejumlah bahasa pemrograman yang bervariasi, kemudian dikombinasikan dengan sejumlah software untuk membentuk sebuah prototype sistem. Sebuah mekanisme komunikasi juga dipilih pada tahap ini untuk mendukung implementasi dari sistem.

3.4 *Evaluation*

Prototype sistem yang diimplementasikan pada tahap sebelumnya kemudian diuji dengan menggunakan serangkaian skenario. Sejumlah data juga digunakan pada tahapan ini. Hasil pengujian kemudian direpresentasikan dalam tabel dan grafik untuk memudahkan dalam menarik kesimpulan.

3.5 *Conclusion*

Kesimpulan merupakan tahapan akhir dari penelitian. Kesimpulan yang diambil merupakan harus dapat menjawab masalah yang didefinisikan.

BAB 4

ANALISIS DAN PERANCANGAN

Bab ini menjelaskan tentang perancangan sistem rekomendasi lokasi pencacahan yang diusulkan. Sebelum dijelaskan tentang sistem usulan, terlebih dahulu akan dilakukan eksperimen dan analisis terhadap solusi yang telah ada.

4.1 Analisis

Pada kondisi saat ini, lokasi pencacahan sudah ditentukan sejak awal dengan menggunakan metode sampling tertentu. Pada tahap perancangan, petugas direkrut dan dialokasikan ke lokasi pencacahan terdekat. Pengalokasian seringkali dilakukan secara subyektif berdasarkan kedekatan lokasi pencacahan dengan domisili petugas pencacahan. Akibatnya, terjadi ketidakmerataan beban kerja dan variasi total waktu penyelesaian pekerjaan yang sangat tinggi antar petugas.

Untuk mengatasi masalah ini, algoritma MDVRP dipilih sebagai metode pemecahan masalah karena memiliki karakteristik yang serupa dengan permasalahan alokasi petugas, yakni:

1. Terdapat lebih dari satu *vehicle*, dimana masing-masing *vehicle* memiliki *depot* yang berbeda. Ini analog dengan permasalahan alokasi petugas, dimana ada lebih dari satu pencacah dan masing-masing pencacah memiliki titik mulai pencacahan yang berbeda.
2. Terdapat *cost* yang harus dikeluarkan untuk melakukan perjalanan dari satu customer ke customer yang lain. Ini sejalan dengan konsep dalam pencacahan dimana terdapat *cost* untuk mengunjungi satu responden ke responden lainnya.

Dalam proses pencacahan, *cost* dapat berupa waktu tempuh, jarak tempuh, atau biaya perjalanan. Dalam penelitian ini, waktu tempuh dipilih sebagai representasi dari *cost* karena waktu tempuh menggambarkan tingkat kesulitan akses untuk tiap-tiap responden. Semakin sulit akses ke suatu wilayah, semakin lama waktu tempuh yang diperlukan.

Eksperimen dilakukan untuk membuktikan fisibilitas MDVRP dalam penyelesaian permasalahan alokasi petugas, dengan melibatkan 2 (dua) komponen utama:

1. Sejumlah pencacah yang masing-masing memiliki *depot* tersendiri.
2. Sejumlah lokasi pencacahan/blok sensus yang masing-masing terdiri dari beberapa responden.

Subbab 4.1.1 hingga subbab 4.1.3 menyajikan penjelasan rinci mengenai langkah-langkah yang dilakukan dalam eksperimen.

4.1.1 Dataset

4.1.1.1 Lokasi Pencacahan

Merujuk kepada konsep MDVRP, lokasi pencacahan dapat dianalogikan sebagai pelanggan (*customer*) yang akan dikunjungi oleh petugas pencacah (*vehicle*). Pengujian dilakukan dengan menggunakan data 182 lokasi nagari/kelurahan yang bersumber dari data wilayah administratif di Kabupaten Pesisir Selatan, Provinsi Sumatera Barat. Masing-masing lokasi memiliki atribut ID dan posisi menurut garis lintang dan bujur, seperti yang terlihat pada Table 4.1.

Tabel 4.1: Lokasi Pencacahan

	Koordinat	
	Latitude	Longitude
1302011001	-2.3504	101.1434
1302011002	-2.4233	101.0285
1302011003	-2.3798	101.0427
1302011004	-2.3884	101.049
1302011005	-2.3936	101.0546
...		
1302110019	-1.2387	100.4853
1302110020	-1.1408	100.4938
1302110021	-1.0883	100.4652
1302110022	-1.0886	100.489
1302110023	-1.1523	100.4978

4.1.1.2 Petugas Pencacahan

Berdasarkan konsep MDVRP, petugas pencacahan diibaratkan sebagai kendaraan yang harus berpindah dari satu lokasi ke lokasi lain secara berurutan. Selain memi-

liki atribut ID, masing-masing pencacah juga dilengkapi dengan atribut *depot*, yaitu lokasi dimana pencacah harus memulai dan mengakhiri kunjungan. Eksperimen ini menggunakan 15 pencacah dengan lokasi *depot* yang bervariasi, seperti contoh yang tercantum pada Table 4.2.

Tabel 4.2: Pencacah

	Koordinat Depot	
	Latitude	Longitude
1302011008	-2.3905	101.1214
1302012003	-2.199	101.1188
1302020006	-2.1225	101.0687
...		
1302100002	-1.23265	100.54314
1302101005	-1.19831	100.58078
1302110003	-1.2475	100.4745

4.1.1.3 Jarak dan Waktu Tempuh

Jarak dan waktu tempuh antar lokasi pencacahan digunakan sebagai penimbang dalam penentuan rekomendasi lokasi. Penghitungan jarak dan waktu tempuh dapat dilakukan dengan cara manual (menggunakan hasil survei dan perkiraan *subject matter*) atau memanfaatkan *Google Directions API* (Google, 2016).

Secara teknis, *Google Direction API* lebih unggul dibandingkan metode manual karena telah otomatis mengikutsertakan faktor rute tercepat, kondisi geografis, moda transportasi yang digunakan, dan kemacetan lalu lintas dalam kalkulasi jarak dan waktu tempuh. Namun, *Google Direction API* menggunakan informasi yang bersumber dari kontribusi para pengguna *Google*, sehingga ada kemungkinan rute-rute yang jarang dilewati akan memiliki informasi yang minim. Sebagai konsekuensinya, terdapat resiko hasil kalkulasi yang bias untuk rute-rute tersebut.

Kode 4.1 menggambarkan contoh *requests URL* yang dikirimkan ke *Google Direction API*. *Request URL* ini dapat dieksekusi dengan menggunakan *HTTP client*, seperti *curl* dan *wget*, atau *HTTP client library*, seperti *requests* dan *urllib* di Python. Contoh *response* Google Direction API dapat dilihat pada Gambar 4.1.

Hasil kalkulasi jarak dan waktu tempuh kemudian disimpan dalam bentuk matriks seperti yang terlihat pada Tabel 4.3. Jika lokasi *depot* dari pencacah tidak

tercakup dalam lokasi pencacahan yang ada, maka lokasi *depot* tersebut harus ditambahkan ke dalam matriks jarak dan waktu tempuh.

```
https://maps.googleapis.com/maps/api/directions/json?origin=origin_lat,
origin_lon&destination=dest_lat,dest_lon&departure_time=timestamp&
traffic_model=best_guess&key=API_KEY
```

Kode 4.1: Google Direction API Request

```
{
+ geocoded_waypoints: [...],
- routes: [
  - {
    + bounds: {...},
    copyrights: "Map data ©2016 Google",
    - legs: [
      - {
        - distance: {
          text: "0.5 km",
          value: 511
        },
        - duration: {
          text: "2 mins",
          value: 130
        },
        - duration_in_traffic: {
          text: "2 mins",
          value: 129
        },
      },
    ],
  },
],
}
```

Gambar 4.1: Google Direction API Response

Tabel 4.3: Data Jarak dan Waktu Tempuh

Lokasi A	Lokasi B	Jarak (m)	Waktu Tempuh (det)
1302021001	1302021003	11119	1055
1302021001	1302021002	9373	868
1302021001	1302021005	490	38
1302021001	1302021004	22760	2044
...			
1302040015	1302012010	77889	8305
1302040014	1302100015	103893	9984
1302040014	1302012010	73561	7546
1302100015	1302012010	171636	16801

4.1.2 Algoritma dan Implementasi

Terdapat banyak *library* yang dapat digunakan untuk mengatasi permasalahan MDVRP, baik yang berbayar maupun *open source*. Dua *open source library* yang cukup terkenal untuk permasalahan MDVRP adalah Jsprit (jsprit, 2014) dan Optaplanner (OptaPlanner, 2016). Eksperimen ini menggunakan JSpirit karena lebih berfokus pada *route finding* serta lebih mudah untuk diimplementasikan.

Jsprit merupakan sebuah library berbasis java yang digunakan untuk menyelesaikan permasalahan *traveling salesman problem* (TSP) dan *vehicle routing problems* (VRP). Jsprit mencakup berbagai skenario seperti : *pickups and deliveries, back hauls, heterogeneous fleets, finite and infinite fleets, multiple depots, time windows, open routes, different start and end locations, multiple capacity dimensions, initial loads, skills*, dll. Jsprit bekerja secara terstruktur, mulai dari pendefinisian masalah, pemilihan algoritma, pencarian solusi, hingga pemilihan solusi terbaik.

4.1.2.1 Definisi Masalah

Pendefinisian masalah dengan library Jsprit dilakukan dengan mendefinisikan lokasi pencacahan, para pencacah, dan matriks jarak dan waktu tempuh yang diimplementasikan dalam Kode 4.2, Kode 4.3, dan Kode 4.4. Ketiga variabel ini kemudian di-*build* menjadi satu dengan menggunakan *syntax* yang tercantum pada Kode 4.5.

```
Service.Builder builder = Service.Builder.newInstance(line[0]);

try {
    Location loc = Location.Builder.newInstance()
        .setId(line[0])
        .setCoordinate(
            Coordinate.newInstance(Double.parseDouble(line[2]),
                Double.parseDouble(line[1]))).build();
    builder.setLocation(loc);
} catch (Exception e) {}

Service node = builder.build();
vrpBuilder.addJob(node);
```

Kode 4.2: Definisi Lokasi Pencacahan

4.1.2.2 Definisi Algoritma

Berdasarkan masalah yang telah didefinisikan sebelumnya, Jsprit kemudian akan menciptakan algoritma yang bersifat *out of the box* sesuai dengan jumlah iterasi dan thread yang didefinisikan pada Kode 4.6.

```

VehicleTypeImpl.Builder vehicleTypeBuilder =
↳ VehicleTypeImpl.Builder.newInstance("enumerator");
vehicleTypeBuilder.setCostPerDistance(0);
vehicleTypeBuilder.setCostPerTransportTime(1);
vehicleTypeBuilder.setCostPerServiceTime(1);
VehicleType vehicleType = vehicleTypeBuilder.build();

VehicleImpl.Builder builder =
↳ VehicleImpl.Builder.newInstance(line[0]);

try {
Location loc = Location.Builder.newInstance()
.setCoordinate(Coordinate.newInstance(Double.parseDouble(line[2]),
Double.parseDouble(line[1])))
.build();
builder.setStartLocation(loc);
} catch (Exception e) {}

builder.setType(vehicleType);
VehicleImpl vehicle = builder.build();
vrpBuilder.addVehicle(vehicle);

```

Kode 4.3: Definisi Pencacah dari File .csv

```

VehicleRoutingTransportCostsMatrix.Builder costMatrixBuilder =
↳ VehicleRoutingTransportCostsMatrix.Builder
.newInstance(true);

while ((line = reader.readNext()) != null) {
try {
costMatrixBuilder.addTransportDistance(line[0], line[1],
↳ Double.parseDouble(line[2]));
costMatrixBuilder.addTransportTime(line[0], line[1],
↳ Double.parseDouble(line[3]));
} catch (Exception e) {
costMatrixBuilder.addTransportDistance(line[0], line[1], 0.0);
costMatrixBuilder.addTransportTime(line[0], line[1], 0.0);
}
}

VehicleRoutingTransportCosts costMatrix =
↳ costMatrixBuilder.build();
vrpBuilder.setRoutingCost(costMatrix);

```

Kode 4.4: Definisi Penimbang Jarak dan Waktu Tempuh dari File .csv

```

VehicleRoutingProblem.Builder vrpBuilder =
↳ VehicleRoutingProblem.Builder.newInstance();
vrpBuilder.setFleetSize(VehicleRoutingProblem.FleetSize.FINITE);
VehicleRoutingProblem problem = vrpBuilder.build();

```

Kode 4.5: Build Problem

```

VehicleRoutingAlgorithm algorithm =
↪ Jsprit.Builder.newInstance(problem)
    .setProperty(Jsprit.Parameter.THREADS, 5).buildAlgorithm();
algorithm.setMaxIterations(iterations);

```

Kode 4.6: Penentuan Algoritma

4.1.2.3 Pencarian Solusi dan Penentuan Solusi Terbaik

Pencarian solusi dilakukan dengan menggunakan algoritma yang dihasilkan pada tahap sebelumnya. Jika algoritma memproduksi lebih dari satu solusi, maka Jsprit akan melakukan pemilihan solusi terbaik seperti yang terlihat pada Kode 4.7.

```

Collection<VehicleRoutingProblemSolution> solutions =
↪ algorithm.searchSolutions();
VehicleRoutingProblemSolution bestSolution =
↪ Solutions.bestOf(solutions);

```

Kode 4.7: Pencarian Solusi

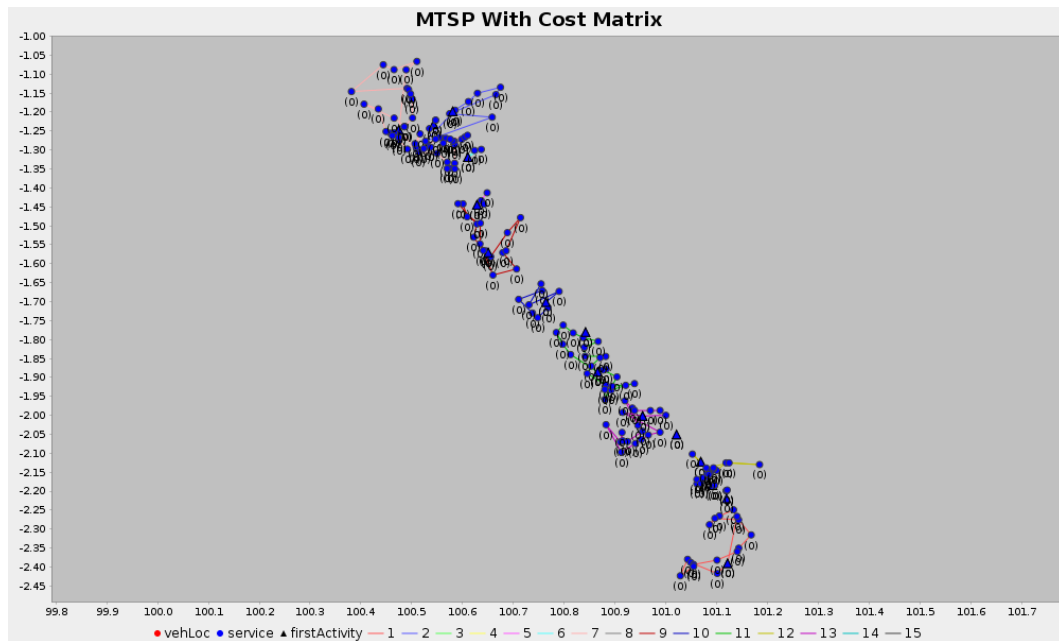
4.1.3 Hasil dan Analisis

Berdasarkan hasil dari eksperimen, diperoleh rekomendasi seperti yang diilustrasikan pada Gambar 4.2 dan Listing 4.8 untuk detail dari masing-masing pencacah. *Service time* untuk masing-masing lokasi pencacahan dirancang untuk mengikuti distribusi *normal*. Pengujian dengan menggunakan 182 lokasi pencacahan dan 15 pencacahan menghasilkan rata-rata dan standar deviasi waktu penyelesaian untuk masing-masing pencacah sebesar: **tambahkan supporting data. jangan lupa di-refer-kan**

$$\mu = 82.24 \text{ jam}$$

$$\sigma = 88.95 \text{ jam}$$

Nilai standar deviasi yang tinggi menunjukkan terjadinya ketidakmerataan beban kerja antar pencacah. Ini sekaligus menunjukkan bahwa MDVRP kurang ideal untuk digunakan dalam penentuan rekomendasi. Hal ini terjadi karena MDVRP memproduksi solusi berupa *precalculated routes* (rute yang harus dikunjungi dari awal hingga akhir tugas pencacahan) tanpa memperhitungkan faktor lamanya waktu kunjungan pada satu lokasi pencacahan (*service time*). Data mengenai *Service time* hanya akan tersedia jika suatu lokasi sudah selesai dikunjungi sehingga tidak bisa diikutsertakan dalam kalkulasi rekomendasi. Figure 4.3 menunjukkan ilustrasi



Gambar 4.2: Grafik Rekomendasi dengan MDVRP

Tabel 4.4: Total Waktu Setiap Pencacah dengan MDVRP

Pencacah	Total Waktu (jam)
1302021005	13.3
1302101005	60.8
1302070006	116
1302050007	6
1302100002	26.6
1302030005	6
1302110003	341
1302012003	19.9
1302080006	27.1
1302031005	136
1302060005	61.3
1302020006	121
1302011008	116
1302040002	162
1302090004	20.2

penerapan MDVRP pada permasalahan alokasi petugas pencacahan yang dapat dijabarkan sebagai berikut:

```

1302021005 : 1302021001 --> 1302021005
1302101005 : 1302101005 --> 1302101001 --> 1302100010 -->
↳ 1302100014 --> 1302100011 --> 1302101004 --> 1302101006 -->
↳ 1302101003 --> 1302101002
1302070006 : 1302070006 --> 1302070002 --> 1302070003 -->
↳ 1302080009 --> 1302080004 --> 1302080001 --> 1302080005 -->
↳ 1302080003 --> 1302070012 --> 1302070011 --> 1302070001 -->
↳ 1302070004 --> 1302070007 --> 1302070008 --> 1302070010 -->
↳ 1302070009 --> 1302070005
1302050007 : 1302050007
1302100002 : 1302100002 --> 1302100003 --> 1302100013 -->
↳ 1302100012
1302030005 : 1302030005
1302110003 : 1302110003 --> 1302100001 --> 1302090006 -->
↳ 1302090012 --> 1302090003 --> 1302090014 --> 1302090009 -->
↳ 1302090015 --> 1302090018 --> 1302090016 --> 1302090017 -->
↳ 1302090013 --> 1302090005 --> 1302090007 --> 1302090002 -->
↳ 1302090008 --> 1302090001 --> 1302100015 --> 1302100004 -->
↳ 1302100016 --> 1302090011 --> 1302090010 --> 1302100017 -->
↳ 1302100006 --> 1302100007 --> 1302100009 --> 1302100008 -->
↳ 1302100005 --> 1302110001 --> 1302110010 --> 1302110013 -->
↳ 1302110002 --> 1302110014 --> 1302110015 --> 1302110011 -->
↳ 1302110016 --> 1302110017 --> 1302110004 --> 1302110018 -->
↳ 1302110019 --> 1302110005 --> 1302110012 --> 1302110023 -->
↳ 1302110020 --> 1302110006 --> 1302110007 --> 1302110008 -->
↳ 1302110021 --> 1302110009 --> 1302110022
1302012003 : 1302012007 --> 1302012003 --> 1302012008
1302080006 : 1302080006 --> 1302080008 --> 1302080002 -->
↳ 1302080007
1302031005 : 1302031005 --> 1302031008 --> 1302030014 -->
↳ 1302030006 --> 1302030009 --> 1302030004 --> 1302030010 -->
↳ 1302031003 --> 1302030002 --> 1302031004 --> 1302030003 -->
↳ 1302030012 --> 1302031001 --> 1302031006 --> 1302040011 -->
↳ 1302031009 --> 1302031010 --> 1302031002 --> 1302031007 -->
↳ 1302030001
1302060005 : 1302060005 --> 1302060009 --> 1302060006 -->
↳ 1302060008 --> 1302060002 --> 1302060007 --> 1302060001 -->
↳ 1302060003 --> 1302060004
1302020006 : 1302020006 --> 1302020015 --> 1302020011 -->
↳ 1302020009 --> 1302020010 --> 1302020001 --> 1302020003 -->
↳ 1302020005 --> 1302021006 --> 1302021002 --> 1302021007 -->
↳ 1302021008 --> 1302021003 --> 1302021009 --> 1302021004 -->
↳ 1302021010 --> 1302020016 --> 1302020017
1302011008 : 1302011008 --> 1302011007 --> 1302011004 -->
↳ 1302011003 --> 1302011002 --> 1302011006 --> 1302011005 -->
↳ 1302011009 --> 1302011010 --> 1302011001 --> 1302012005 -->
↳ 1302012002 --> 1302012009 --> 1302012010 --> 1302012004 -->
↳ 1302012001 --> 1302012006
1302040002 : 1302040002 --> 1302040003 --> 1302040004 -->
↳ 1302040015 --> 1302040008 --> 1302040009 --> 1302040010 -->
↳ 1302040014 --> 1302040012 --> 1302040013 --> 1302040001 -->
↳ 1302040016 --> 1302040007 --> 1302040006 --> 1302040005 -->
↳ 1302050001 --> 1302050004 --> 1302050006 --> 1302050002 -->
↳ 1302050008 --> 1302050010 --> 1302050009 --> 1302050005 -->
↳ 1302050003
1302090004 : 1302090004 --> 1302090019 --> 1302090020

```

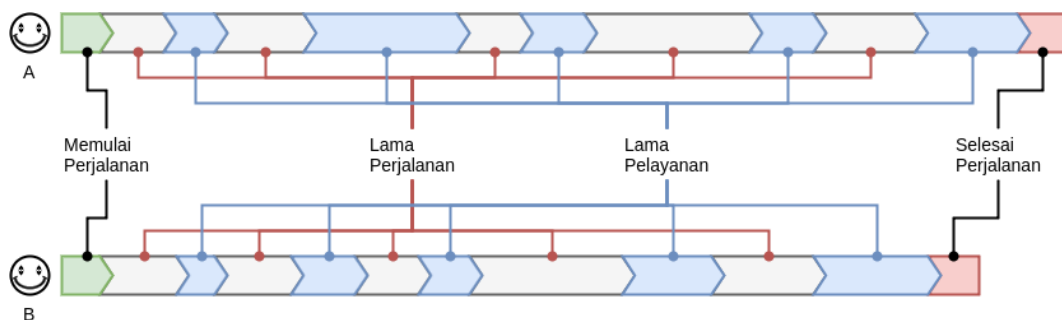
Kode 4.8: Rekomendasi dengan MDVRP

1. Terdapat 10 lokasi pencacahan yang harus dikunjungi. MDVRP *generate* solusi berupa *precalculated routes* untuk petugas A dan B, dimana

masing-masing mendapatkan jumlah beban tugas yang sama besar, yakni 5 lokasi.

2. Petugas pencacahan A dan B sama-sama memulai tugas pada waktu t_s (*time start*) yang digambarkan sebagai segmen berwarna hijau.
3. Petugas A dan B berjalan menuju lokasi pencacahan masing-masing (lokasi i) yang memakan waktu selama t_i (segmen berwarna putih).
4. Pada setiap lokasi i , masing-masing petugas memerlukan waktu sebesar $t_{i,1}$ untuk menyelesaikan tugas pencacahan (segmen berwarna biru).
5. Petugas A dan B menyelesaikan tugas pada waktu t_f (*time finish*) yang digambarkan sebagai segmen berwarna merah muda.

Dari Figure 4.3, terlihat bahwa kalkulasi rekomendasi rute tanpa melibatkan *service time*, mengakibatkan t_i dan $t_{i,1}$ yang tidak berimbang antara petugas A dan B. Petugas A mendapatkan rute dengan rata-rata *service time* ($t_{i,1}$) yang lebih panjang sehingga t_f petugas A menjadi lebih besar dibandingkan t_f petugas B. Perbedaan yang paling signifikan terlihat pada lokasi pencacahan kedua dari masing-masing petugas. Petugas A memiliki $t_{2,1}$ yang hampir 3 (tiga) kali $t_{2,1}$ petugas B. Sebagai dampaknya, terjadi ketimpangan dalam total waktu penyelesaian pencacahan antar petugas, dimana beban tugas petugas A menjadi jauh lebih berat dibandingkan petugas B.



Gambar 4.3: Penerapan MDVRP pada Permasalahan Alokasi Petugas Pencacahan

Untuk mengatasi masalah ini diperlukan suatu mekanisme kalkulasi rekomendasi yang tidak terikat *service time*. Mekanisme ini kemudian akan dikombinasikan dengan MDVRP sehingga menghasilkan suatu metode baru yang dapat menutupi kelemahan MDVRP sekaligus memberikan rekomendasi rute yang lebih adil kepada semua petugas pencacahan.

4.2 Perancangan Solusi

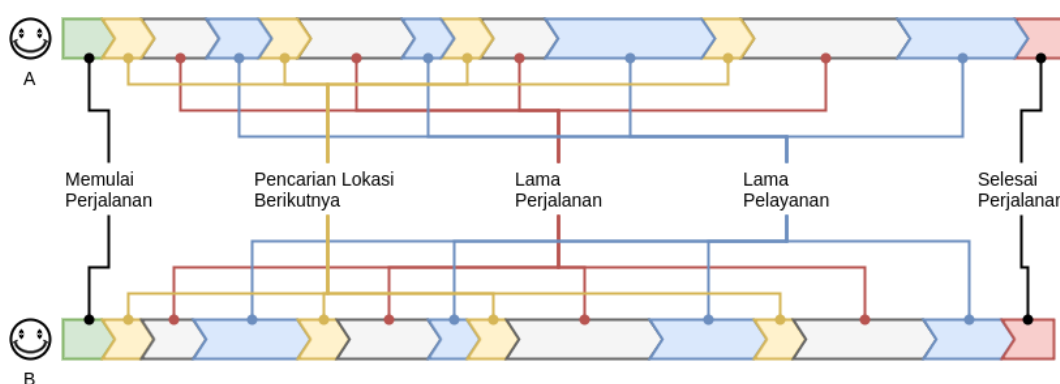
Hasil eksperimen pada subsection 4.1.3 menunjukkan bahwa mekanisme kalkulasi yang *independent* dari *service time* diperlukan untuk menghilangkan bias dan kesenjangan rute untuk tiap-tiap petugas. Solusi yang diusulkan untuk mengatasi masalah ini adalah dengan melakukan penghitungan rute secara bertahap dengan melakukan ‘hibridisasi’ antara algoritma MDVRP dengan suatu mekanisme *real time*. Idennya adalah, daripada menghitung rute secara lengkap diawal pencacahan, lebih baik rute dihitung secara bertahap dan *real time* dengan memanfaatkan *current location* dari petugas sebagai depot yang baru pada setiap tahapnya. Pada saat awal pencacahan, petugas hanya mengetahui lokasi pertama yang akan mereka kunjungi. Setelah menyelesaikan tugas pencacahan pada lokasi tersebut, petugas harus mengirimkan *request* ke *server* guna mengetahui lokasi pencacahan berikutnya.

Figure 4.4 menggambarkan penerapan *real time* MDVRP untuk mengatasi permasalahan alokasi petugas pencacahan yang dijabarkan sebagai berikut:

1. Terdapat 10 lokasi pencacahan yang harus dikunjungi. Petugas pencacahan A dan B sama-sama memulai tugas pada waktu t_s (segmen berwarna hijau).
2. Pada waktu t_1 (segmen berwarna kuning), A dan B mengirimkan *request* ke *server* untuk mendapatkan lokasi pertama yang harus mereka kunjungi.
3. A dan B berjalan menuju lokasi pencacahan masing-masing (lokasi i) yang memerlukan waktu tempuh sebesar $t_{i,1}$ (segmen berwarna putih).
4. A dan B menyelesaikan tugas pencacahan di lokasi pertama yang menghabiskan waktu sebesar $t_{i,2}$ (segmen berwarna biru).
5. Setelah menyelesaikan tugas di lokasi pertama, A dan B mengirimkan *request* ke server untuk mengetahui lokasi berikutnya yang harus dikunjungi. Server kemudian melakukan rekalkulasi rute dengan melakukan pengecekan terhadap 8 lokasi yang belum dikunjungi dan mencocokkannya dengan *current location* dari petugas. Lokasi pencacahan berikutnya dari *current location* akan dikirimkan sebagai *reply* terhadap *request* dari petugas.
6. A dan B melanjutkan tugas pencacahan ke lokasi berikutnya (lokasi i), dimana masing-masing lokasi memerlukan waktu tempuh sebesar $t_{i,1}$ dan waktu penyelesaian pencacahan (*service time*) sebesar $t_{i,2}$.
7. Petugas A dan B menyelesaikan tugas pada waktu t_f (segmen berwarna merah muda) yang hampir sama. Dari Figure 4.4 terlihat bahwa jumlah lokasi yang

dikunjungi oleh petugas A lebih sedikit dibandingkan petugas B. Hal ini dikarenakan petugas A memerlukan waktu yang lebih lama untuk melakukan perjalanan ke lokasi serta menyelesaikan tugas pencacahan di lokasi tersebut.

Mayoritas *real time system* diimplementasikan dengan menggunakan *web service*. Namun, sistem kerjanya yang *synchronous* (*request* dan *reply* harus diproses secara berurutan) menyebabkan *web service* tidak cocok digunakan untuk aplikasi yang bersifat *information driven* (Mhl, 2002). Sebagai alternatif, mekanisme *publish/subscribe* dipilih karena memungkinkan terjadinya komunikasi yang *asynchronous* antara *server/publisher* dan *client/subscriber*, dimana komunikasi tetap dapat berjalan walaupun salah satu pihak sedang dalam kondisi *offline*.

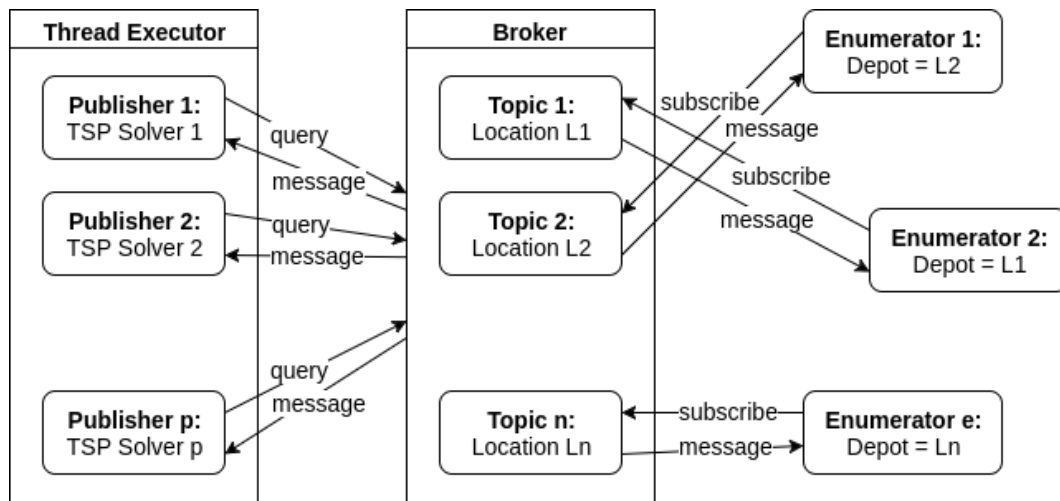


Gambar 4.4: Penerapan *real time* MDVRP pada Permasalahan Alokasi Petugas Pencacahan

4.2.1 Garis Besar Sistem Usulan

Sistem usulan untuk rekomendasi lokasi pencacahan dirancang berdasarkan arsitektur *publish/subscribe* yang terdiri dari 3 (tiga) komponen utama: *publisher* rekomendasi, petugas pencacahan yang berperan sebagai *subscriber*, dan *message broker* yang berperan sebagai penerus pesan (*message router*). Figure 4.5 memberikan ilustrasi komponen yang menyusun sistem rekomendasi lokasi pencacahan usulan.

Komunikasi antara *publisher* dan *subscriber* terjadi atas dasar kesamaan topik/*event*. Topik/*event* didapatkan dari *context* masing-masing pencacah. *Context* dari pencacah dapat berupa jenis kelamin, tingkat pendidikan, umur, pengalaman, atau domisili/lokasi pencacah. Pada penelitian ini, *current location* dari pencacah dipilih sebagai *topik* karena bersifat *unique*, dimana setiap lokasi pencacahan hanya akan dikunjungi oleh satu orang pencacah saja dan tidak akan ada lebih dari satu pencacah pada *current location* yang sama.



Gambar 4.5: Garis Besar Sistem Usulan

4.2.2 Recommendation Publisher

Berdasarkan konsep dasar mekanisme *publish/subscribe*, *publisher* akan mem-*publish* informasi ke seluruh *subscriber* tanpa melihat apakah *subscriber* tersebut men-*subscribe* topik dari informasi yang bersangkutan atau tidak. Akibatnya komunikasi yang berlangsung menjadi tidak efisien. Untuk mengatasi inefisiensi komunikasi ini, perlu dilakukan beberapa penyesuaian agar *publisher* hanya akan mem-*publish* informasi kepada *subscriber* yang bersesuaian.

Seorang *subscriber* (petugas pencacahan) me-*request* 'lokasi pencacahan yang harus dituju' kepada *publisher*. *Request* ini tidak langsung diterima oleh *publisher*, melainkan ditampung oleh *message broker*. *Publisher* akan mengecek *message broker* secara berkala untuk mendeteksi ada atau tidaknya *request* baru dengan menggunakan thread *TopicWatcher* yang dideskripsikan pada Algoritma 4.1 serta diilustrasikan dengan *flowchart* pada Figure 4.6.

Untuk setiap *request* yang diterima, *publisher* akan menyiapkan sebuah *thread* baru dengan menggunakan *current location* dari *subscriber*/petugas sebagai ID. *Thread* ini dilengkapi dengan sebuah *VRP solver procedure* (Algoritma 4.2) yang akan melakukan pencarian solusi/rute. Seluruh *thread* dari masing-masing topik ditampung di dalam sebuah *threadpool* yang menangani *thread* berdasarkan urutan waktu kedatangan. Pada akhir proses pencacahan, dimana seluruh lokasi pencacahan sudah pernah di-*request* dan di-*assign* kepada petugas, ukuran *threadpool* ini akan sama dengan jumlah seluruh topik yang tersedia.

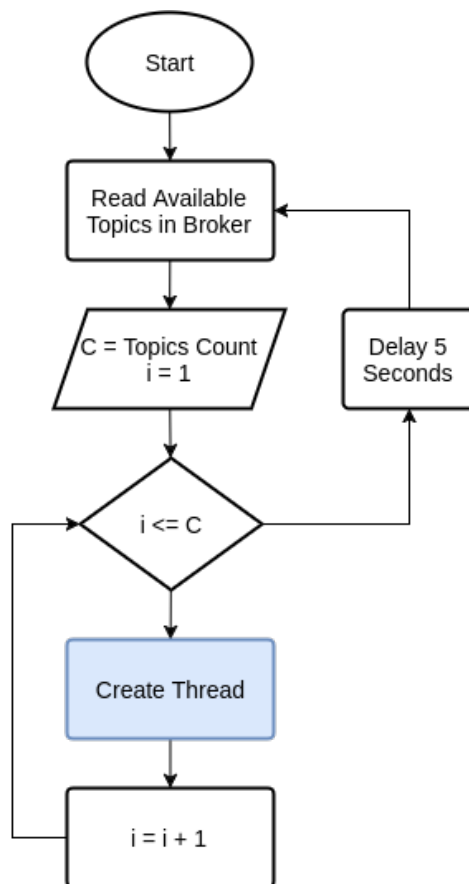
Thread-thead di dalam *threadpool* dieksekusi satu per satu sesuai dengan waktu kedatangan. Setiap eksekusi akan menjalankan *VRPSolver procedure* yang melibatkan M *subscribers* (petugas pencacahan) dan *unassigned* N lokasi pencacahan.

Algoritma 4.1 TopicWatcher

Input: *None***Output:** *None* TP = Threadpool N = Number of locations M = Number of enumerators

```

1: while true do
2:    $C \leftarrow \text{readAvailableTopicFromBroker}()$  // channel
3:   for  $m = 1$  to  $\text{len}(C)$  do
4:     for  $n = 1$  to  $N$  do
5:       if ( $C_m == L_n$ ) then
6:          $T_n = \text{Thread}(C_m, E_1 \dots E_M, (\text{unassigned})L_1 \dots L_N)$  // E = enumerator
7:          $\text{submitThreadToThreadpool}(T_n, TP)$ 
8:       end if
9:     end for
10:  end for
11: end while
  
```

**Gambar 4.6:** Flowchart Topic Watcher

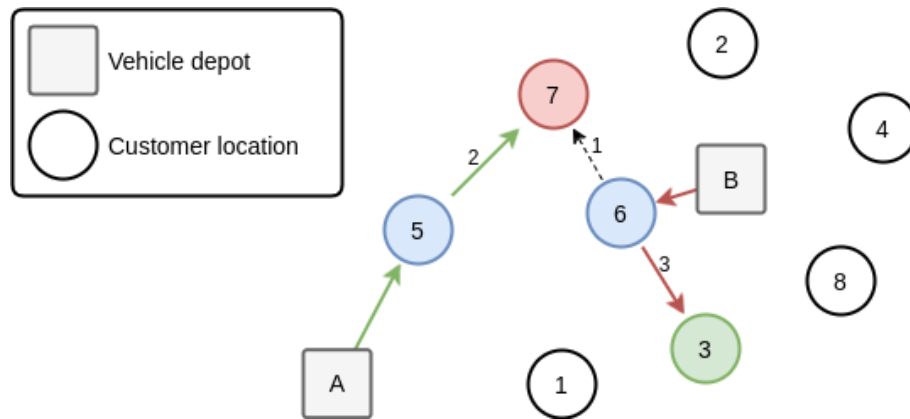
VRPSolver procedure harus mengikutsertakan seluruh *subscriber* untuk memastikan diperolehnya solusi terbaik yang bersifat global (*global best solution*). Fig-

ure 4.7 memberikan ilustrasi algoritma *Global Best Solution* dengan penjelasan sebagai berikut:

1. Petugas A dan B masing-masing melakukan pencacahan pada lokasi 5 dan 6. *Current location* mereka akan di-update secara bersesuaian.
2. Petugas B menyelesaikan pekerjaannya lebih cepat dari petugas A dan segera me-request 'lokasi pencacahan berikutnya'. Sistem kemudian menghitung 'lokasi berikutnya' berdasarkan *current location* dari petugas B.
3. Jika kalkulasi hanya melibatkan 1 petugas yang bersangkutan saja, maka *VRPSolver procedure* akan merekomendasikan lokasi terdekat dari *current location* petugas B, yakni lokasi 7. Namun, solusi yang dibuat berdasarkan sudut pandang lokal (*local best solution*) seperti ini, dapat 'merugikan' petugas lain yang tidak ikut dalam penghitungan.
4. Sebagai gambaran, misalkan saat A selesai melakukan pencacahan dan me-request 'lokasi berikutnya', lokasi 7 sudah tidak *available* lagi bagi A. Rekomendasi terbaik yang bisa diberikan oleh sistem adalah lokasi 1 yang memiliki jarak yang cukup jauh dari A.
5. Rekomendasi yang lebih tepat bisa diperoleh jika *VRPSolver procedure* mengikutsertakan seluruh petugas dalam penghitungan rute, analisis terhadap *current location* dari seluruh petugas, menghasilkan *global best solution* dimana petugas B akan mendapatkan lokasi 3 dan petugas A akan mendapatkan lokasi 7.

Proses pencarian solusi akan berakhir ketika sudah tidak ada lagi lokasi pencacahan yang berstatus *unassigned*. *VRP Solver Procedure* dijelaskan secara lebih rinci pada subsection 4.2.3.

Jumlah rute yang dihasilkan oleh *VRPSolver* berkisar antara 1 (satu) hingga M rute, dimana M merujuk pada jumlah petugas pencacahan yang mengirimkan *request*. *request* subscript i merujuk pada apa? petugas pencacah kan? diganti m aja gmn? Terus didalam algoritma ada simbol T . apa itu? modifikasi simbol huruf menyesuaikan dg isi paragraf. Setiap rute R_m ($1 \leq m \leq M$) yang dihasilkan oleh *VRP-Solver* akan di-publish kepada petugas m yang merupakan *subscriber* topik (lokasi) C_n ($1 \leq n \leq N$). *request* berarti 1 rute = 1 topik = 1 petugas, benar g??? jumlah rute = jumlah petugas. Setelah solusi diteruskan kepada *subscriber*, *message broker* akan memberikan informasi tentang jumlah *subscriber* yang menerima *request* atau memesan???? pesan kepada *publisher*, tetapi identitas dari penerima tetap



Gambar 4.7: Algoritma Global Best Solution

Algoritma 4.2 VRPSolver Procedure**Input:** T **Output:** *None*

```

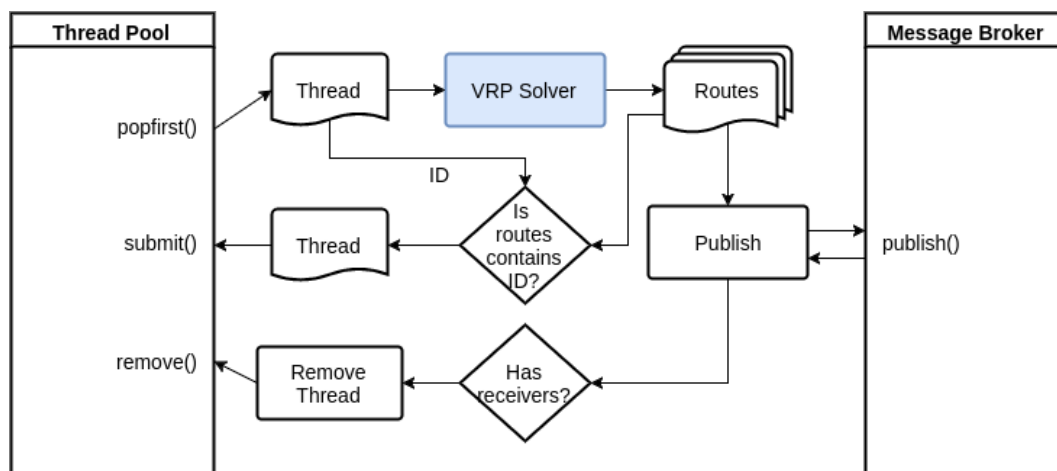
1: while true do
2:    $T = \text{popFirstThreadOrWaitNewThreadFromThreadpool}()$ 
3:    $R = \text{VRPSolver}(T)$ 
4:   for  $j = 1$  to  $\text{len}(R)$  do
5:      $r = \text{publish}(C_{R_j}, R_j)$ 
6:     if  $(r > 0)$  then
7:        $\text{cancelSolver}(T_j)$ 
8:     else if  $(C_{T_i} \notin C_{R_j})$  then
9:        $T_i = \text{Thread}(C_{T_i}, V_m, (\text{unassigned})E_1 \dots E_N)$ 
10:    end if
11:  end for
12: end while

```

tidak diketahui. $-i$ lho kok message broker baru menghubungi publisher setelah solusi disampaikan ke subscriber sih? kan harusnya message broker menghubungi publisher dulu, baru solusinya bisa diterusin ke subscriber???. Jika rute R_i memiliki satu atau lebih penerima pesan, maka *thread* yang memiliki ID $C_n -i$ atau C_{n_i} ??? akan dianulir $-i$ knp kalo cm punya 1 subscriber juga ikut dibatalkan? jelasin lagi. aku lupa. Pembatalan ini dilakukan agar solusi/rute yang telah diterima oleh *subscriber* tidak dikalkulasi ulang. $-i$ prevsent: kalimat ini rasanya masih kurang. perlu sedikit lebih diperjelas lagi

Pada praktiknya, dapat terjadi suatu kondisi dimana di antara rute-rute yang dihasilkan oleh *VRPSolver*, tidak terdapat rute untuk topik/lokasi $C_n -i$ atau C_{n_i} ??? terus siapa yg sebenarnya punya ID? thread atau *VRPSolver* ??? . Pada kondisi demikian, topik $C_n -i$ atau C_{n_i} ??? akan diantrikan kembali dengan masih melibatkan $(\text{unassigned})N$ lokasi namun hanya 1 subscriber m yang bersangkutan saja.

Ini dilakukan untuk memberikan jaminan bahwa akan selalu ada solusi/rute untuk setiap topik $C_n - i$ atau C_{n_i} ??? .



Gambar 4.8: VRP Worker

Seperti yang telah dijelaskan pada subsection xxx ??? , karakteristik *loose coupling* memberikan fleksibilitas **fleksibilitas dalam artian???** kepada mekanisme *publish/subscribe*.

Akan tetapi, keunggulan ini juga menjadi salah satu kelemahan dari mekanisme ini. Ketidaktahuan *publisher* terhadap identitas *subscriber* menyebabkan *current location* dari setiap *subscriber* tidak dapat diketahui. Untuk itu, diperlukan sebuah mekanisme yang dapat digunakan untuk pertukaran informasi antara *publisher* dan *subscriber*. Pada penelitian ini akan digunakan *shared-memory* yang akan digunakan untuk menyimpan informasi *current location* dari seluruh *subscriber*, dan *publisher* dapat mengaksesnya untuk mengetahui *current location* dari setiap *subscriber*.

Arsitektur *publish/subscribe* merupakan sebuah arsitektur yang bersifat *loose coupling*, dimana antara *publisher* dan *subscriber* tidak saling mengetahui (Mhl, 2002). Salah satu konsekuensinya, ketika seorang pencacah melakukan *subscribe*, pada saat solusi diperoleh pencacah tersebut sudah terputus dari *broker*. Untuk itu diperlukan mekanisme *caching*, sehingga pada saat pencacah terkoneksi kembali dengan *broker*, maka solusi yang terdapat pada *cache* dapat langsung *dipublish*.

4.2.3 VRP Solver

VRP solver merupakan sebuah modul yang digunakan dalam pencarian solusi/rute yang akan dikunjungi oleh setiap kendaraan. Vrp solver dipanggil dari setiap *thread* dalam *thread pool* (Algoritma 4.2). Terdapat beberapa algoritma yang dapat digunakan untuk menyelesaikan permasalahan MDVRP, seperti: tabu search

Cordeau et al. (1997), adaptive large neighborhood search (Pisinger and Ropke, 2007), fuzzy logic guided genetic algorithm (Lau et al., 2010), paralel iterated tabu search (Cordeau and Maischberger, 2012), hybrid algorithm combining iterated local search and set partitioning (Subramanian et al., 2013), hybrid genetic algorithm with adaptive diversity control (Vidal et al., 2014), hybrid granular tabu search (Escobar et al., 2014), dan cooperative coevolution algorithms (CoEAs) (de Oliveira et al., 2016). Pada penelitian ini akan digunakan *coevolutionary algorithms* (CoEAs), karena algoritma CoEAs menghasilkan nilai *mean solution values* yang kompetitif dengan waktu pemrosesan yang relatif rendah dibandingkan algoritma yang lain.

Langkah-langkah yang digunakan pada implementasi algoritma CoEAs pada VRP solver adalah sebagai berikut:

1. Definisi masalah

Masalah yang didefinisikan pada algoritma CoEAs sama seperti permasalahan MDVRP umumnya. Data yang diikutsertakan meliputi kendaraan sejumlah M beserta masing-masing kapasitasnya D_i dan konsumen sejumlah N beserta masing-masing *demand* d_i . Setiap lokasi $i, i \in (M \cup N)$ dilengkapi dengan koordinat (x_i, y_i) . Biaya untuk setiap kombinasi lokasi c_{ij} juga diikutsertakan dalam permasalahan yang didefinisikan, dimana $i, j \in (M \cup N)$. Terdapat dua macam cara yang dapat digunakan dalam menghitung biaya, yaitu: *euclidean*, dan *Google location API*. Pada metode *euclidean*, biaya dapat diasumsikan sebagai jarak antar lokasi, sementara pada metode *Google location API* dapat digunakan waktu tempuh sebagai biaya.

2. Dekomposisi masalah

Pada CoEAs, terdapat beberapa spesies yang akan saling bekerja sama untuk membentuk solusi/rute terbaik. Pada implementasinya, masalah yang didefinisikan diatas kemudian didekomposisi menjadi beberapa sub-masalah, dimana masing-masing sub-masalah akan merepresentasikan sebuah spesies.

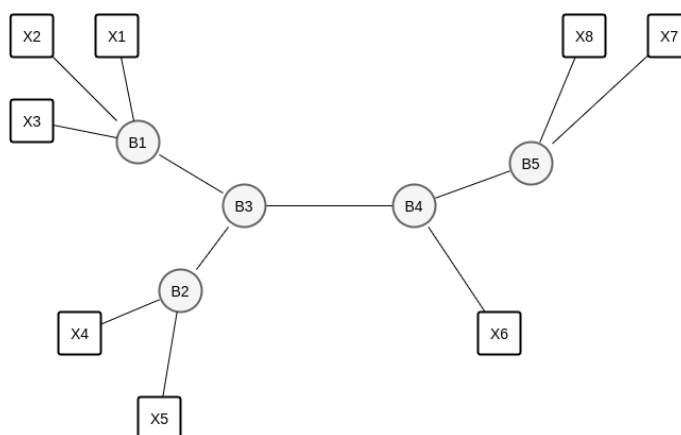
3. Evolusi

Setelah terbentuk beberapa spesies, masing-masing individu dalam spesies tersebut akan berevolusi. Evolusi terjadi secara iteratif dalam beberapa generasi. Setiap kali generasi baru terbentuk, maka generasi tersebut akan dievaluasi dengan membandingkannya dengan generasi terbaik yang pernah ada sebelumnya. Jika generasi baru lebih baik, maka generasi tersebut diterima dan generasi terbaik akan diperbarui. Jumlah iterasi yang diperlukan dalam kasus ini akan dibatasi dengan batasan waktu. Iterasi akan berhenti setelah 60 detik,

atau 40 detik tanpa perubahan generasi terbaik. Setelah iterasi terhenti, maka generasi terbaik akan menjadi solusi yang digunakan.

4.2.4 Message Broker

Message broker adalah sebuah komponen yang bertanggung jawab dalam menyalurkan (*routing*) pesan dari *publisher* ke *subscriber* sesuai dengan topik yang *unsubscribe* (Banavar et al., 1999). Suatu sistem *publish/subscribe* dapat memiliki *single broker* maupun *multi broker*. Pada arsitektur *single broker*, seluruh *subscriber* dan *publisher* terkoneksi pada satu *broker*, sementara pada *multi broker*, setiap *subscriber* maupun *publisher* dapat terkoneksi pada broker terdekat. Arsitektur *multi depot* ini juga disebut dengan *distributed pub/sub system* (Mhl, 2002), seperti ilustrasi pada Gambar 4.9. Adapun rancangan yang diusulkan akan menerapkan arsitektur terdistribusi, dikarenakan lokasi pencacahan secara geografis tersebar.



Gambar 4.9: Arsitektur *Publish-Subscribe* terdistribusi

BAB 5

IMPLEMENTASI DAN PENGUJIAN

5.1 Implementasi

Algoritma yang disusun pada Bab 4.2 diimplementasikan dalam beberapa bahasa pemrograman. Hasil implementasi kemudian dikombinasikan dengan aplikasi dan *library* pihak ketiga untuk membentuk sebuah *prototype*. Pada bagian ini akan dijabarkan tentang implementasi sistem secara lebih terperinci.

5.1.1 VRP Solver

VRP Solver yang telah dirancang pada subsection 4.2.3 diimplementasikan dalam bahasa pemrograman C++. Bahasa pemrograman C++ dipilih pada implementasi VRP Solver karena permasalahan kombinatorial pada VRP membutuhkan *resources* (*processor* dan *memory*) yang intensif, sehingga penggunaan bahasa pemrograman C++ dianggap lebih efisien dari bahasa pemrograman lainnya. Source code yang merupakan implementasi dari algoritma CoEAs yang digunakan dalam VRP Solver dapat diunduh pada tautan berikut <https://github.com/soedomoto/coes-mdvrp/tree/jni-coes-mdvrp>. Adapun lingkungan yang digunakan dalam implementasi dan kompilasi VRP Solver adalah sebagai berikut:

- Sistem Operasi : Elementary OS Loki (Berbasis Ubuntu 16.04)
- C++ Compiler : c++ (Ubuntu 5.4.0-6ubuntu1 16.04.4) 5.4.0 20160609
- Hardware : Asus TP300L, Quad-Core Intel Core i3-4030U CPU @ 1.90GHz, 3,7 GiB DDRIII, 256GB SSD

5.1.2 Publisher

Algoritma *publisher* rekomendasi lokasi pencacahan, yang meliputi Algoritma 4.1 dan Algoritma 4.2, diimplementasikan dalam bahasa pemrograman Python. Bahasa pemrograman Python dipilih karena keringkasn kode dan bersifat *interpreter*, sehingga lebih cepat ketika digunakan dalam penyusunan prototype. Source code dari implementasi Publisher dapat diunduh di <https://github.com/soedomoto/coes-mdvrp/tree/py-mdvrp-producer-redis>. Adapun lingkungan yang digunakan dalam implementasi *publisher* rekomendasi lokasi adalah sebagai berikut:

- Sistem Operasi : Elementary OS Loki (Berbasis Ubuntu 16.04)
- Python version : Python 2.7.12
- Hardware : Asus TP300L, Quad-Core Intel Core i3-4030U CPU @ 1.90GHz, 3,7 GiB DDRIII, 256GB SSD

5.2 Pengujian

Setelah seluruh algoritma diimplementasikan maka selanjutnya akan dilakukan pengujian terhadap sistem usulan. Pengujian dilakukan untuk mengetahui tingkat ketercapaian dari tujuan penelitian ini, yaitu bagaimana merancang algoritma *publisher* yang dapat memberikan rekomendasi terbaik secara global, dan bagaimana menyusun mekanisme *conflict resolution*, agar sistem tidak merekomendasikan lokasi yang sama pada dua atau lebih pencacah. Untuk mengukur keakuratan algoritma, maka hasil pengujian dari sistem usulan akan dibandingkan dengan hasil pengujian dari algoritma MDVRP berbasis CoEAs tanpa mekanisme publish/subscribe.

5.2.1 Lingkungan Pengujian

Lingkungan yang digunakan dalam pengujian sistem rekomendasi lokasi adalah sebagai berikut:

- Sistem Operasi : Elementary OS Loki (Berbasis Ubuntu 16.04)
- Redis Environment : Redis 3.2.6, Debian Jessie (Docker version)
- Python version : Python 2.7.12
- Hardware : Asus TP300L, Quad-Core Intel Core i3-4030U CPU @ 1.90GHz, 3,7 GiB DDRIII, 256GB SSD

5.2.2 Dataset dan Metric

5.2.2.1 Dataset

Untuk memastikan sistem dapat bekerja dengan baik, maka sistem perlu diujicobakan dengan menggunakan berbagai variasi data. Dataset yang paling banyak digunakan dalam pengujian kasus terkait *Vehicle Routing Problem* adalah data Breedam, Cordeau, Solomon, Homberger, dan Russell. Dalam pengujian kali ini

akan digunakan dataset Cordeau tipe 2 (Multi-Depot VRP). Dataset Cordeau sendiri tersedia dalam 7 (tujuh) tipe VRP *problem*, antara lain:

1. Tipe 0 untuk kasus VRP
2. Tipe 1 untuk kasus Periodic VRP
3. Tipe 2 untuk kasus Multi-Depot VRP
4. Tipe 3 untuk kasus Split Delivery VRP
5. Tipe 4 untuk kasus VRP dengan Time Windows
6. Tipe 5 untuk kasus Periodic VRP dengan Time Windows
7. Tipe 6 untuk kasus Multi-Depot VRP dengan Time Windows
8. Tipe 7 untuk kasus Split Delivery VRP dengan Time Windows

Adapun penjelasan format dari dataset Cordeau tipe 2 adalah sebagai berikut:

1. Baris pertama berformat **TYPE M N T**, dimana:
 - M = Jumlah *vehicle*
 - N = Jumlah *customer*
 - T = Jumlah *depot*
2. Baris kedua sampai T baris berikutnya berformat **D Q**, dimana:
 - D = Durasi maksimum dari setiap rute
 - Q = Kapasitas maksimum dari setiap *vehicle*
3. Baris selanjutnya sampai M baris berikutnya berformat **i x y d q f a list e l**, dimana:
 - i = nomer *customer*
 - x = koordinat x
 - y = koordinat y
 - d = durasi pelayanan (*service time*)
 - q = *demand*
 - f = frekuensi kunjungan
 - a = jumlah kombinasi kunjungan
 - list = list dari semua kombinasi kunjungan
 - e = jika ada, waktu dimulainya kunjungan
 - l = jika ada, waktu selesainya kunjungan

4. Baris selanjutnya sampai M baris berikutnya berformat **i x y**, dimana:

i = nomer *vehicle*

x = koordinat depot x

y = koordinat depot y

Selain menggunakan data Cordeau tipe 2, sistem juga diuji dengan menggunakan data lapangan. Data lapangan digunakan untuk merepresentasikan kondisi pencacahan yang sebenarnya, dimana antar lokasi (*node*) terdapat jarak dan waktu tempuh yang digunakan untuk merepresentasikan *cost*.

Data lapangan yang digunakan meliputi 182 lokasi pencacahan (*customer*) (*N*) beserta koordinatnya, 15 pencacah (kendaraan) (*M*) beserta koordinat *initial-depot* masing-masing pencacah. Dari seluruh lokasi pencacahan dan *initial-depot* dari pencacah, dihitung waktu tempuh dari seluruh kombinasi dengan memanfaatkan *Google Direction API*, sebagaimana terdapat pada subsubsection 4.1.1.3.

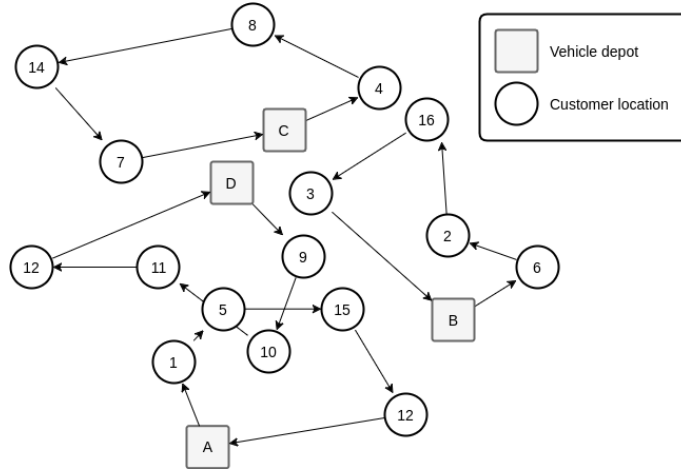
5.2.2.2 *Metric*

Data yang digunakan dalam pengujian kemudian akan diuji pada sistem usulan yang menggunakan algoritma MDVRP berbasis CoEAs dan mekanisme Publish/-Subscribe. Hasil ujicoba kemudian dibandingkan dengan hasil yang diperoleh dari program pembanding yang menggunakan algoritma MDVRP berbasis CoEAs tanpa mekanisme Publish/Subscribe. Dari masing-masing pengujian tersebut akan diperoleh output yang berupa rute untuk masing-masing pencacah (kendaraan) dengan contoh sebagai berikut:

- *Vehicle A* = Loc1 → Loc5 → Loc15 → Loc12
- *Vehicle B* = Loc 6 → Loc2 → Loc16 → Loc3
- *Vehicle C* = Loc4 → Loc8 → Loc14 → Loc 7
- *Vehicle D* = Loc9 → Loc10 → Loc11 → Loc12

Total *cost* untuk masing-masing rute kemudian dikalkulasi. *Cost* dihitung dalam satuan waktu. Untuk setiap rute, total *cost* tersusun atas penjumlahan seluruh waktu tempuh dan waktu pelayanan (*service time*) dari lokasi yang dikunjungi. Pada contoh Figure 5.1, total *cost* dari setiap kendaraan adalah:

- $TC_A = C_{A-1} + ST_5 + C_{1-5} + ST_5 + C_{5-15} + ST_{15} + C_{15-12} + ST_{12} + C_{12-A}$
- $TC_B = C_{B-6} + ST_6 + C_{6-2} + ST_2 + C_{2-16} + ST_{16} + C_{16-3} + ST_3 + C_{3-B}$



Gambar 5.1: Ilustrasi Rute yang Dihasilkan

- $TC_C = C_{C-4} + ST_4 + C_{4-8} + ST_8 + C_{8-14} + ST_{14} + C_{14-7} + ST_7 + C_{7-C}$
- $TC_D = C_{D-9} + ST_9 + C_{9-10} + ST_{10} + C_{10-11} + ST_{11} + C_{11-12} + ST_{12} + C_{12-D}$

dimana TC adalah *total cost*, C adalah *transport cost* yang direpresentasikan dengan waktu tempuh, dan ST adalah *service time*.

Metric atau ukuran yang akan digunakan dalam perbandingan adalah standar deviasi dari seluruh rute. Standar deviasi dipilih sebagai ukuran perbandingan karena merepresentasikan kondisi pencacahan yang sebenarnya, dimana semakin kecil variasi waktu dari seluruh pencacah, maka penyelesaian pencacahan akan semakin merata dan penyelesaian pencacahan secara keseluruhan akan lebih cepat. Sistem yang lebih baik akan menghasilkan standar deviasi yang lebih kecil.

5.2.3 Skenario dan Hasil Pengujian

Untuk memastikan program dapat bekerja dengan baik, selain menggunakan data yang bervariasi, juga akan digunakan berbagai variasi skenario pengujian. *Setup* pengujian dan skenario yang diujicobakan pada penelitian ini beserta hasilnya dijelaskan dibawah ini.

5.2.3.1 Message Broker Setup

Seluruh skenario dalam pengujian ini akan menggunakan Redis sebagai *message broker*. Redis merupakan *in-memory data structure store* yang dapat digunakan sebagai *database*, *cache*, dan *message broker* (redis, 2017a). Redis mempunyai *feature* Redis Cluster, sehingga dapat dengan mudah disusun menjadi *distributed*

message broker. Selain itu, Redis *client* tersedia dalam sebagian besar bahasa pemrograman (redis, 2017b), sehingga memungkinkan pemilihan bahasa pemrograman dalam implementasi sistem secara lebih fleksibel.

Pada penelitian ini, Redis Cluster dikonfigurasi sebanyak 6 (enam) buah *nodes*, 3 (tiga) node digunakan sebagai *master* dan sisanya sebagai *slave*. Konfigurasi Redis Cluster pada setiap node sebagaimana terdapat pada 5.1. Pada konteks Redis, master dapat dianggap sebagai partisi, dan slave dianggap sebagai replikasi dari master. Seluruh node, baik master maupun slave dapat digunakan sebagai *entry-point*, dimana client terkoneksi.

```
cluster-enabled yes
cluster-node-timeout 5000
cluster-config-file nodes.conf
appendonly yes
dir /data
```

Kode 5.1: Konfigurasi Redis Cluster

Setelah seluruh Redis *nodes* dikonfigurasi dan dijalankan, maka kemudian *cluster* dapat dibuat dengan menyertakan *nodes* tersebut. Pembuatan *cluster* menggunakan *script redis-trib.rb* yang telah disediakan oleh redis. *Command* yang dieksekusi sebagaimana terdapat pada 5.2 berikut. Sementara *response* yang diperoleh adalah sebagaimana terdapat pada 5.3. Figure 5.2a menunjukkan alur Redis Cluster dibuat dan dijalankan.

```
/redis-trib.rb create --replicas 1 172.17.0.3 172.17.0.4 172.17.0.5
↪ 172.17.0.6 172.17.0.7 172.17.0.8
```

Kode 5.2: Pembuatan Redis Cluster

5.2.3.2 *Publisher Setup*

Publisher dikonfigurasi dan dijalankan pada lingkungan yang telah didefinisikan pada subsection 5.2.1. Publisher dijalankan sesuai dengan alur yang digambarkan pada Figure 5.2b. 5.4 menunjukkan format penggunaan dari Publisher.

5.2.3.3 *Subscriber Setup*

Pada masing-masing pengujian juga dibuat sebuah program tambahan yang berperan sebagai pencacah. Seluruh pencacah akan berperan sebagai subscriber.

```

Creating cluster
Performing hash slots allocation on 6 nodes...
Using 3 masters:
172.17.0.3:6379
172.17.0.4:6379
172.17.0.5:6379
Adding replica 172.17.0.6:6379 to 172.17.0.3:6379
Adding replica 172.17.0.7:6379 to 172.17.0.4:6379
Adding replica 172.17.0.8:6379 to 172.17.0.5:6379
M: 2f0c681921fe52900a6774fb2cc808a8c4e69216 172.17.0.3:6379
slots:0-5460 (5461 slots) master
M: 41a343142847138301ceeb710206284a50bb44a0 172.17.0.4:6379
slots:5461-10922 (5462 slots) master
M: 88ae20b9e75dd5aa58973f13aa89479f52cedfd3 172.17.0.5:6379
slots:10923-16383 (5461 slots) master
S: c5f6764d82ed10793c0c81e54830b4f68bleacd7 172.17.0.6:6379
replicates 2f0c681921fe52900a6774fb2cc808a8c4e69216
S: 5906f61963d4a5a45478736d976b79db4280b3c7 172.17.0.7:6379
replicates 41a343142847138301ceeb710206284a50bb44a0
S: 635ed817b134b5d14bffd61fe9089867037fee8c 172.17.0.8:6379
replicates 88ae20b9e75dd5aa58973f13aa89479f52cedfd3
Can I set the above configuration? (type ☐yes☐ to accept): yes
Nodes configuration updated
Assign a different config epoch to each node
Sending CLUSTER MEET messages to join the cluster
Waiting for the cluster to join...
Performing Cluster Check (using node 172.17.0.3:6379)
M: 2f0c681921fe52900a6774fb2cc808a8c4e69216 172.17.0.3:6379
slots:0-5460 (5461 slots) master
1 additional replica(s)
M: 88ae20b9e75dd5aa58973f13aa89479f52cedfd3 172.17.0.5:6379
slots:10923-16383 (5461 slots) master
1 additional replica(s)
S: c5f6764d82ed10793c0c81e54830b4f68bleacd7 172.17.0.6:6379
slots: (0 slots) slave
replicates 2f0c681921fe52900a6774fb2cc808a8c4e69216
M: 41a343142847138301ceeb710206284a50bb44a0 172.17.0.4:6379
slots:5461-10922 (5462 slots) master
1 additional replica(s)
S: 635ed817b134b5d14bffd61fe9089867037fee8c 172.17.0.8:6379
slots: (0 slots) slave
replicates 88ae20b9e75dd5aa58973f13aa89479f52cedfd3
S: 5906f61963d4a5a45478736d976b79db4280b3c7 172.17.0.7:6379
slots: (0 slots) slave
replicates 41a343142847138301ceeb710206284a50bb44a0
[OK] All nodes agree about slots configuration.
Check for open slots...
Check slots coverage...
[OK] All 16384 slots covered.

```

Kode 5.3: Respon Pembuatan Redis Cluster

Alur kerja dari masing-masing *subscriber*, sebagaimana digambarkan pada Figure 5.2c adalah sebagai berikut:

1. Lakukan *subscription* pada *message broker* dengan topik *current location*,
2. Sebuah pesan akan diterima, dimana pesan dari *publisher* yang dipublish melalui *message broker* menunjukkan lokasi berikutnya yang akan dikunjungi,


```

Usage: mdvrp-redis-producer [options]

Run location recommendation server

Options:
-h, --help            show this help message and exit
-b BIN, --coes-bin=BIN
                      Binary of CoES MDVRP library
-D D, --data=D        Data problem file
-C C, --cost-file=C    Cost matrix file
-O O, --output-dir=O   Output directory
-t, --use-timestamp    Append timestamp to output directory
-B B, --broker-url=B   Redis broker URL
-X X, --solver-execution-time=X
                      VRP Solver execution time

```

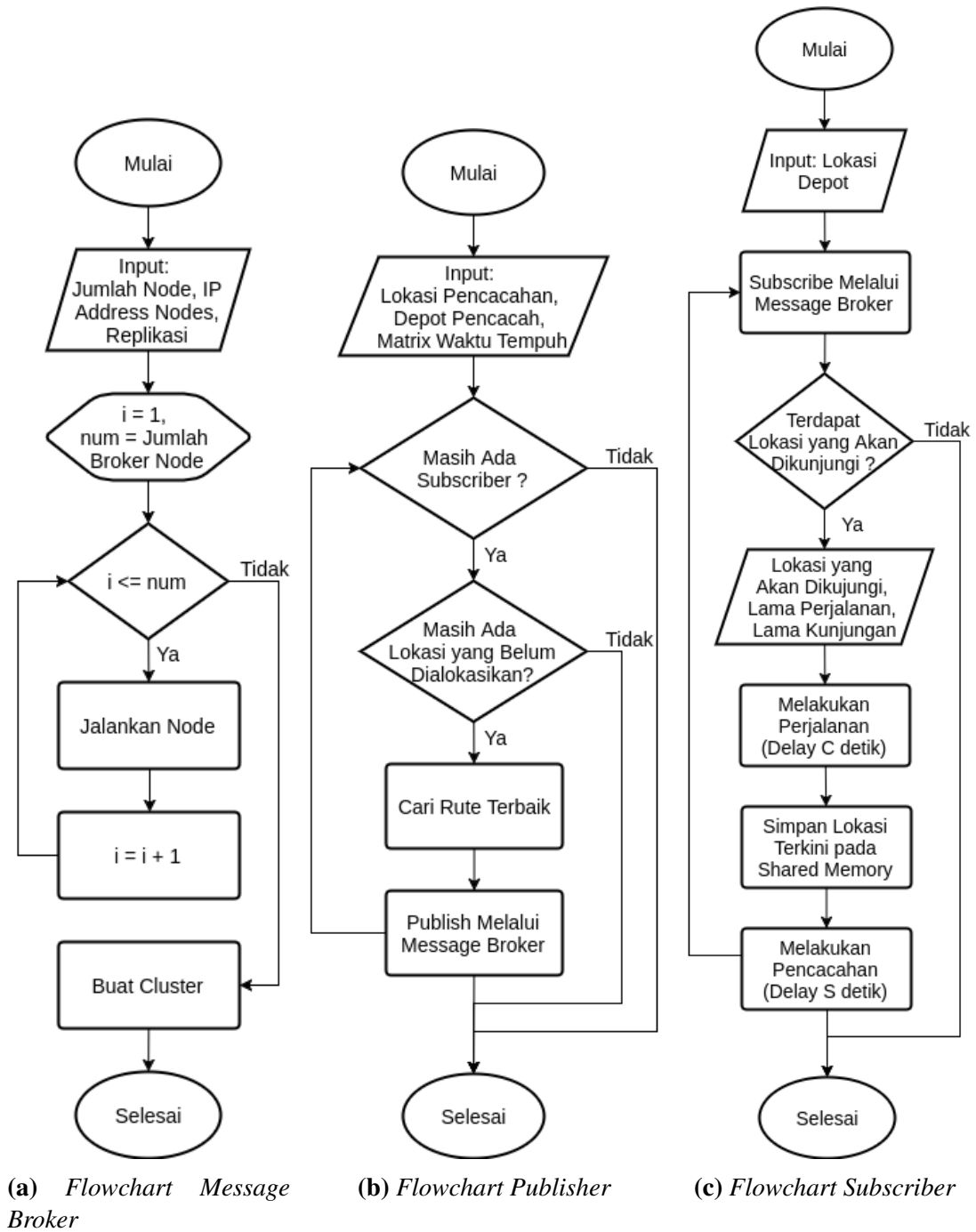
Kode 5.4: Format penggunaan Publisher

3. Simulasikan perjalanan ke lokasi yang akan dikunjungi dengan *delay*,
4. Setelah sampai ke lokasi, maka simpan *current location* pada *shared memory*,
5. Simulasikan pencacahan dengan *delay*,
6. Ulangi lagi dari langkah pertama.

5.2.3.4 Pengujian Tanpa *Service Time*

Pengujian ini bertujuan membandingkan keakuratan sistem dalam memproses data yang tidak memasukkan *service time* pada *customernya*. Pada pengujian ini digunakan data dimana kendaraan dan *customer* digenerate secara random, baik secara jumlah maupun koordinat. Data yang digunakan diperoleh dari *instance* Cordeau P01 sampai P10.

Dari hasil simulasi, diperoleh rute untuk masing-masing *instance* Cordeau sebagaimana digambarkan pada Lampiran 1. Kemudian dari seluruh rute yang diperoleh, dikalkulasi *total time* untuk masing-masing rute, sebagaimana cara yang dijelaskan pada subsection 5.2.2.2. Berdasarkan Table 5.1, diperoleh hasil bahwasannya dengan menggunakan sistem usulan hanya 2 dari 10 *instance* menghasilkan **total waktu** yang lebih kecil. Sementara dari sisi **standar deviasi**, diperoleh hasil 8 dari 10 *instance* menghasilkan standar deviasi yang lebih kecil. Hal ini berarti, program pembanding, yaitu MDVRP berbasis CoEAs, lebih efisien, tetapi program usulan, yaitu MDVRP berbasis CoEAs yang dikombinasikan dengan mekanisme Publish/Subscribe lebih merata. Figure 5.3 dan Figure 5.4 menggambarkan perbandingan hasil pengujian dari 10 *instance* Cordeau.

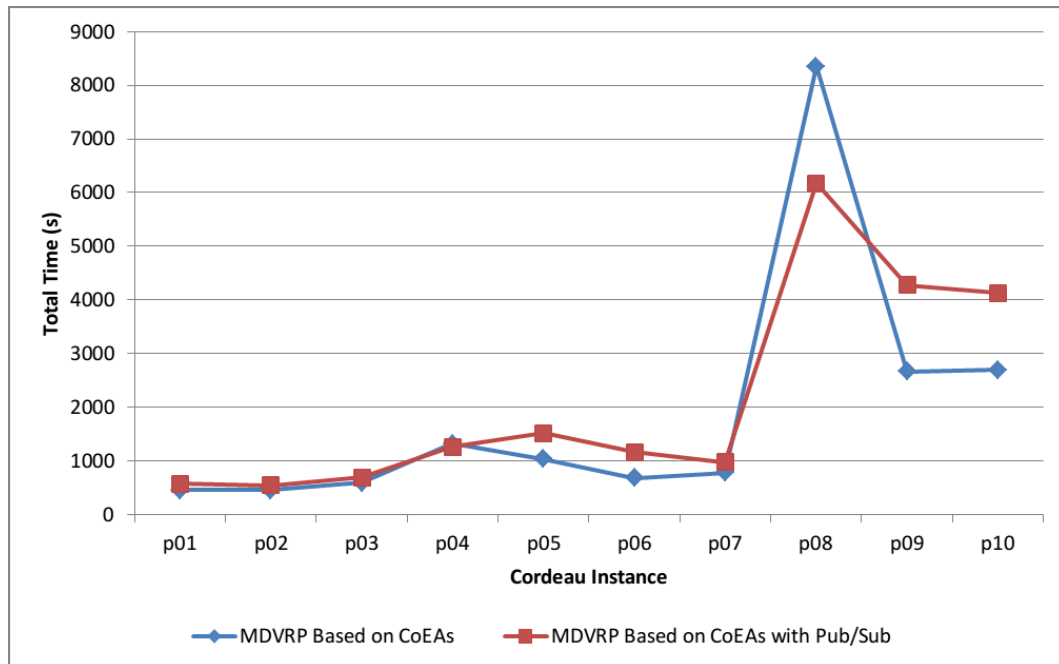


Gambar 5.2: Flowchart Subsistem pada Pengujian

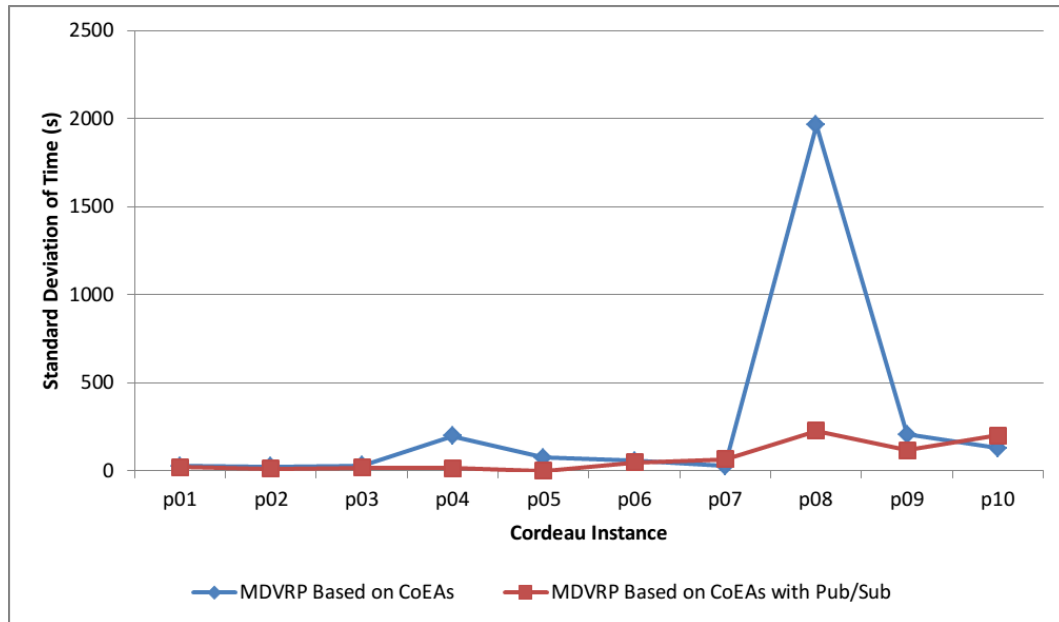
Tabel 5.1: Hasil Pengujian Tanpa *Service Time* pada Data Cordeau

<i>Instance</i>	Total CoES MDVRP	Total CoES MDVRP + Pub/Sub	Stdev CoES MDVRP	Stdev CoES MDVRP + Pub/Sub
p01	454.91	573.77	27.24	20.84

<i>Instance</i>	Total CoES MDVRP	Total CoES MDVRP + Pub/Sub	Stdev CoES MDVRP	Stdev CoES MDVRP + Pub/Sub
p02	462.61	550.61	25.90	13.32
p03	590.98	696.93	32.47	20.09
p04	1.321.89	1.264.53	199.16	16.10
p05	1.039.68	1.520.53	76.61	1.09
p06	679.11	1.160.76	57.15	48.82
p07	782.41	973.72	29.08	67.27
p08	8.353.76	6.172.66	1.962.81	228.95
p09	2.669.29	4.275.12	208.99	117.76
p10	2.692.06	4.132.31	130.08	200.31



Gambar 5.3: Perbandingan Waktu Total dari 10 *Instance* Cordeau



Gambar 5.4: Perbandingan Standar Deviasi Waktu dari 10 *Instance* Cordeau

5.2.3.5 Pengujian Kondisi Normal dengan *Service Time*

Skenario pengujian kondisi normal dimaksudkan untuk membandingkan sistem yang dijalankan pada kondisi normal. Yang dimaksud kondisi normal disini merupakan kondisi dimana tidak ada yang menyebabkan penundaaan, seperti permasalahan koneksi. Pengujian kondisi ini akan dilakukan satu kali untuk masing-masing data set, Cordeau P01 dan data lapangan.

Sementara data *service time* digenerate secara random dengan mengikuti komposisi dari (Sudman, 1965), yaitu:

1. 21 persen dari total waktu untuk perpindahan antar segmen,
2. 15 persen dari total waktu untuk perpindahan antar rumah tangga dalam segmen,
3. 37 persen dari total waktu untuk wawancara seluruh responden, dan
4. 27 persen untuk hal-hal yang lain, seperti pengenalan wilayah dan perbaikan data.

Service time merupakan gabungan dari waktu perpindahan antar rumah tangga dan waktu wawancara.

Setelah simulasi dilakukan, dari seluruh rute yang diperoleh kemudian dikalkulasi *total time* untuk masing-masing rute, sebagaimana cara yang dijelaskan pada

subsubsection 5.2.2.2. Selain itu juga sebuah grafik dibuat untuk memberikan ilustrasi rute pada masing-masing program pembandingan dan sistem usulan.

Pada pengujian dengan dataset Cordeau, diperoleh hasil sebagaimana Table 5.2 untuk algoritma MDVRP berbasis CoEAs tanpa mekanisme Publish/Subscribe, dan Table 5.3 untuk algoritma MDVRP berbasis CoEAs dengan mekanisme Publish/Subscribe. Sementara Figure 5.5 menggambarkan rute yang diperoleh untuk masing-masing program.

Berdasarkan Table 5.4, diperoleh hasil bahwasannya **total waktu** yang diperoleh dengan menggunakan sistem MDVRP berbasis CoEAs dan Publish/Subscribe lebih buruk 6,25 persen dibandingkan dengan program usulan. Akan tetapi dari sisi **standar deviasi** dari keseluruhan rute diperoleh hasil metode CoEAs yang dikombinasikan dengan Publish/Subscribe menghasilkan angka yang lebih baik 68.83 persen, yang artinya total waktu dari seluruh rute lebih merata.

Tabel 5.2: Hasil Pengujian Kondisi Normal (Cordeau), MDVRP berbasis CoEAs

<i>Vehicle</i>	Rute	Waktu Total
51	42-19-40-41-13-18-4-17-37-15-44	194,775.99
52	27-48-6-14-25-24-43-23-7-26-8-1-32-11-12-47-46	312,308.79
53	9-38-5-45-33-39-10-30-34-50-49	198,178.68
54	29-21-16-2-22-28-31-3-20-35-36	199,754.93

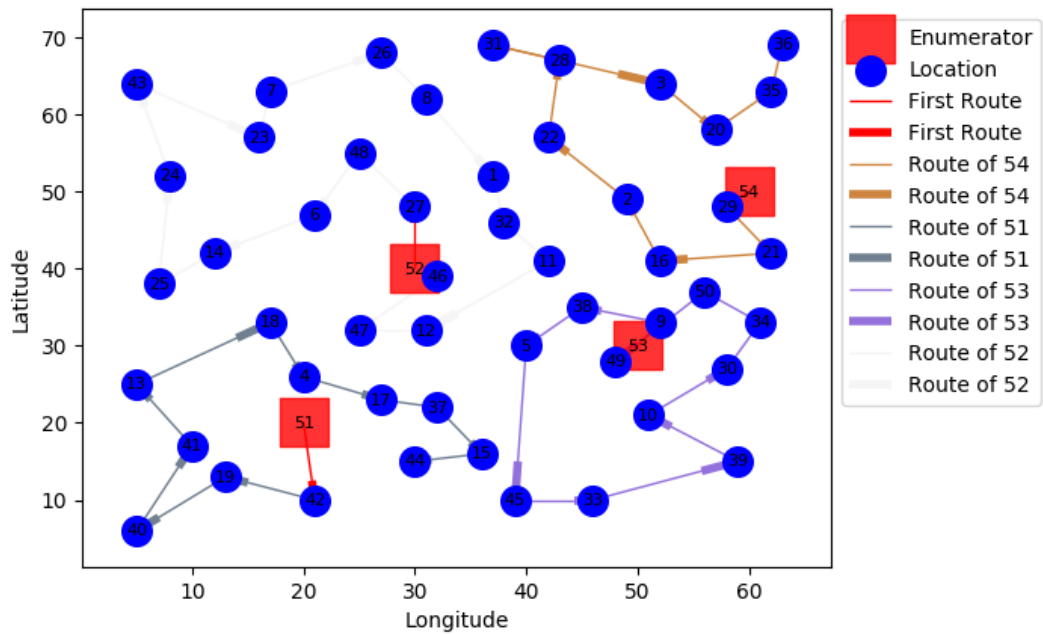
Tabel 5.3: Hasil Pengujian Kondisi Normal (Cordeau), MDVRP berbasis CoEAs dan Publish/Subscribe

<i>Vehicle</i>	Rute	Waktu Total
51	42-19-44-40-45-33-15-37-39-30-17-25-43	234,931.69
52	27-46-12-47-18-24-4-41-6-48-13-14-23	236,669.22
53	9-50-34-49-5-10-38-11-16-2-32	199,526.52
54	29-21-20-35-36-31-28-3-22-26-1-8-7	234,154.12

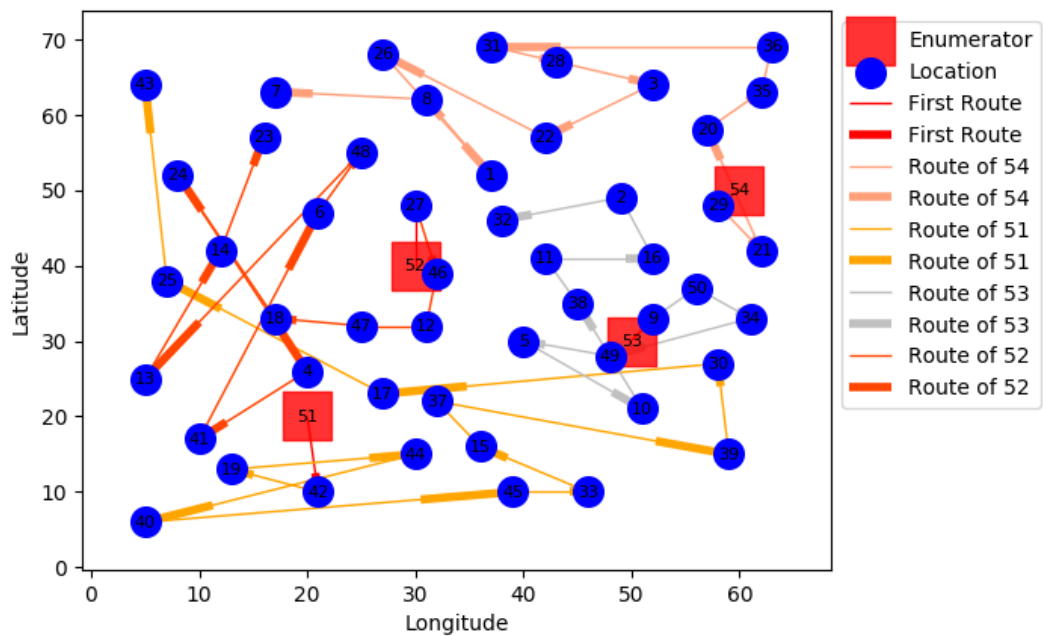
Tabel 5.4: Komparasi Hasil Pengujian Kondisi Normal Dengan Data Cordeau

<i>Parameter</i>	CoEAs MDVRP	Pub/Sub CoEAs MDVRP
Total Waktu	905,018.39	905,281.55
Rata-rata	226,254.60	226,320.39

<i>Parameter</i>	CoEAs MDVRP	Pub/Sub CoEAs MDVRP
Standar Deviasi	49,715.99	15,496.22



(a) MDVRP berbasis CoEAs



(b) MDVRP berbasis CoEAs dengan Publish/Subscribe

Gambar 5.5: Perbandingan Rute Hasil Pengujian Data Cordeau P01 Dengan *Service Time*

Sementara itu, pada pengujian dengan menggunakan data lapangan, diperoleh hasil sebagaimana Table 5.5 untuk algoritma MDVRP berbasis CoEAs tanpa

mekanisme Publish/Subscribe, dan Table 5.6 untuk algoritma MDVRP berbasis Co-EAs dengan mekanisme Publish/Subscribe. Sementara ?? menggambarkan rute yang diperoleh untuk masing-masing program.

Berdasarkan Table 5.7, diperoleh hasil bahwasannya **total waktu** yang diperoleh dengan menggunakan sistem MDVRP berbasis CoEAs dan Publish/Subscribe lebih buruk 8,25 persen dibandingkan dengan program usulan. Akan tetapi dari sisi **standar deviasi** dari keseluruhan rute diperoleh hasil metode CoEAs yang dikombinasikan dengan Publish/Subscribe menghasilkan angka yang lebih baik 59,11 persen, yang artinya total waktu dari seluruh rute lebih merata.

Tabel 5.5: Hasil Pengujian Kondisi Normal (Lapangan), MDVRP berbasis CoEAs

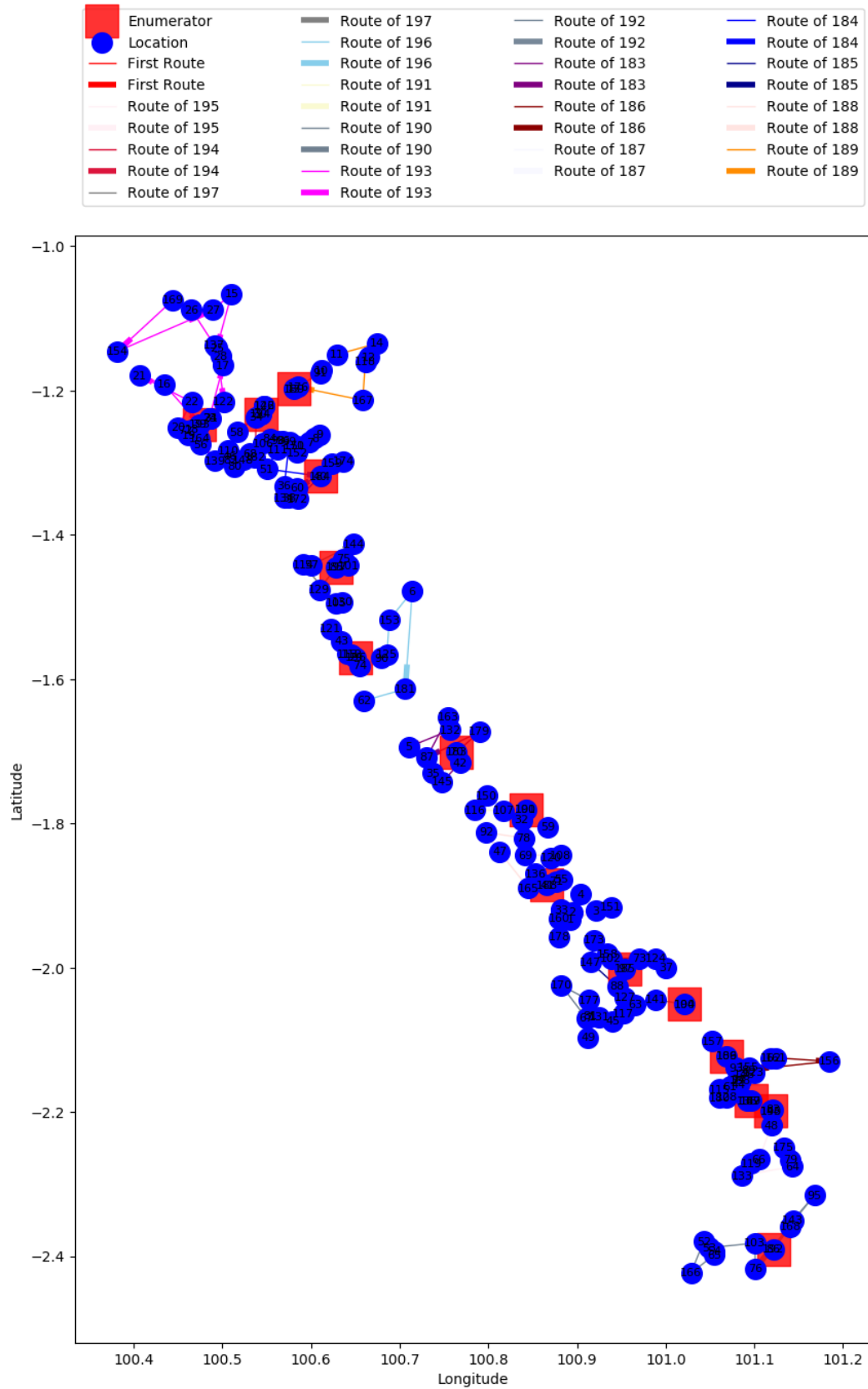
<i>Vehicle</i>	Rute	Waktu Total
183	70-179-87-163-132-5-35-145-42	170,349.48
184	159-174-60-36-99-98-171-50-9-7-8-152-111-51-40-172-38-134	331,759.24
185	127-88-147-173-158-102-124-37-73-97	184,993.17
186	157-93-94-146-155-123-162-161-156-89-138-77-44-109	254,089.65
187	149-135-61-115-180-128	109,758.28
188	165-47-92-78-69-108-120-136-41-71-55-4-33-178-160-1-151-3-2	355,614.60
189	13-91-10-11-14-12-118-167-176	168,298.11
190	104-141-63-117-45-131-81-177-170-67-49	193,952.81
191	100-107-116-150-32-59	116,272.15
192	86-168-143-95-76-103-53-52-166-65-54	211,842.84
193	164-139-56-18-72-19-20-22-21-16-39-24-23-17-28-137-26-169-154-27-15-25-122-31	451,562.34
194	142-126-30-58-110-46-82-80-148-106-96-84-68-182-34	275,108.47
195	83-140-175-79-64-133-119-66-48	160,970.13
196	29-112-113-43-121-74-90-125-153-6-181-62	220,804.42
197	85-129-130-105-114-57-144-75-101	168,014.48

Tabel 5.6: Hasil Pengujian Kondisi Normal (Lapangan), MDVRP berbasis CoEAs dan Publish/Subscribe

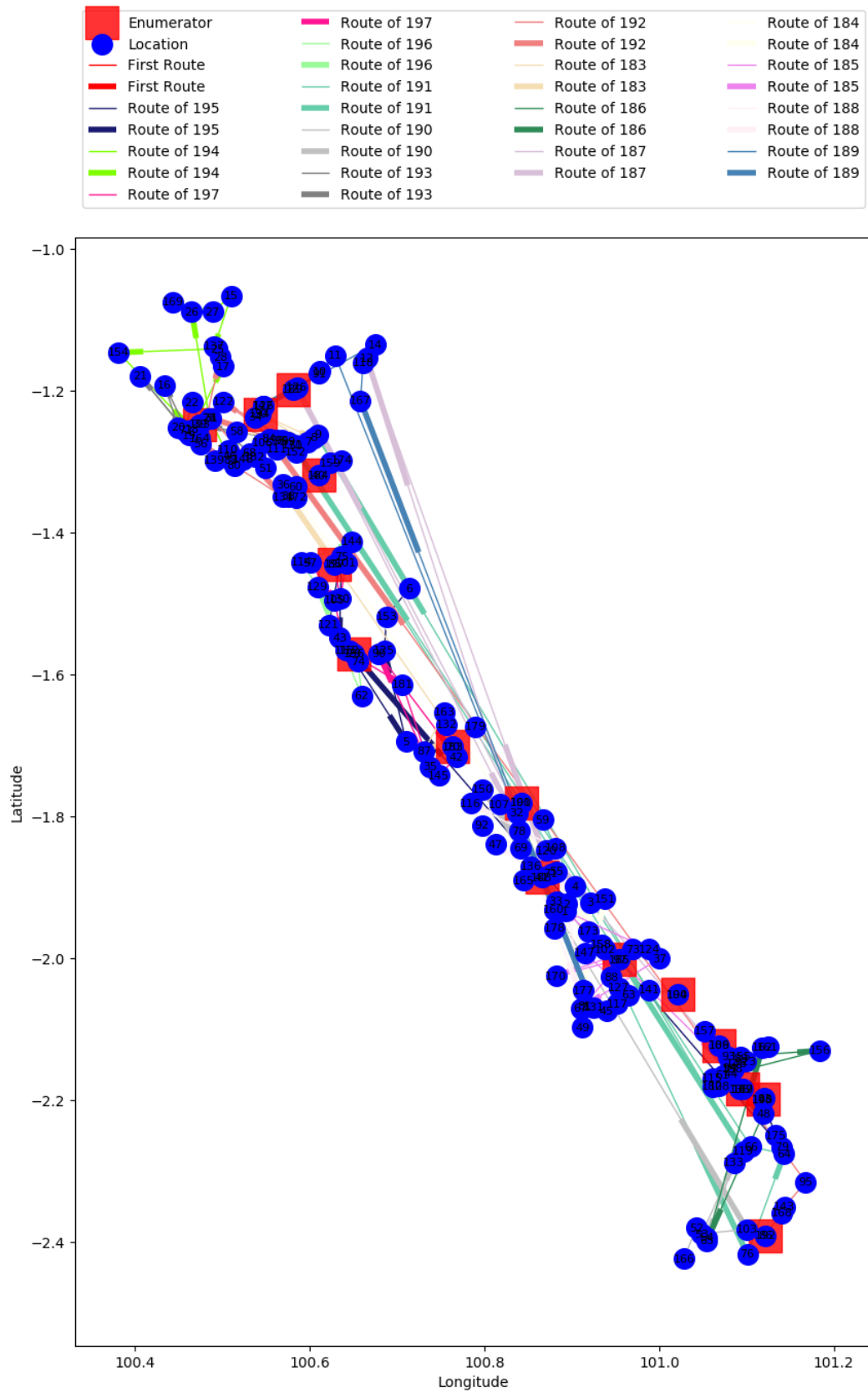
<i>Vehicle</i>	<i>Rute</i>	<i>Waktu Total</i>
183	70-179-87-163-132-38-51-68-96-139-58-7-30-8	277,856.33
184	159-36-174-40-60-172-102-178-49-3-177-151	241,245.62
185	127-88-147-173-158-1-124-37-67-73-170-97-77	253,954.73
186	157-109-93-156-44-155-89-123-161-115-180-128-48-65-162	280,828.06
187	149-135-61-94-146-138-120-136-12-71-176-33	269,049.32
188	165-41-47-69-78-81-108-116-92-55-4-160-2	257,633.11
189	13-91-10-11-131-167-118-14-126-110-148	229,909.92
190	104-141-117-45-63-150-103-53-133-54-52-166	242,338.23
191	100-107-32-59-76-64-66-171-119-9	246,133.18
192	86-168-143-95-84-152-98-122-50-31-134-56-28	275,294.42
193	164-24-182-106-111-99-39-22-72-21-19-16-20	258,783.38
194	142-34-46-80-82-23-26-169-27-15-137-25-154-18-17	284,033.04
195	83-140-175-79-121-75-43-5-125-153-6	220,121.19
196	29-112-74-62-129-114-144	132,181.37
197	85-130-101-57-105-113-181-35-90-42-145	207,356.29

Tabel 5.7: Komparasi Hasil Pengujian Kondisi Normal Dengan Data Lapangan

<i>Parameter</i>	<i>CoEAs MDVRP</i>	<i>Pub/Sub CoEAS MDVRP</i>
Total Waktu	3,373,390.19	3,676,718.19
Rata-rata	224,892.68	245,114.55
Standar Deviasi	91,123.23	37,261.85



Gambar 5.6: Rute Hasil Pengujian Data Lapangan Pada MDVRP berbasis CoEAs

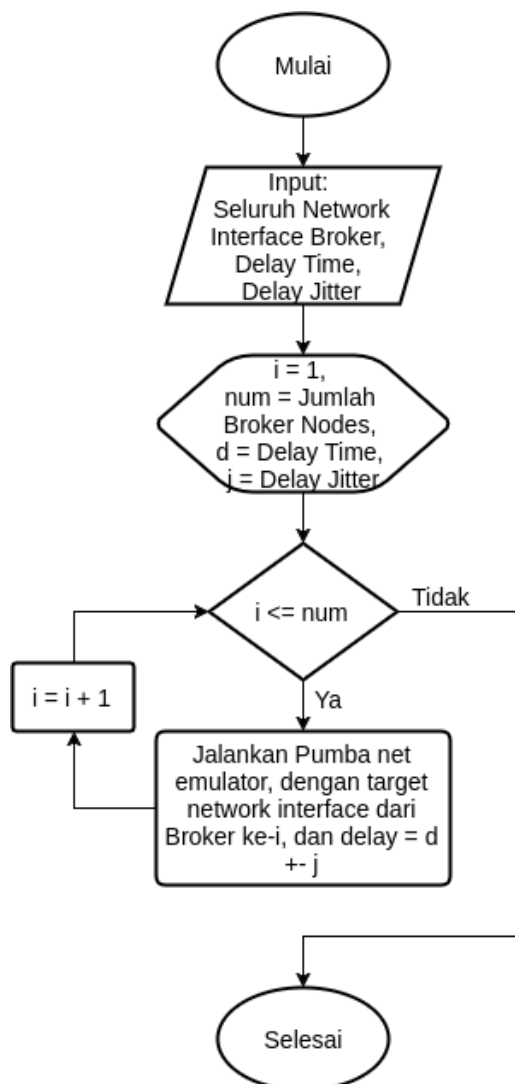


Gambar 5.7: Rute Hasil Pengujian Data Lapangan Pada MDVRP berbasis CoEAs dengan Publish/-Subscribe

5.2.3.6 Pengujian Kondisi *Delay* pada Jaringan dengan *Service Time*

Skenario pengujian kondisi *delay* pada jaringan dimaksudkan untuk membandingkan program yang dijalankan pada kondisi dimana terjadi *delay* secara random. Pengujian ini digunakan sebagai cerminan kondisi lapangan dimana pada beberapa lokasi tidak terdapat koneksi yang stabil. Pengujian akan dilakukan satu kali untuk data lapangan. Sementara data *service time* digenerate secara random dengan mengikuti komposisi dari (Sudman, 1965) sebagaimana dijelaskan pada subsubsection 5.2.3.5.

Pada pengujian ini akan digunakan sebuah software 3rd, yaitu Pumba (?) untuk meng-emulasi-kan *delay*. *Delay* yang digunakan adalah $100\text{ms} \pm 10\%$ pada setiap koneksi yang terjadi. Langkah-langkah yang dilakukan dalam konfigurasi Pumba adalah seperti pada Figure 5.8.



Gambar 5.8: Flowchart Setup Pumba Net Emulator

Berdasarkan hasil pengujian, diperoleh hasil sebagaimana Table 5.8 untuk algoritma MDVRP berbasis CoEAs tanpa mekanisme Publish/Subscribe, dan Table 5.9 untuk algoritma MDVRP berbasis CoEAs dengan mekanisme Publish/Subscribe. Sementara Figure 5.10 menggambarkan rute yang diperoleh untuk masing-masing program.

Berdasarkan Table 5.10, diperoleh hasil bahwasannya **total waktu** yang diperoleh dengan menggunakan sistem MDVRP berbasis CoEAs dan Publish/Subscribe lebih buruk 4,83 persen dibandingkan dengan program usulan. Akan tetapi dari sisi **standar deviasi** dari keseluruhan rute diperoleh hasil metode CoEAs yang dikombinasikan dengan Publish/Subscribe menghasilkan angka yang lebih baik 3,62 persen, yang artinya total waktu dari seluruh rute lebih merata.

Tabel 5.8: Hasil Pengujian Kondisi Network Delay, MDVRP berbasis CoEAs

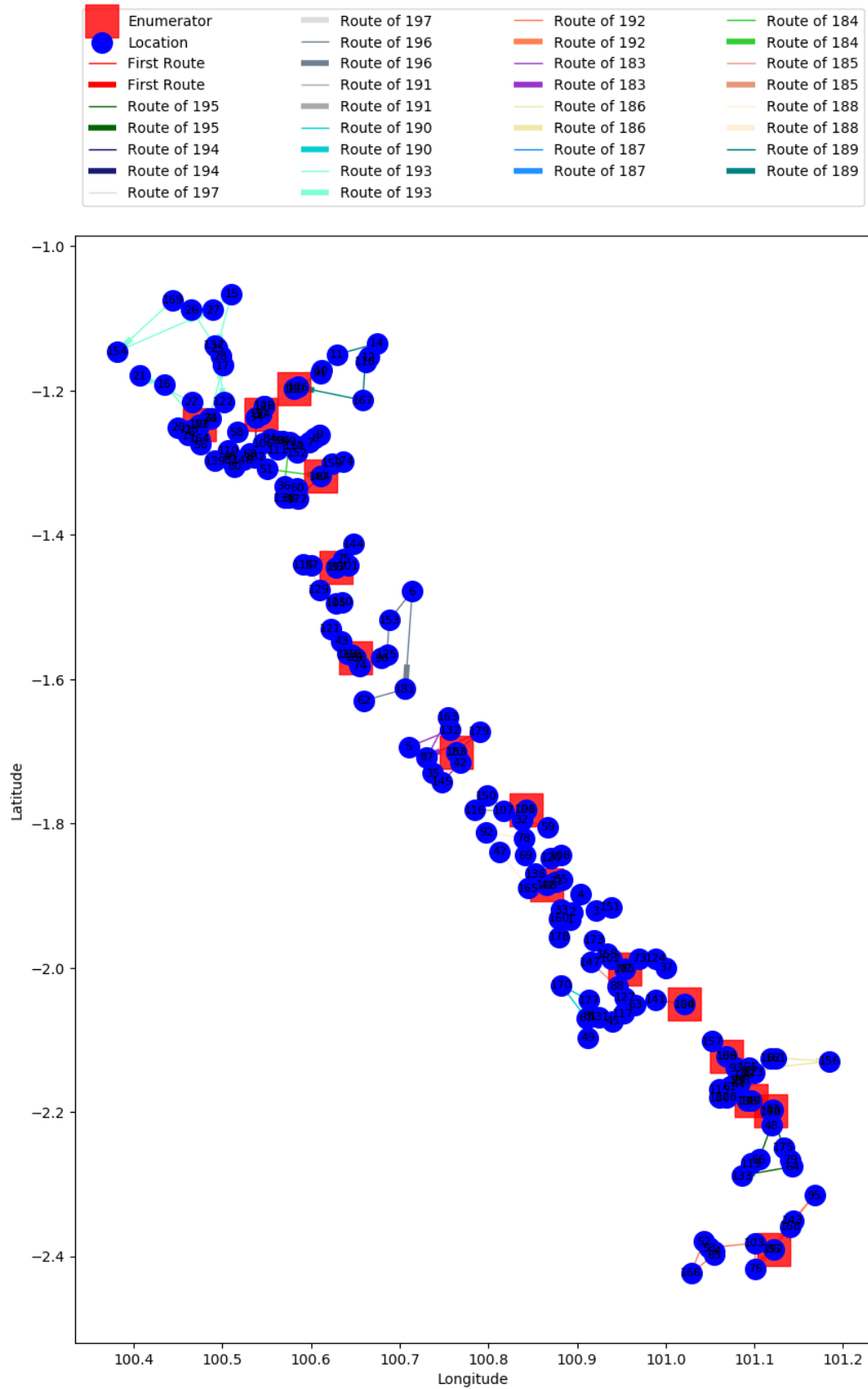
<i>Vehicle</i>	Rute	Waktu Total
183	70-179-87-163-132-5-35-145-42	170,349.48
184	172-38-134-60-36-99-98-171-50-9-7-8-152-111-51-40-159-174	330,854.24
185	127-88-147-173-158-102-124-37-73-97	184,993.17
186	109-157-93-94-146-155-123-162-161-156-89-138-77-44	253,318.65
187	149-135-61-115-180-128	109,758.28
188	41-71-55-4-33-178-160-1-151-3-2-165-47-92-78-69-108-120-136	355,930.60
189	13-91-10-11-14-12-118-167-176	168,298.11
190	104-141-63-117-45-131-81-177-170-67-49	193,952.81
191	100-107-116-150-32-59	116,272.15
192	86-168-143-95-76-103-53-52-166-65-54	211,842.84
193	24-23-17-28-137-26-169-154-27-15-25-122-31-164-139-56-18-72-19-20-22-21-16-39	451,654.34
194	30-58-110-46-82-80-148-106-96-84-68-182-34-142-126	275,108.47
195	83-140-175-79-64-133-119-66-48	160,970.13
196	29-112-113-43-121-74-90-125-153-6-181-62	220,804.42
197	85-129-130-105-114-57-144-75-101	168,014.48

Tabel 5.9: Hasil Pengujian Kondisi Network Delay, MDVRP berbasis CoEAs dan Publish/Subscribe

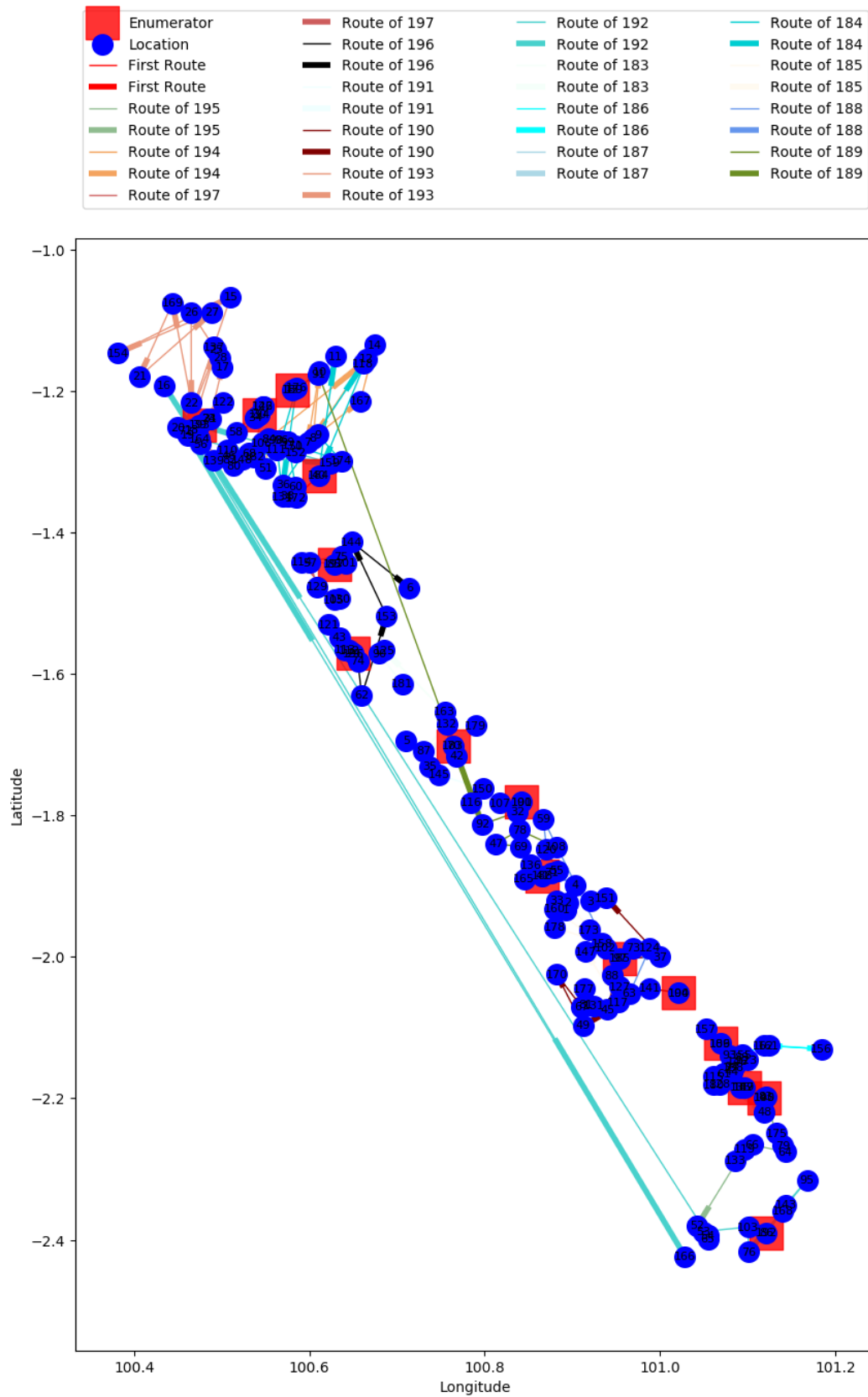
<i>Vehicle</i>	<i>Rute</i>	<i>Waktu Total</i>
183	70-179-87-132-163-125-90-181	151,466.36
184	172-38-60-99-134-98-152-171-111-50-174-9- 80-96-176-36-12-40-11	377,075.90
185	127-88-117-147-173-158-178-1-160-102-2-33	225,310.18
186	109-93-157-146-89-123-155-162-156-161	181,632.75
187	149-135-61-115-180-128-48-94-138-77-44	201,434.48
188	41-165-71-4-136-55-120-59-63-124-73	210,047.14
189	13-91-92-32-69-47-78-108	159,479.98
190	104-141-49-170-67-177-81-131-45-97-37- 151-3	240,785.25
191	100-107-116-150-145-5-35-42	148,534.65
192	86-168-143-95-76-103-53-54-16-166-19-139- 164-68-46-159-39	389,335.84
193	24-31-17-28-137-26-22-169-21-27-154-15-72- 25-18-122-20-23-56-82	394,523.78
194	30-34-110-142-58-126-182-106-51-84-148- 118-8-10-7-167-14	334,526.44
195	83-140-175-79-64-66-133-119-52-65	183,551.19
196	29-112-113-43-121-74-62-153-144-6	194,797.02
197	85-101-57-129-130-105-114-75	150,698.22

Tabel 5.10: Komparasi Hasil Pengujian Kondisi Network Delay Dengan Data Lapangan

<i>Parameter</i>	<i>CoEAs MDVRP</i>	<i>Pub/Sub CoEAS MDVRP</i>
Total Waktu	3,372,122.19	3,543,199.19
Rata-rata	224,808.15	236,213.28
Standar Deviasi	91,081.99	87,783.27



Gambar 5.9: Rute Hasil Pengujian Kondisi *Network Delay* Data Lapangan Pada MDVRP berbasis CoEAs

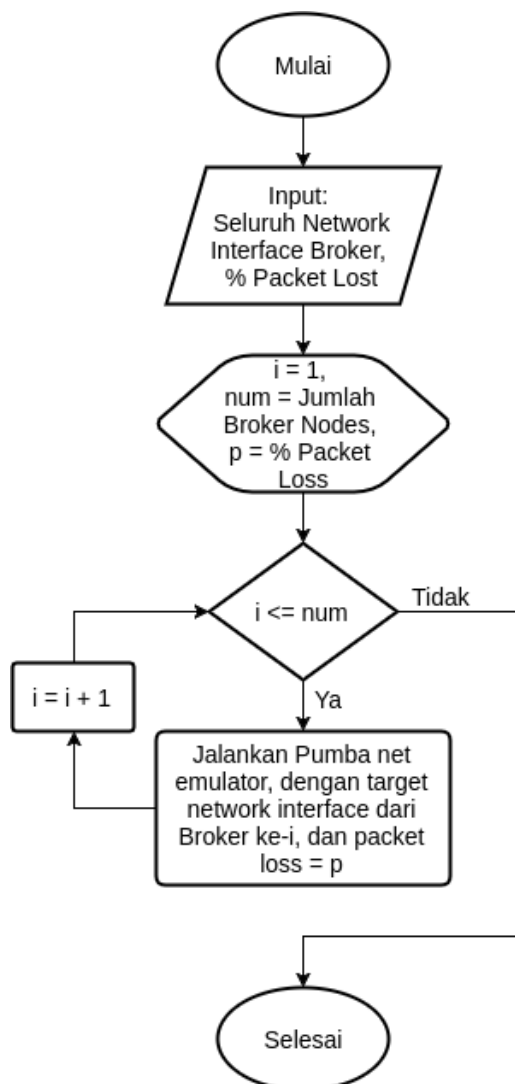


Gambar 5.10: Rute Hasil Pengujian Kondisi *Network Delay* Data Lapangan Pada MDVRP berbasis CoEAs dengan Publish/Subscribe

5.2.3.7 Pengujian Kondisi *Packet Loss* pada Jaringan dengan *Service Time*

Skenario pengujian kondisi *packet loss* pada jaringan dimaksudkan untuk membandingkan program yang dijalankan pada kondisi dimana terjadi *packet loss*. Pengujian ini digunakan sebagai cerminan kondisi lapangan dimana pada beberapa lokasi tidak terdapat koneksi yang stabil. Pengujian akan dilakukan satu kali untuk data lapangan. Sementara data *service time* digenerate secara random dengan mengikuti komposisi dari (Sudman, 1965) sebagaimana dijelaskan pada subsubsection 5.2.3.5.

Pada pengujian ini juga akan digunakan sebuah software 3rd, yaitu Pumba (?) untuk meng-emulasi-kan *packet loss*. Nilai *Packet loss* yang digunakan adalah 10% dari packet yang lewat yang dipilih secara random. Langkah-langkah yang dilakukan dalam konfigurasi Pumba adalah seperti pada Figure 5.8.



Gambar 5.11: Flowchart Setup Pumba Net Emulator

Berdasarkan hasil pengujian, diperoleh hasil sebagaimana Table 5.11 untuk algoritma MDVRP berbasis CoEAs tanpa mekanisme Publish/Subscribe, dan Table 5.12 untuk algoritma MDVRP berbasis CoEAs dengan mekanisme Publish/Subscribe. Sementara ?? menggambarkan rute yang diperoleh untuk masing-masing program.

Berdasarkan ??, diperoleh hasil bahwasannya **total waktu** yang diperoleh dengan menggunakan sistem MDVRP berbasis CoEAs dan Publish/Subscribe lebih buruk 12,79 persen dibandingkan dengan program usulan. Akan tetapi dari sisi **standar deviasi** dari keseluruhan rute diperoleh hasil metode CoEAs yang dikombinasikan dengan Publish/Subscribe menghasilkan angka yang lebih baik 36,14 persen, yang artinya total waktu dari seluruh rute lebih merata.

Tabel 5.11: Hasil Pengujian Kondisi *Packet Loss*, MDVRP berbasis CoEAs

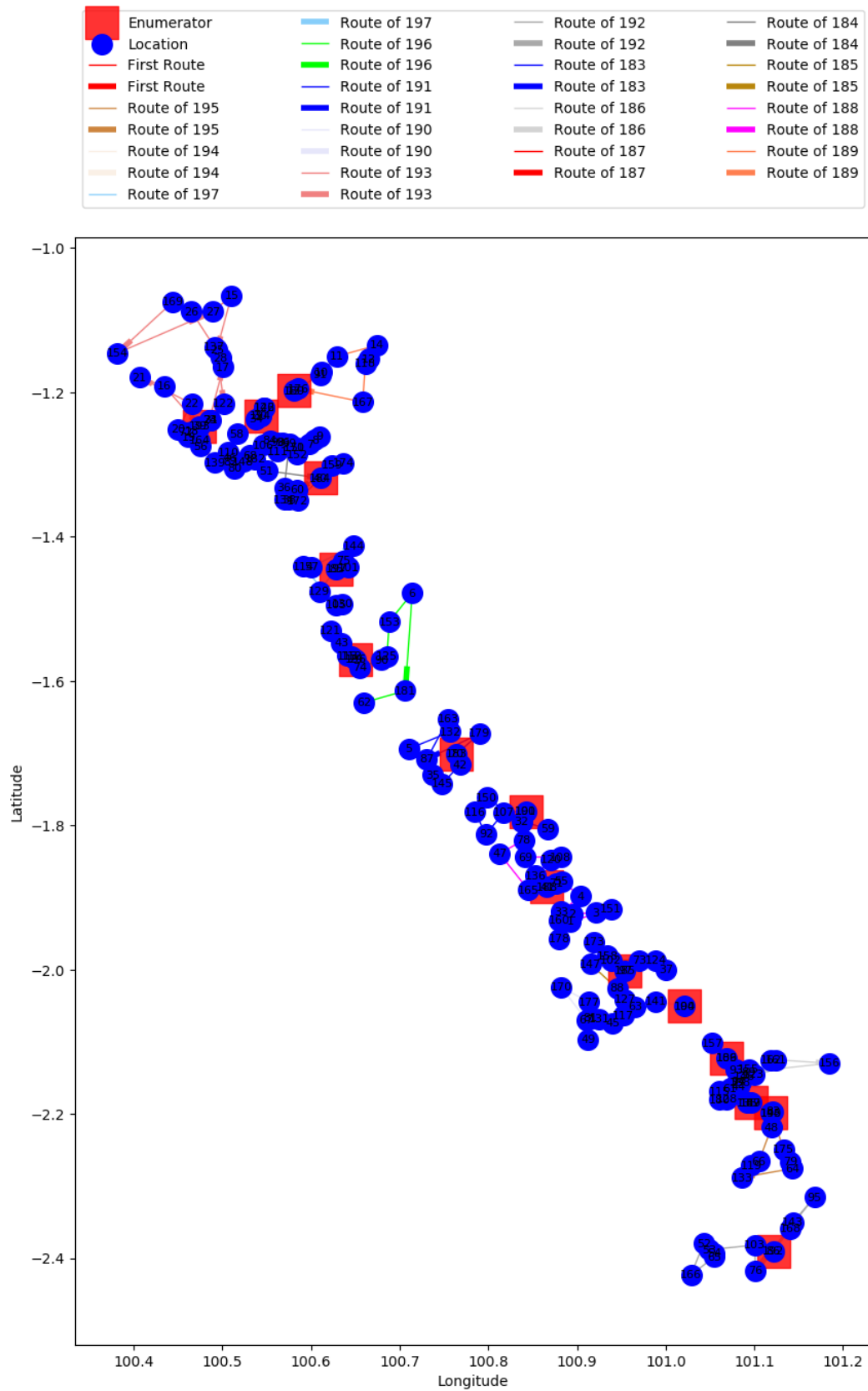
<i>Vehicle</i>	Rute	Waktu Total
183	70-179-87-163-132-5-35-145-42	170,349.48
184	159-174-60-36-99-98-171-50-9-7-8-152-111-51-40-172-38-134	331,759.24
185	127-88-147-173-158-102-124-37-73-97	184,993.17
186	157-93-94-146-155-123-162-161-156-89-138-77-44-109	254,089.65
187	149-135-61-115-180-128	109,758.28
188	41-165-47-78-69-108-120-136-71-55-4-33-178-160-1-151-3-2	337,519.02
189	13-91-10-11-14-12-118-167-176	168,298.11
190	104-141-63-117-45-131-81-177-170-67-49	193,952.81
191	100-107-92-116-150-32-59	134,042.73
192	86-168-143-95-76-103-53-52-166-65-54	211,842.84
193	164-139-56-18-72-19-20-22-21-16-39-24-23-17-28-137-26-169-154-27-15-25-122-31	451,562.34
194	142-126-30-58-110-46-82-80-148-106-96-84-68-182-34	275,108.47
195	83-140-175-79-64-133-119-66-48	160,970.13
196	29-112-113-43-121-74-90-125-153-6-181-62	220,804.42
197	85-129-130-105-114-57-144-75-101	168,014.48

Tabel 5.12: Hasil Pengujian Kondisi *Packet Loss*, MDVRP berbasis CoEAs dan Publish/Subscribe

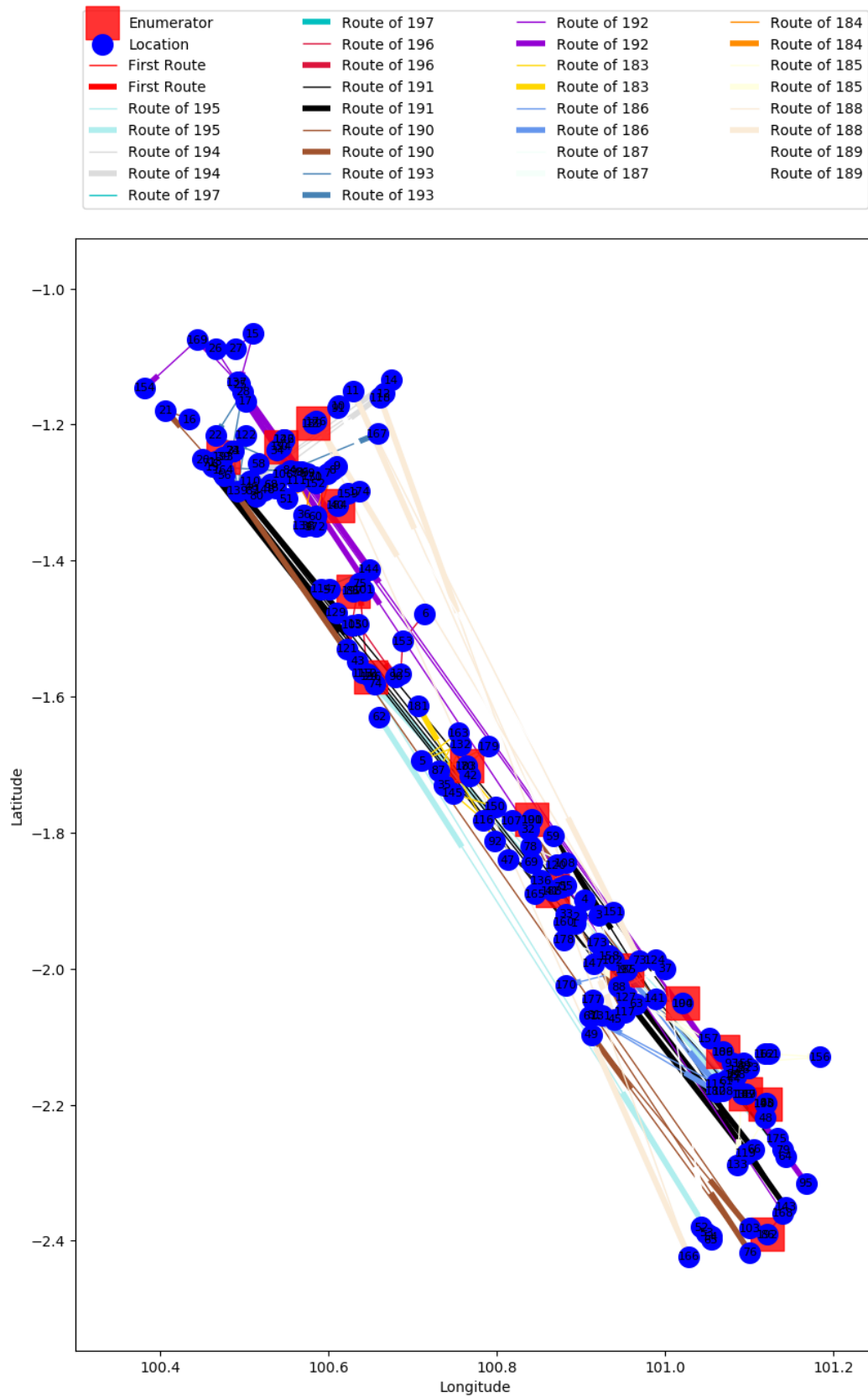
<i>Vehicle</i>	Rute	Waktu Total
183	70-179-87-163-5-132-35-42-150-145-116-92-181	248,263.36
184	159-38-36-134-171-98-99-50-152-51-9-111-60-7-8-172-40-174	337,070.24
185	127-88-93-147-89-123-162-161-156-155-133-48	227,271.63
186	157-109-128-146-138-94-77-44-61-158-115-67-180-97-170	293,534.49
187	149-135-136-47-69-108-78-55-4-1-33-2-160-178-102-3-151	327,824.92
188	41-165-71-120-45-173-11-73-176-166-118	292,972.01
189	13-91-63-10-124-177-37-53-52-65	233,724.33
190	104-141-32-117-131-81-76-49-103-56-21-16	271,621.19
191	100-107-143-59-64-18-66-72-119-20	294,953.97
192	86-168-17-95-137-26-27-15-25-169-154	257,340.44
193	164-31-139-122-24-39-23-28-22-96-19-68-84-167	272,916.02
194	142-30-34-126-46-182-148-58-106-80-82-12-110-14	266,327.10
195	83-140-113-175-74-79-43-54-62	245,755.86
196	29-112-75-121-105-129-130-90-125-153-6	206,966.83
197	85-114-144-101-57	91,540.79

Tabel 5.13: Komparasi Hasil Pengujian Kondisi *Packet Loss* Dengan Data Lapangan

<i>Parameter</i>	CoEAs MDVRP	Pub/Sub CoEAS MDVRP
Total Waktu	3,373,065.19	3,868,083.19
Rata-rata	224,871.01	257,872.21
Standar Deviasi	88,167.80	56,301.47



Gambar 5.12: Rute Hasil Pengujian Kondisi *Packet Loss* Data Lapangan Pada MDVRP berbasis CoEAs



Gambar 5.13: Rute Hasil Pengujian Kondisi *Packet Loss* Data Lapangan Pada MDVRP berbasis CoEAs dengan Publish/Subscribe

BAB 6

KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan hasil pengujian, diperoleh kesimpulan sebagai berikut:

1. Pada permasalahan MDVRP yang tidak mempertimbangkan *service time*, diperoleh kesimpulan bahwasannya sistem usulan yang menggunakan algoritma MDVRP berbasis CoEAs yang dikombinasikan dengan mekanisme Publish/Subscribe menghasilkan total waktu 80 persen lebih buruk dari program pembandingan yang menggunakan algoritma MDVRP berbasis CoEAs tanpa mekanisme Publish/Subscribe. Sementara dari sisi *standar deviasi*, sistem usulan menghasilkan 80 persen lebih baik dari aplikasi pembandingan.
2. Pada permasalahan MDVRP yang mempertimbangkan *service time*, dengan menggunakan data Cordeau PO1 diperoleh hasil total waktu yang dihasilkan dari sistem usulan 6,25 persen lebih buruk dari aplikasi pembandingan, tetapi 68,83 persen lebih baik dari sisi standar deviasi. Sementara pada pengujian dengan menggunakan data lapangan, sistem usulan lebih buruk 8,25 persen dibandingkan program pembandingan, tetapi 59,11 persen dari sisi standar deviasi.
3. Pada permasalahan MDVRP yang mempertimbangkan *service time* dan terdapat *delay* dalam jaringan, pada pengujian dengan menggunakan data lapangan diperoleh hasil bahwasannya sistem usulan 4,83 persen lebih buruk dari sisi total waktu, tetapi 3,62 persen lebih baik dari sisi standar deviasi.
4. Pada permasalahan MDVRP yang mempertimbangkan *service time* dan terdapat *packet loss* dalam jaringan, pada pengujian dengan menggunakan data lapangan diperoleh hasil bahwasannya sistem usulan 12,79 persen lebih buruk dari sisi total waktu, tetapi 36,14 persen lebih baik dari sisi standar deviasi.

6.2 Saran

Dari penelitian ini, saran yang diberikan untuk penelitian selanjutnya sebagai berikut:

1. Mengkaji mekanisme komunikasi yang digunakan, misalnya dengan membandingkan dengan mekanisme Request/Reply dan Push/Pull untuk mendapatkan mekanisme komunikasi yang paling tepat.
2. Mengkaji parameter yang dapat digunakan dalam VRP Solver yaitu lama waktu eksekusi dan lama waktu dimana tidak ada perubahan solusi terbaik, sehingga diperoleh waktu eksekusi yang paling optimal.
3. Mengkaji penggunaan algoritma MDVRP yang lain, misalnya Tabu Search, Particle Swarm Optimization, dan lain lain, sehingga diperoleh algoritma MDVRP yang paling sesuai.

DAFTAR PUSTAKA

- Anders Kofod-Petersen and Agnar Aamodt. Case-based situation assessment in a mobile context-aware system. In *Proceedings of AIMS2003, Workshop on Artificial Intelligence for Mobile Systems, Seattle*. Citeseer, 2003. URL <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.440.7355&rep=rep1&type=pdf>.
- Thomas Weise, Alexander Podlich, and Christian Gorltd. Solving real-world vehicle routing problems with evolutionary algorithms. In *Natural intelligence for scheduling, planning and packing problems*, pages 29–53. Springer, 2009. URL http://link.springer.com/chapter/10.1007/978-3-642-04039-9_2.
- Fernando Bernardes de Oliveira, Rasul Enayatifar, Hossein Javedani Sadaei, Frederico Gadelha Guimares, and Jean-Yves Potvin. A cooperative coevolutionary algorithm for the Multi-Depot Vehicle Routing Problem. *Expert Systems with Applications*, 43:117–130, 2016. URL <http://www.sciencedirect.com/science/article/pii/S0957417415005771>.
- Patrick Th Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM computing surveys (CSUR)*, 35(2):114–131, 2003. URL <http://dl.acm.org/citation.cfm?id=857078>.
- BPS BPS. Badan Pusat Statistik - Tugas, Fungsi, dan Kewenangan, 2016a. URL <https://www.bps.go.id/index.php/masterMenu/view/id/1#masterMenuTab4>.
- BPS BPS. Sistem Rujukan Statistik BPS RI - Blok Sensus, 2016b. URL <https://sirusa.bps.go.id/index.php?r=istilah/view&id=43>.
- Seymour Sudman. Time Allocation in Survey Interviewing and in Other Field Occupations. *The Public Opinion Quarterly*, 29(4):638–648, 1965. ISSN 0033-362X. URL <http://www.jstor.org.proxy.lib.odu.edu/stable/2747041>.
- Guanling Chen, David Kotz, and others. A survey of context-aware mobile computing research. Technical report, Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, 2000. URL https://mmlab.snu.ac.kr/courses/2005_advanced_internet/handout/ppt/36%20-%20context_aware.pdf.

- Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, pages 85–90. IEEE, 1994. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4624429.
- Albrecht Schmidt, Michael Beigl, and Hans-W. Gellersen. There is more to context than location. *Computers & Graphics*, 23(6):893–901, 1999. URL <http://www.sciencedirect.com/science/article/pii/S009784939900120X>.
- Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *International Symposium on Handheld and Ubiquitous Computing*, pages 304–307. Springer, 1999. URL http://link.springer.com/chapter/10.1007/3-540-48157-5_29.
- M. J. Tsai, C. L. Wu, S. K. Pradhan, Y. Xie, T. Y. Li, L. C. Fu, and Y. C. Zeng. Context-aware activity prediction using human behavior pattern in real smart home environments. In *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 168–173, August 2016. doi: 10.1109/COASE.2016.7743376.
- M. B. Magara, S. Ojo, S. Ngwira, and T. Zuva. MPlist: Context aware music playlist. In *2016 IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies (EmergiTech)*, pages 309–316, August 2016. doi: 10.1109/EmergiTech.2016.7737358.
- Alan Said, Shlomo Berkovsky, and Ernesto W. De Luca. Introduction to Special Section on CAMRa2010: Movie Recommendation in Context. *ACM Trans. Intell. Syst. Technol.*, 4(1):13:1–13:9, February 2013. ISSN 2157-6904. doi: 10.1145/2414425.2414438. URL <http://doi.acm.org/10.1145/2414425.2414438>.
- Paul Y. Cao, Gang Li, Guoxing Chen, and Biao Chen. Mobile Data Collection Frameworks: A Survey. In *Proceedings of the 2015 Workshop on Mobile Big Data, Mobidata '15*, pages 25–30, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3524-9. doi: 10.1145/2757384.2757396. URL <http://doi.acm.org/10.1145/2757384.2757396>.
- Trinh Minh Tri Do and Daniel Gatica-Perez. Groupus: Smartphone proximity data and human interaction type mining. In *2011 15th Annual International Symposium on Wearable Computers*, pages 21–28. IEEE, 2011. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5959586.

- Jiangpeng Dai, Jin Teng, Xiaole Bai, Zhaohui Shen, and Dong Xuan. Mobile phone based drunk driving detection. In *2010 4th International Conference on Pervasive Computing Technologies for Healthcare*, pages 1–8. IEEE, 2010a. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5482295.
- Xueyang Zou, M. Gonzales, and S. Saeedi. A Context-aware Recommendation System using smartphone sensors. In *2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 1–6, October 2016. doi: 10.1109/IEMCON.2016.7746307.
- Jiangpeng Dai, Xiaole Bai, Zhimin Yang, Zhaohui Shen, and Dong Xuan. Per-FallD: A pervasive fall detection system using mobile phones. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, pages 292–297. IEEE, 2010b. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5470652.
- Hong Lu, Wei Pan, Nicholas D. Lane, Tanzeem Choudhury, and Andrew T. Campbell. SoundSense: scalable sound sensing for people-centric applications on mobile phones. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 165–178. ACM, 2009. URL <http://dl.acm.org/citation.cfm?id=1555834>.
- Xuan Bao and Romit Roy Choudhury. Movi: mobile phone based video highlights via collaborative sensing. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 357–370. ACM, 2010. URL <http://dl.acm.org/citation.cfm?id=1814468>.
- Paul Rubel, Jocelyne Fayn, Giandomenico Nollo, Deodato Assanelli, Bo Li, Lioara Restier, Stefano Adami, Sebastien Arod, Hussein Atoui, Mattias Ohlsson, and others. Toward personal eHealth in cardiology. Results from the EPI-MEDICS telemedicine project. *Journal of electrocardiology*, 38(4):100–106, 2005. URL <http://www.sciencedirect.com/science/article/pii/S0022073605001214>.
- Martin Atzmueller and Katy Hilgenberg. Towards Capturing Social Interactions with SDCF: An Extensible Framework for Mobile Sensing and Ubiquitous Data Collection. In *Proceedings of the 4th International Workshop on Modeling Social Media, MSM '13*, pages 6:1–6:4, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2007-8. doi: 10.1145/2463656.2463662. URL <http://doi.acm.org/10.1145/2463656.2463662>.

- Z. Yuan and H. Li. Location recommendation algorithm based on temporal and geographical similarity in location-based social networks. In *2016 12th World Congress on Intelligent Control and Automation (WCICA)*, pages 1697–1702, June 2016. doi: 10.1109/WCICA.2016.7578804.
- Yehuda Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010. URL <http://dl.acm.org/citation.cfm?id=1721677>.
- H. Pragarauskas and Oliver Gross. Temporal collaborative filtering with bayesian probabilistic tensor factorization. 2010. URL <http://epubs.siam.org/doi/abs/10.1137/1.9781611972801.19>.
- Norman Biggs, E. Keith Lloyd, and Robin J. Wilson. *Graph Theory, 1736-1936*. Oxford University Press, 1976. URL https://www.google.com/books?hl=en&lr=&id=XqYTk0sXmpoC&oi=fnd&pg=PR9&dq=Graph+Theory+1736-1936+biggs&ots=BJk_eD4NGR&sig=x1B-VZ38Zw0gfJ-EkJXwfKOL0r4.
- Tolga Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209–219, June 2006. ISSN 0305-0483. doi: 10.1016/j.omega.2004.10.004. URL <http://www.sciencedirect.com/science/article/pii/S0305048304001550>.
- Samuel Gorenstein. Printing press scheduling for multi-edition periodicals. *Management Science*, 16(6):B–373, 1970. URL <http://pubsonline.informs.org/doi/abs/10.1287/mnsc.16.6.B373>.
- Joseph A. Svestka and Vaughn E. Huckfeldt. Computational experience with an m-salesman traveling salesman algorithm. *Management Science*, 19(7):790–799, 1973. URL <http://pubsonline.informs.org/doi/abs/10.1287/mnsc.19.7.790>.
- R. D. Angel, W. L. Caudle, R. Noonan, and ANDA Whinston. Computer-assisted school bus scheduling. *Management Science*, 18(6):B–279, 1972. URL <http://pubsonline.informs.org/doi/abs/10.1287/mnsc.18.6.b279>.
- Kenneth C. Gilbert and Ruth B. Hofstra. A new multiperiod multiple traveling salesman problem with heuristic and application to a scheduling problem. *Decision Sciences*, 23(1):250–259, 1992. URL <http://onlinelibrary.wiley.com/doi/10.1111/j.1540-5915.1992.tb00387.x/full>.
- Barry L. Brumitt and Anthony Stentz. Dynamic mission planning for multiple mobile robots. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE*

- International Conference on*, volume 3, pages 2396–2401. IEEE, 1996. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=506522.
- Lixin Tang, Jiyin Liu, Aiyong Rong, and Zihou Yang. A multiple traveling salesman problem model for hot rolling scheduling in Shanghai Baoshan Iron & Steel Complex. *European Journal of Operational Research*, 124(2):267–282, 2000. URL <http://www.sciencedirect.com/science/article/pii/S037722179900380X>.
- Hussain Aziz Saleh and Rachid Chelouah. The design of the global navigation satellite system surveying networks using genetic algorithms. *Engineering Applications of Artificial Intelligence*, 17(1):111–122, 2004. URL <http://www.sciencedirect.com/science/article/pii/S0952197603001350>.
- Françoise Fabret, H. Arno Jacobsen, François Llirbat, Joo Pereira, Kenneth A. Ross, and Dennis Shasha. Filtering Algorithms and Implementation for Very Fast Publish/Subscribe Systems. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, SIGMOD '01, pages 115–126, New York, NY, USA, 2001. ACM. ISBN 978-1-58113-332-5. doi: 10.1145/375663.375677. URL <http://doi.acm.org/10.1145/375663.375677>.
- Xiaoyan Chen, Ying Chen, and Fangyan Rao. An efficient spatial publish/subscribe system for intelligent location-based services. In *Proceedings of the 2nd international workshop on Distributed event-based systems*, pages 1–6. ACM, 2003a. URL <http://dl.acm.org/citation.cfm?id=966625>.
- John W. Creswell. *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications, 2013. URL https://www.google.com/books?hl=en&lr=&id=EbogAQAAQBAJ&oi=fnd&pg=PP1&dq=J.+W.+Creswell,+Research+Design+Qualitative,+Quantitative,+and+Mixed+Methods+Approaches,+3rd+ed.+SAGE+Publications,+Inc,+2008&ots=caiStUSxA3&sig=Nb8n_G4Z_8Fr-UH56kQsUZQQYU8.
- Andreas Zimmermann, Andreas Lorenz, and Reinhard Oppermann. An operational definition of context. In *International and Interdisciplinary Conference on Modeling and Using Context*, pages 558–571. Springer, 2007. URL http://link.springer.com/chapter/10.1007/978-3-540-74255-5_42.
- R. Hull, P. Neaves, and J. Bedford-Roberts. Towards situated computing. In *Digest of Papers. First International Symposium on Wearable Computers*, pages 146–153, October 1997. doi: 10.1109/ISWC.1997.629931.

- Peter J. Brown. The stick-e document: a framework for creating context-aware applications. *Electronic Publishing-Chichester*-, 8:259–272, 1995. URL <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.8.7472&rep=rep1&type=pdf>.
- Peter J. Brown, John D. Bovey, and Xian Chen. Context-aware applications: from the laboratory to the marketplace. *IEEE personal communications*, 4(5):58–64, 1997. URL <http://ieeexplore.ieee.org/abstract/document/626984/>.
- Tom Gross and Marcus Specht. Awareness in context-aware information systems. In *Mensch & Computer 2001*, pages 173–182. Springer, 2001. URL http://link.springer.com/chapter/10.1007/978-3-322-80108-1_19.
- Nick Ryan, Jason Pascoe, and David Morse. Enhanced reality fieldwork: the context aware archaeological assistant. *Bar International Series*, 750:269–274, 1999. URL https://bibliographie.uni-tuebingen.de/xmlui/bitstream/handle/10900/62097/44_Ryan_et_al_CAA_1997.pdf?sequence=2&isAllowed=y.
- Harry Chen, T. Finin, and A. Joshi. An intelligent broker architecture for context-aware systems. *PhD proposal in computer science, University of Maryland, Baltimore, USA*, 2003b. URL <http://aisl.umbc.edu/get/softcopy/id/789/789.pdf>.
- Anind K. Dey. Understanding and using context. *Personal and ubiquitous computing*, 5(1):4–7, 2001. URL <http://dl.acm.org/citation.cfm?id=593572>.
- Gbor Nagy and Sad Salhi. Location-routing: Issues, models and methods. *European Journal of Operational Research*, 177(2):649–672, 2007. URL <http://www.sciencedirect.com/science/article/pii/S0377221706002670>.
- A.D. Bruns. *Zweistufige Standortplanung unter Berücksichtigung von Tourenplanungsaspekten Primale Heuristiken und Lokale Suchverfahren*. PhD Thesis, Sankt Gallen University, 1998.
- Anantaram Balakrishnan, James E. Ward, and Richard T. Wong. Integrated Facility Location and Vehicle Routing Models: Recent Work and Future Prospects. *American Journal of Mathematical and Management Sciences*, 7(1-2):35–61, January 1987. ISSN 0196-6324. doi: 10.1080/01966324.1987.10737207. URL <http://dx.doi.org/10.1080/01966324.1987.10737207>.
- F. E. Maranzana. On the Location of Supply Points to Minimize Transport Costs. *OR*, 15(3):261–270, 1964. ISSN 1473-2858. doi: 10.2307/3007214. URL <http://www.jstor.org.proxy.lib.odu.edu/stable/3007214>.

- Igor Averbakh and Oded Berman. Technical Note Routing and Location-Routing p-Delivery Men Problems on a Path. *Transportation Science*, 28(2):162–166, 1994. URL <http://pubsonline.informs.org/doi/abs/10.1287/trsc.28.2.162>.
- Igor Averbakh and Oded Berman. Probabilistic Sales-Delivery Man and Sales-Delivery Facility Location Problems on a Tree. *Transportation Science*, 29(2): 184–197, May 1995. ISSN 0041-1655. doi: 10.1287/trsc.29.2.184. URL <http://pubsonline.informs.org.proxy.lib.odu.edu/doi/abs/10.1287/trsc.29.2.184>.
- David Simchi-Levi. The Capacitated Traveling Salesman Location Problem. *Transportation Science*, 25(1):9–18, February 1991. ISSN 0041-1655. doi: 10.1287/trsc.25.1.9. URL <http://pubsonline.informs.org.proxy.lib.odu.edu/doi/abs/10.1287/trsc.25.1.9>.
- Gilbert Laporte and Yves Nobert. An exact algorithm for minimizing routing and operating costs in depot location. *European Journal of Operational Research*, 6(2):224–226, February 1981. ISSN 0377-2217. doi: 10.1016/0377-2217(81)90212-5. URL <http://www.sciencedirect.com/science/article/pii/0377221781902125>.
- James H. Bookbinder and Kathleen E. Reece. Vehicle routing considerations in distribution system design. *European Journal of Operational Research*, 37(2):204–213, 1988. URL <http://www.sciencedirect.com/science/article/pii/037722178890330X>.
- Said Salhi and M. Fraser. An Intergrated Heuristic Approach for the Combined Location Vehicle Fleet Mix Problem. *Studies in Locational Analysis*, 8:3–22, March 1996. URL <http://kar.kent.ac.uk/5232/>.
- I. M. Branco and J. D. Coelho. The hamiltonian p-median problem. *European Journal of Operational Research*, 47(1):86–95, July 1990. ISSN 0377-2217. doi: 10.1016/0377-2217(90)90092-P. URL <http://www.sciencedirect.com/science/article/pii/037722179090092P>.
- G. B. Dantzig and J. H. Ramser. The Truck Dispatching Problem. *Management Science*, 6(1):80–91, October 1959. ISSN 0025-1909. doi: 10.1287/mnsc.6.1.80. URL <http://pubsonline.informs.org.proxy.lib.odu.edu/doi/abs/10.1287/mnsc.6.1.80>.
- Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European journal of operational research*, 59

- (3):345–358, 1992. URL <http://www.sciencedirect.com/science/article/pii/037722179290192C>.
- Michael R. Garey and David S. Johnson. *Computers and intractability*, volume 29. wh freeman New York, 2002. URL <https://pdfs.semanticscholar.org/fa33/c3a89a423f0eb00134fd291894949bf9d54e.pdf>.
- Roberto Baldacci, Paolo Toth, and Daniele Vigo. Exact algorithms for routing problems under vehicle capacity constraints. *Annals of Operations Research*, 175(1):213–245, March 2010. ISSN 0254-5330, 1572-9338. doi: 10.1007/s10479-009-0650-0. URL <http://link.springer.com.proxy.lib.odu.edu/article/10.1007/s10479-009-0650-0>.
- Jean-Francois Cordeau, Gilbert Laporte, Martin W. P. Savelsbergh, and Daniele Vigo. Chapter 6 Vehicle Routing. In Cynthia Barnhart and Gilbert Laporte, editor, *Handbooks in Operations Research and Management Science*, volume 14 of *Transportation*, pages 367–428. Elsevier, 2007. URL <http://www.sciencedirect.com/science/article/pii/S0927050706140062>. DOI: 10.1016/S0927-0507(06)14006-2.
- Paolo Toth and Daniele Vigo. The vehicle routing problem, volume 9 of SIAM Monographs on Discrete Mathematics and Applications. *SIAM, Philadelphia, PA*, 2002.
- Marius M. Solomon and Jacques Desrosiers. Survey papertime window constrained routing and scheduling problems. *Transportation science*, 22(1):1–13, 1988. URL <http://pubsonline.informs.org/doi/abs/10.1287/trsc.22.1.1>.
- Harilaos N. Psaraftis. Dynamic vehicle routing: Status and prospects. *Annals of operations research*, 61(1):143–164, 1995. URL <http://link.springer.com/article/10.1007/BF02098286>.
- Jacques Renaud, Gilbert Laporte, and Faye F. Boctor. A tabu search heuristic for the multi-depot vehicle routing problem. *Computers & Operations Research*, 23(3):229–235, March 1996a. ISSN 0305-0548. doi: 10.1016/0305-0548(95)00026-P. URL <http://www.sciencedirect.com/science/article/pii/030505489500026P>.
- R. V. Kulkarni and P. R. Bhawe. Integer programming formulations of vehicle routing problems. *European Journal of Operational Research*, 20(1):58–67, April 1985. ISSN 0377-2217. doi: 10.1016/0377-2217(85)90284-X. URL <http://www.sciencedirect.com/science/article/pii/037722178590284X>.

- Gilbert Laporte, Yves Nobert, and Danielle Arpin. *Optimal solutions to capacitated multidepot vehicle routing problems*. Universit de Montral, Centre de recherche sur les transports, 1984.
- Gilbert Laporte, Yves Nobert, and Serge Taillefer. Solving a Family of Multi-Depot Vehicle Routing and Location-Routing Problems. *Transportation Science*, 22(3): 161–172, August 1988. ISSN 0041-1655. doi: 10.1287/trsc.22.3.161. URL <http://pubsonline.informs.org.proxy.lib.odu.edu/doi/abs/10.1287/trsc.22.3.161>.
- Frank A. Tillman. The multiple terminal delivery problem with probabilistic demands. *Transportation Science*, 3(3):192–204, 1969. URL <http://pubsonline.informs.org/doi/abs/10.1287/trsc.3.3.192>.
- Frank A. Tillman and Thomas M. Cain. An upperbound algorithm for the single and multiple terminal delivery problem. *Management Science*, 18(11):664–682, 1972. URL <http://pubsonline.informs.org/doi/abs/10.1287/mnsc.18.11.664>.
- Frank A. Tillman and Robert W. Hering. A study of a look-ahead procedure for solving the multiterminal delivery problem. *Transportation Research*, 5(3):225–229, 1971. URL <http://www.sciencedirect.com/science/article/pii/0041164771900232>.
- Anthony Wren and Alan Holliday. Computer scheduling of vehicles from one or more depots to a number of delivery points. *Journal of the Operational Research Society*, 23(3):333–344, 1972. URL <http://link.springer.com/article/10.1057/jors.1972.53>.
- Billy E. Gillett and Jerry G. Johnson. Multi-terminal vehicle-dispatch algorithm. *Omega*, 4(6):711–718, 1976. URL <http://www.sciencedirect.com/science/article/pii/0305048376900979>.
- Bruce L. Golden, Thomas L. Magnanti, and Hien Q. Nguyen. Implementing vehicle routing algorithms. *Networks*, 7(2):113–148, 1977. URL <http://onlinelibrary.wiley.com/doi/10.1002/net.3230070203/full>.
- Ole M. Raft. A modular algorithm for an extended vehicle scheduling problem. *European Journal of Operational Research*, 11(1):67–76, 1982. URL <http://www.sciencedirect.com/science/article/pii/S0377221782800118>.
- I.-Ming Chao, Bruce L. Golden, and Edward Wasil. A New Heuristic for the Multi-Depot Vehicle Routing Problem that Improves upon Best-Known Solutions. *American Journal of Mathematical and Management Sciences*, 13(3-4):371–406,

- February 1993. ISSN 0196-6324. doi: 10.1080/01966324.1993.10737363. URL <http://dx.doi.org/10.1080/01966324.1993.10737363>.
- Gunter Dueck. New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational physics*, 104(1):86–92, 1993. URL <http://www.sciencedirect.com/science/article/pii/S0021999183710107>.
- S. Lin. Computer solutions of the traveling salesman problem. *The Bell System Technical Journal*, 44(10):2245–2269, December 1965. ISSN 0005-8580. doi: 10.1002/j.1538-7305.1965.tb04146.x.
- Jacques Renaud, F. Bector, and Gilbert Laporte. An improved petal heuristic for the vehicle routing problem. *Journal of the Operational Research Society*, 47(2):329–336, 1996b. URL <http://link.springer.com/article/10.1057/jors.1996.29>.
- Jean-François Cordeau, Michel Gendreau, and Gilbert Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2):105–119, September 1997. ISSN 1097-0037. doi: 10.1002/(SICI)1097-0037(199709)30:2<105::AID-NET5>3.0.CO;2-G. URL [http://onlinelibrary.wiley.com.proxy.lib.odu.edu/doi/10.1002/\(SICI\)1097-0037\(199709\)30:2<105::AID-NET5>3.0.CO;2-G/abstract](http://onlinelibrary.wiley.com.proxy.lib.odu.edu/doi/10.1002/(SICI)1097-0037(199709)30:2<105::AID-NET5>3.0.CO;2-G/abstract).
- Michel Gendreau, Alain Hertz, and Gilbert Laporte. New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 40(6):1086–1094, 1992. URL <http://pubsonline.informs.org/doi/abs/10.1287/opre.40.6.1086>.
- David Pisinger and Stefan Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435, August 2007. ISSN 0305-0548. doi: 10.1016/j.cor.2005.09.012. URL <https://www.sciencedirect.com/science/article/pii/S0305054805003023>.
- H. C. W. Lau, T. M. Chan, W. T. Tsui, and W. K. Pang. Application of Genetic Algorithms to Solve the Multidepot Vehicle Routing Problem. *IEEE Transactions on Automation Science and Engineering*, 7(2):383–392, April 2010. ISSN 1545-5955. doi: 10.1109/TASE.2009.2019265.
- Jean-François Cordeau and Mirko Maischberger. A parallel iterated tabu search heuristic for vehicle routing problems. *Computers & Operations Research*, 39(9):2033–2050, September 2012. ISSN 0305-0548. doi: 10.1016/j.cor.2011.09.021. URL <https://www.sciencedirect.com/science/article/pii/S0305054811002826>.

- Anand Subramanian, Eduardo Uchoa, and Luiz Satoru Ochi. A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research*, 40(10): 2519–2531, October 2013. ISSN 0305-0548. doi: 10.1016/j.cor.2013.01.013. URL <https://www.sciencedirect.com/science/article/pii/S030505481300021X>.
- Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. Implicit depot assignments and rotations in vehicle routing heuristics. *European Journal of Operational Research*, 237(1):15–28, August 2014. ISSN 0377-2217. doi: 10.1016/j.ejor.2013.12.044. URL <https://www.sciencedirect.com/science/article/pii/S037722171301028X>.
- John Willmer Escobar, Rodrigo Linfati, Paolo Toth, and Maria G. Baldoquin. A hybrid Granular Tabu Search algorithm for the Multi-Depot Vehicle Routing Problem. *Journal of Heuristics*, 20(5):483–509, October 2014. ISSN 1381-1231, 1572-9397. doi: 10.1007/s10732-014-9247-0. URL <http://link.springer.com.proxy.lib.odu.edu/article/10.1007/s10732-014-9247-0>.
- Ingo Rechenberg. Cybernetic solution path of an experimental problem. 1965. URL <http://www.citeulike.org/group/1662/article/1505176>.
- Alois Huning, Ingo Rechenberg, and Manfred Eigen. *Evolutionsstrategie. Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. JSTOR, 1976. URL <http://www.jstor.org/stable/23679080>.
- H.-P. Schwefel. *Evolutionsstrategie und numerische Optimierung*. Technische Universität Berlin, 1975.
- John H. Holland and others. Echo: Explorations of evolution in a miniature world. 1990. URL <http://www.citeulike.org/group/468/article/674061>.
- Hiroaki Sengoku and Ikuo Yoshihara. A fast TSP solver using GA on JAVA. In *Third International Symposium on Artificial Life, and Robotics (AROB III98)*, pages 283–288, 1998. URL <http://prgmr.gcd.org/sengoku/docs/arob98.pdf>.
- Luis FG Sarmenta, Sandra JV Chua, Paul Echevarria, Jose M. Mendoza, Rene-Russelle Santos, Stanley Tan, and Richard Lozada. Bayesian Computing. NET: Grid Computing with XML Web Services. In *ccgrid*, pages 434–435, 2002. URL <http://ieeexplore.ieee.org/iel5/7937/21887/01017182.pdf>.
- Bernabe Dorronsoro Diaz. *Vrp web*. 2012.
- Gero Mhl. *Large-scale content-based publish-subscribe systems*. PhD thesis, TU Darmstadt, 2002. URL <http://tuprints.ulb.tu-darmstadt.de/274>.

Google Google. Google Maps Directions API, 2016. URL <https://developers.google.com/maps/documentation/directions/>.

jsprit. jsprit | java toolkit for rich VRPs and TSPs, 2014. URL <https://jsprit.github.io/>.

OptaPlanner. Constraint satisfaction solver (Java, Open Source), 2016. URL <https://www.optaplanner.org/>.

Guruduth Banavar, Tushar Chandra, Bodhi Mukherjee, Jay Nagarajarao, Robert E. Strom, and Daniel C. Sturman. An efficient multicast protocol for content-based publish-subscribe systems. In *Distributed Computing Systems, 1999. Proceedings. 19th IEEE International Conference on*, pages 262–272. IEEE, 1999. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=776528.

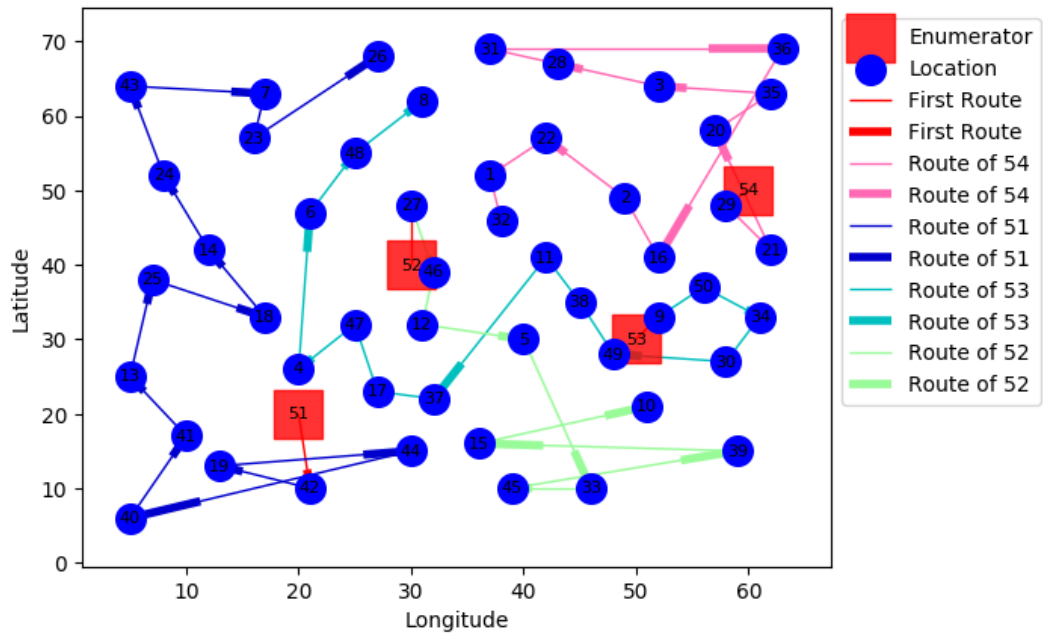
redis. Introduction to Redis Redis, 2017a. URL <https://redis.io/topics/introduction>.

redis. Clients - Redis, 2017b. URL <https://redis.io/clients>.

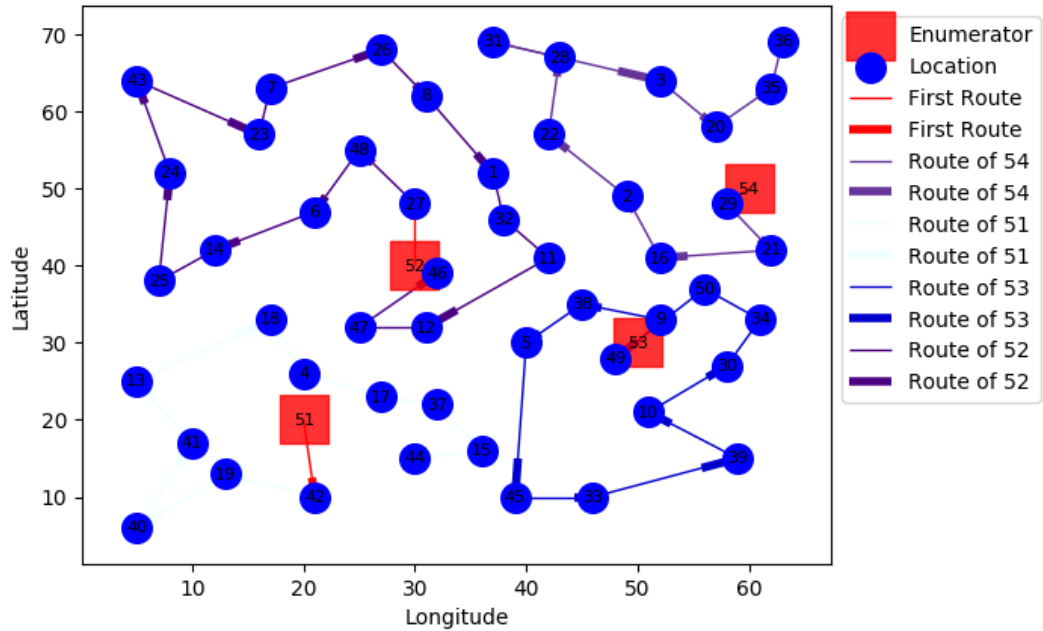
LAMPIRAN

LAMPIRAN 1

Hasil pengujian tanpa *Service Time* dengan menggunakan data Cordeau.

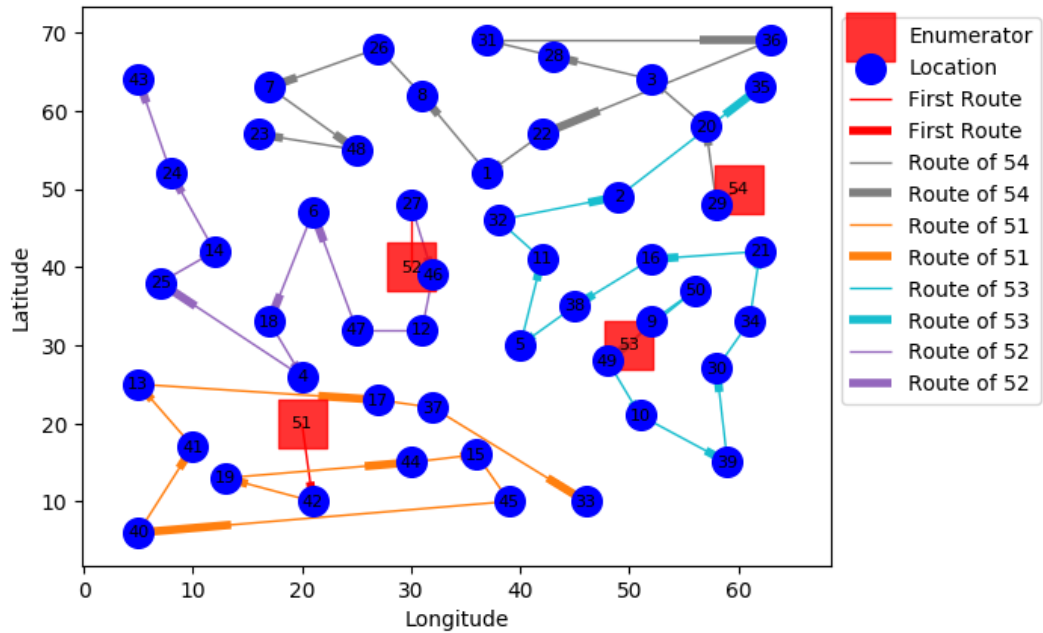


(a) MDVRP berbasis CoEAs

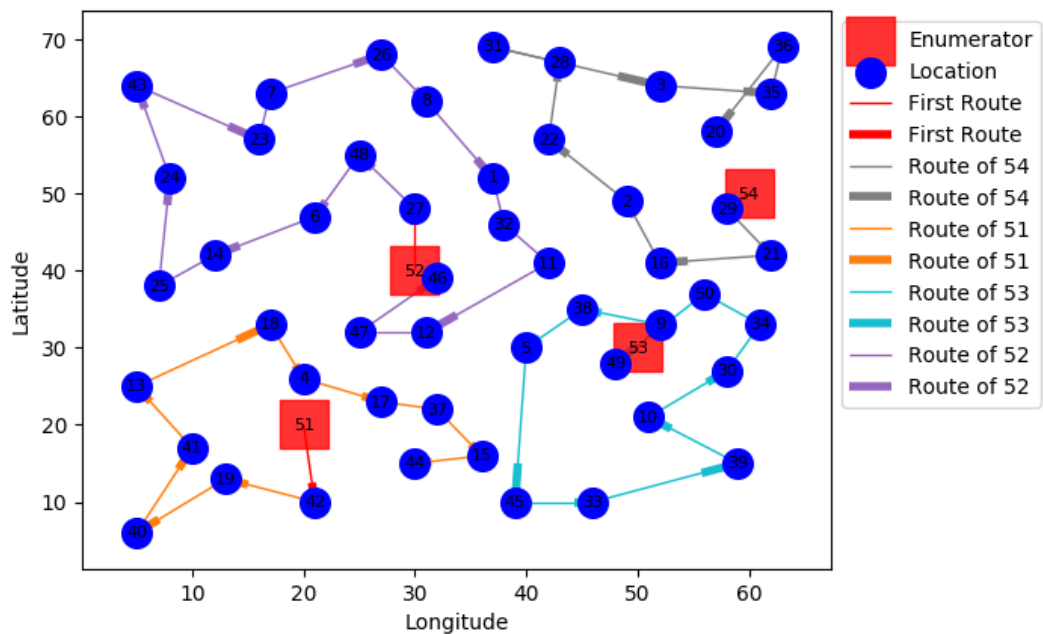


(b) MDVRP berbasis CoEAs dengan Publish/Subscribe

Gambar 1: Perbandingan Rute Hasil Pengujian Data Cordeau P01 Tanpa *Service Time*

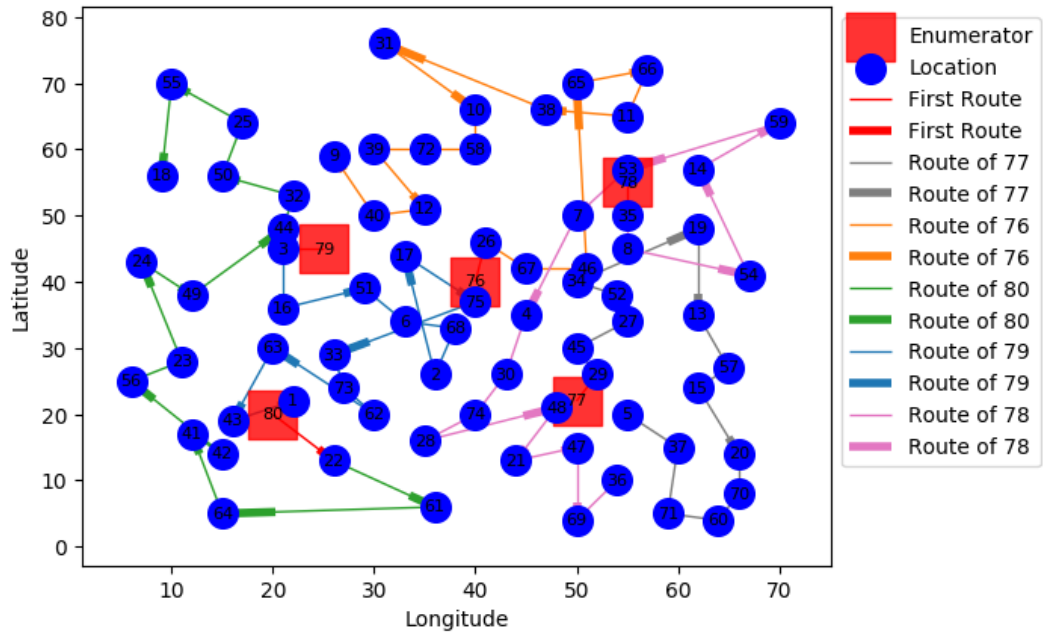


(a) MDVRP berbasis CoEAs

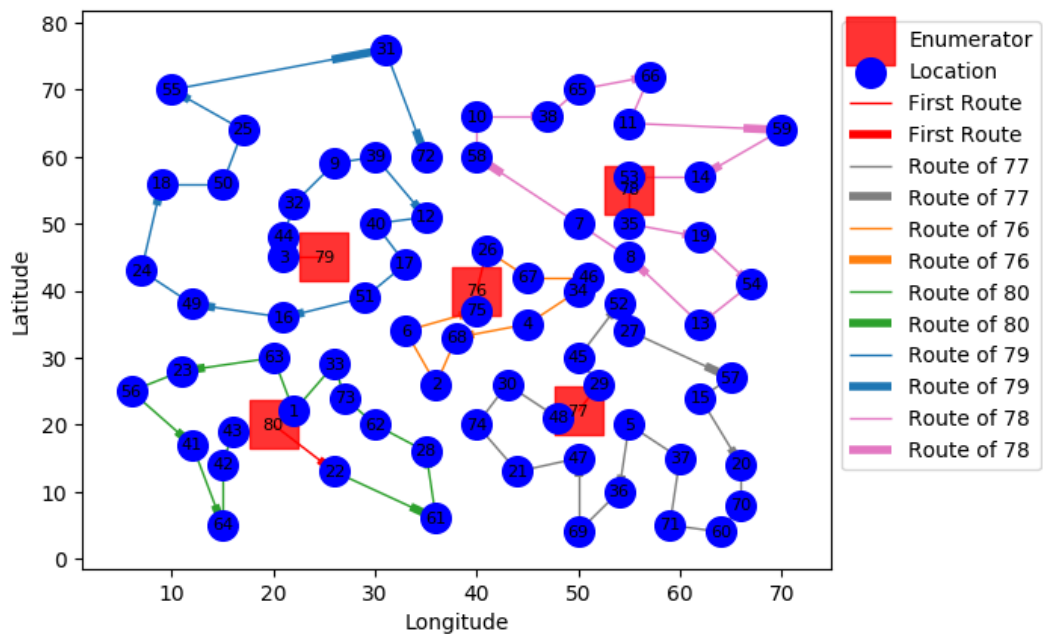


(b) MDVRP berbasis CoEAs dengan Publish/Subscribe

Gambar 2: Perbandingan Rute Hasil Pengujian Data Cordeau P02 Tanpa *Service Time*

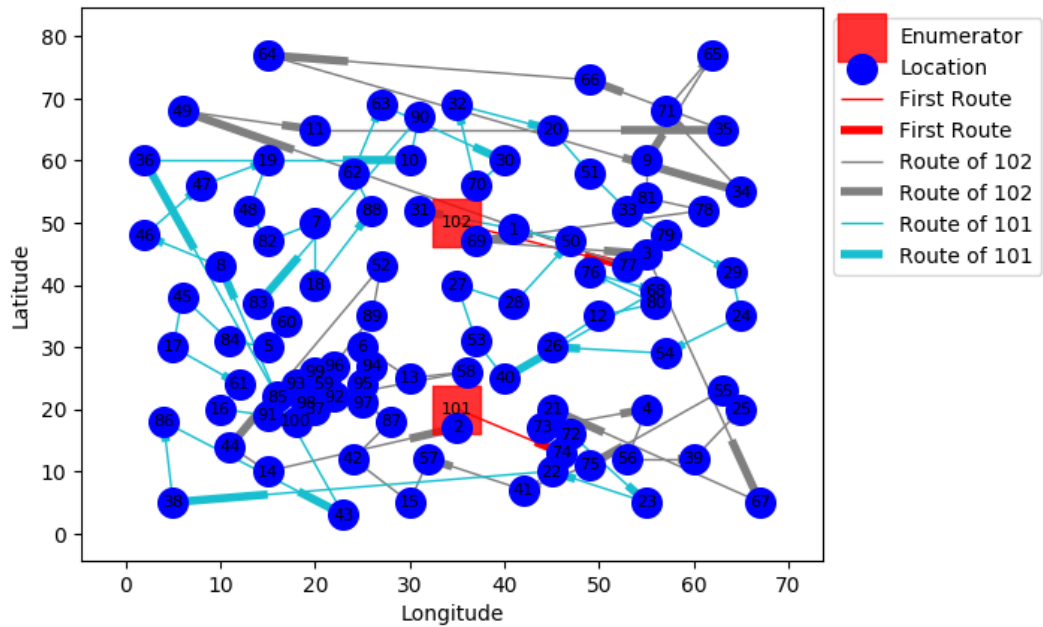


(a) MDVRP berbasis CoEAs

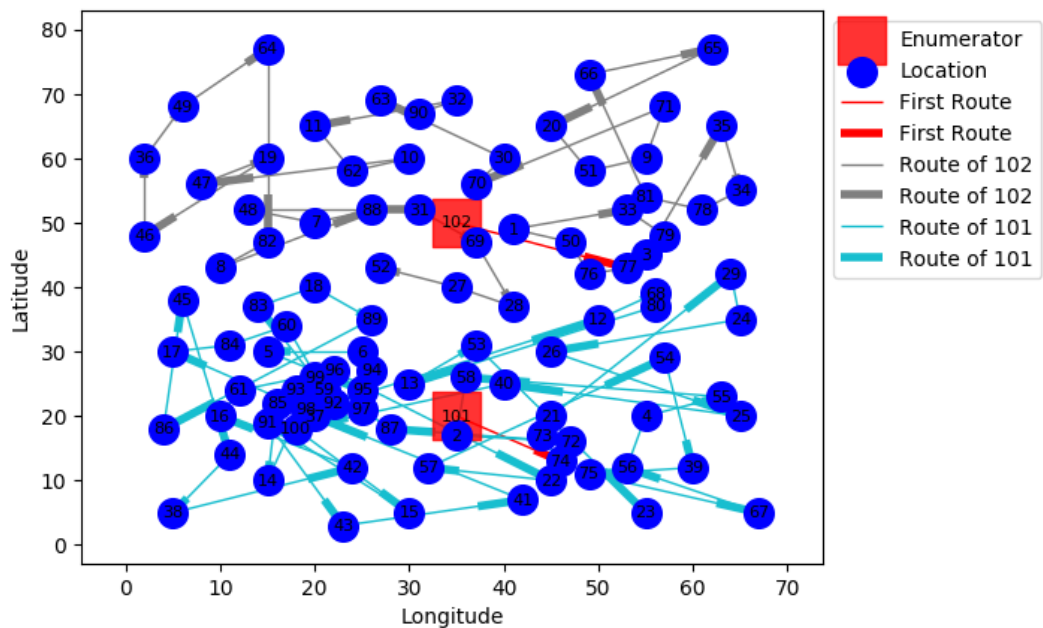


(b) MDVRP berbasis CoEAs dengan Publish/Subscribe

Gambar 3: Perbandingan Rute Hasil Pengujian Data Cordeau P03 Tanpa *Service Time*

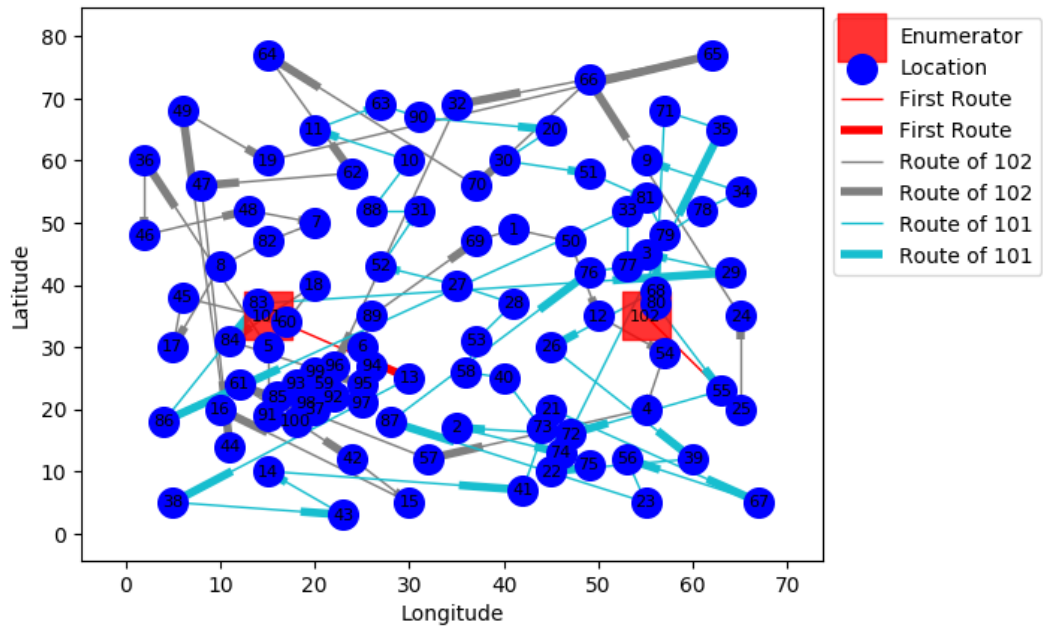


(a) MDVRP berbasis CoEAs

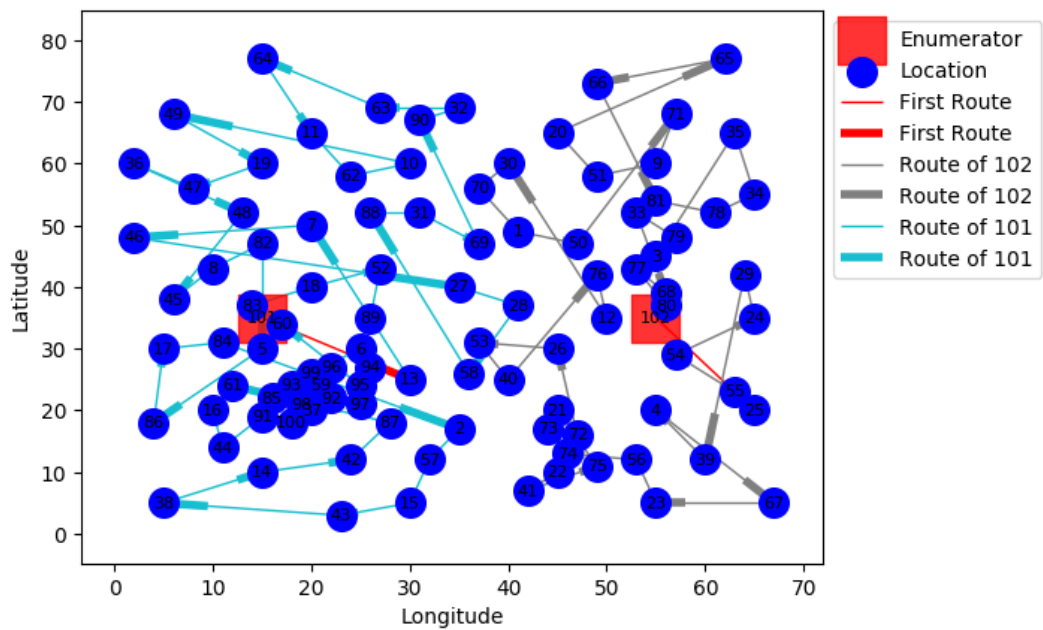


(b) MDVRP berbasis CoEAs dengan Publish/Subscribe

Gambar 4: Perbandingan Rute Hasil Pengujian Data Cordeau P04 Tanpa *Service Time*

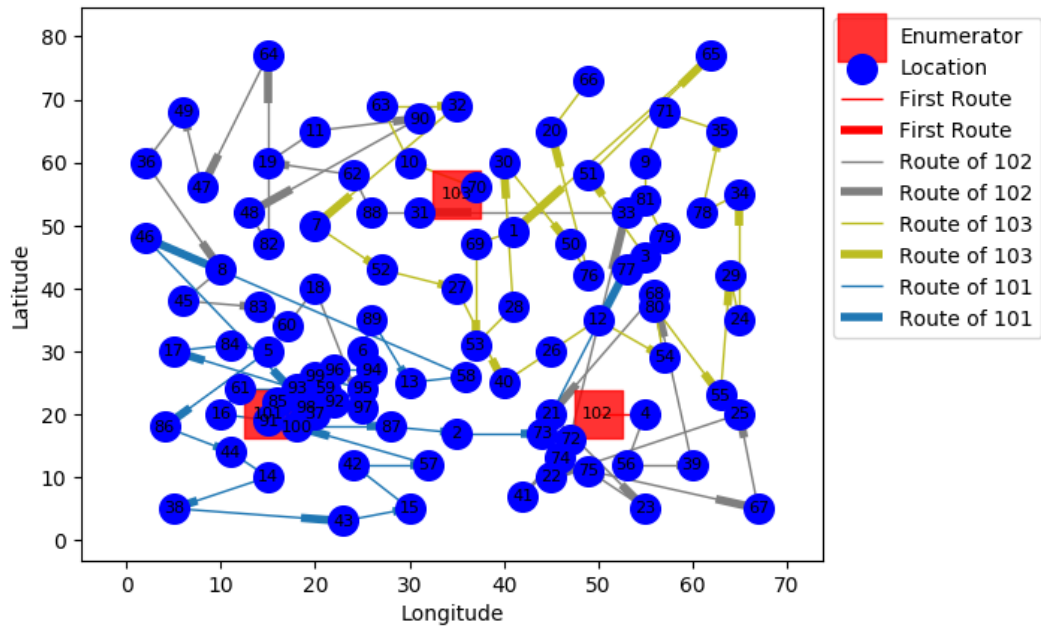


(a) MDVRP berbasis CoEAs

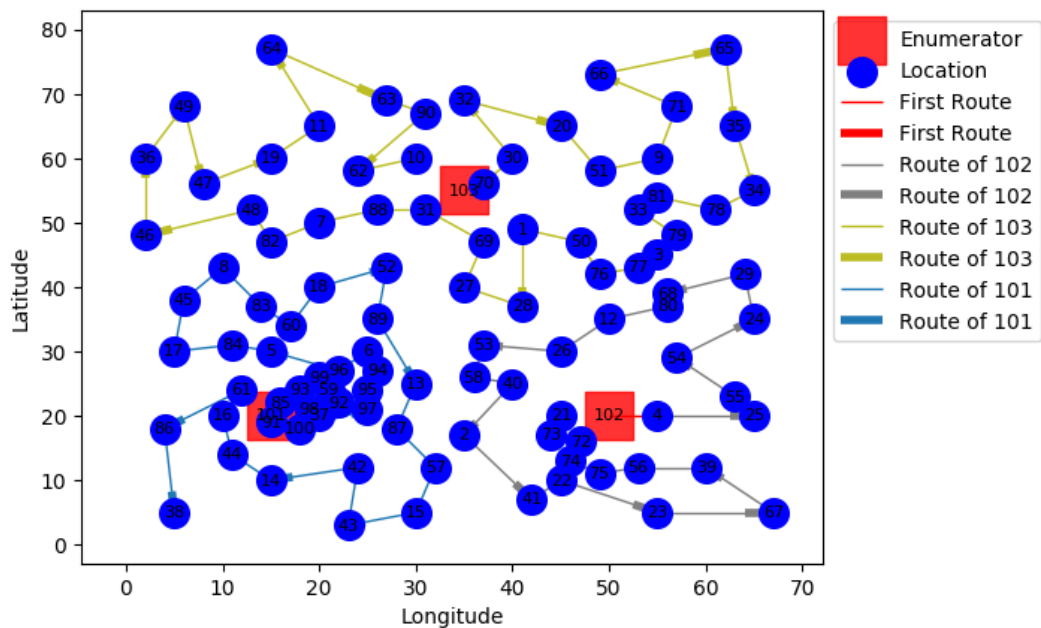


(b) MDVRP berbasis CoEAs dengan Publish/Subscribe

Gambar 5: Perbandingan Rute Hasil Pengujian Data Cordeau P05 Tanpa *Service Time*

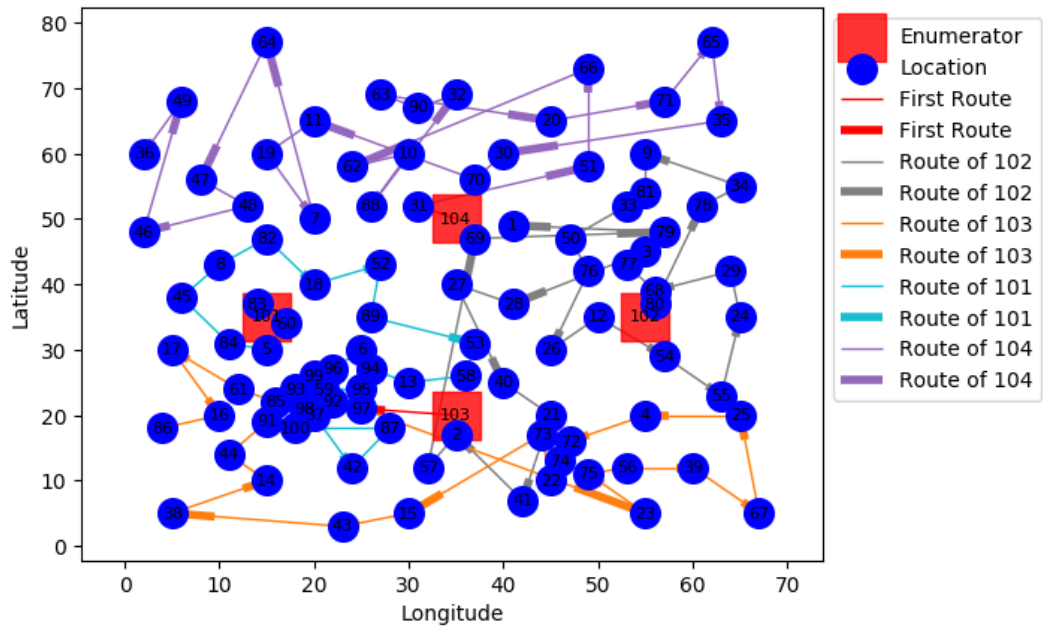


(a) MDVRP berbasis CoEAs

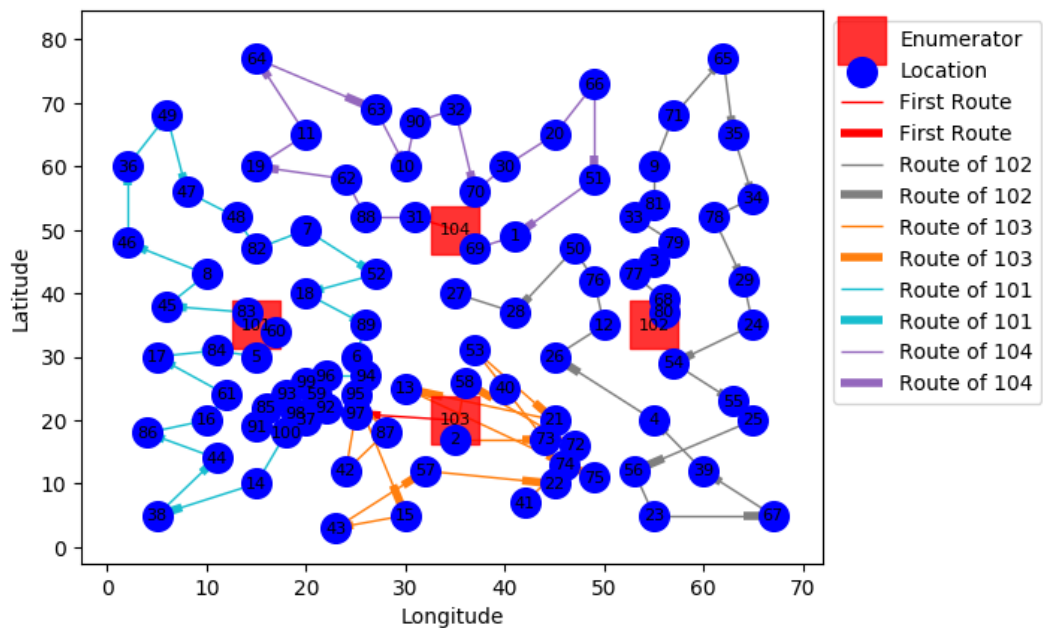


(b) MDVRP berbasis CoEAs dengan Publish/Subscribe

Gambar 6: Perbandingan Rute Hasil Pengujian Data Cordeau P06 Tanpa *Service Time*

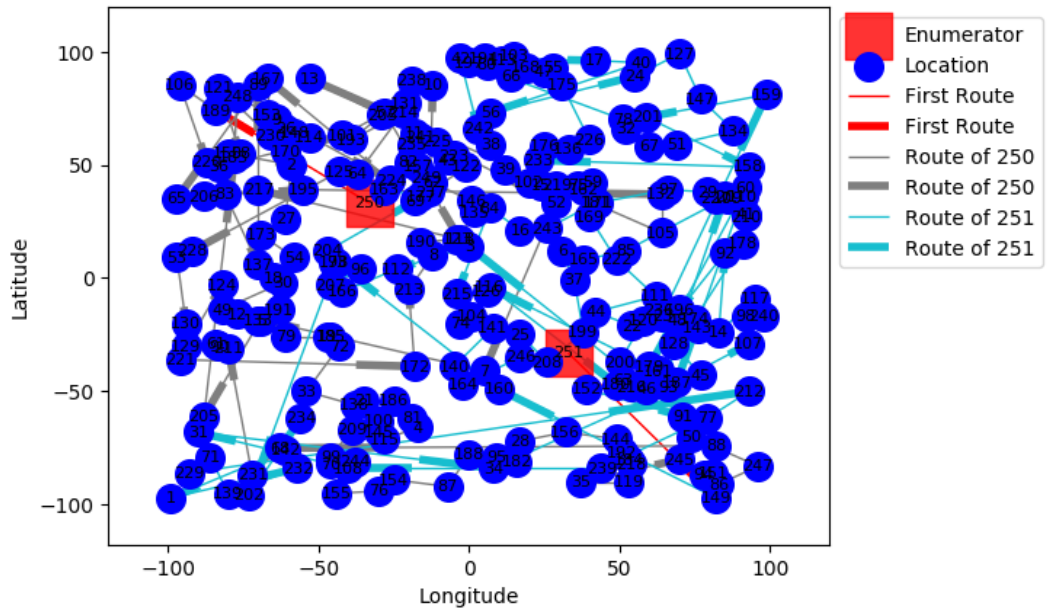


(a) MDVRP berbasis CoEAs

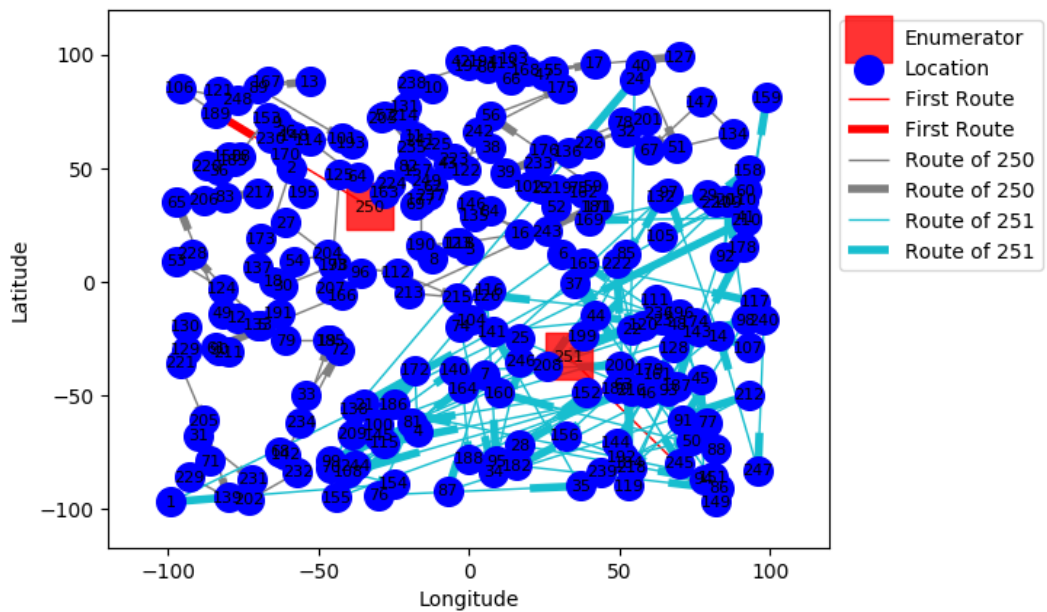


(b) MDVRP berbasis CoEAs dengan Publish/Subscribe

Gambar 7: Perbandingan Rute Hasil Pengujian Data Cordeau P07 Tanpa *Service Time*

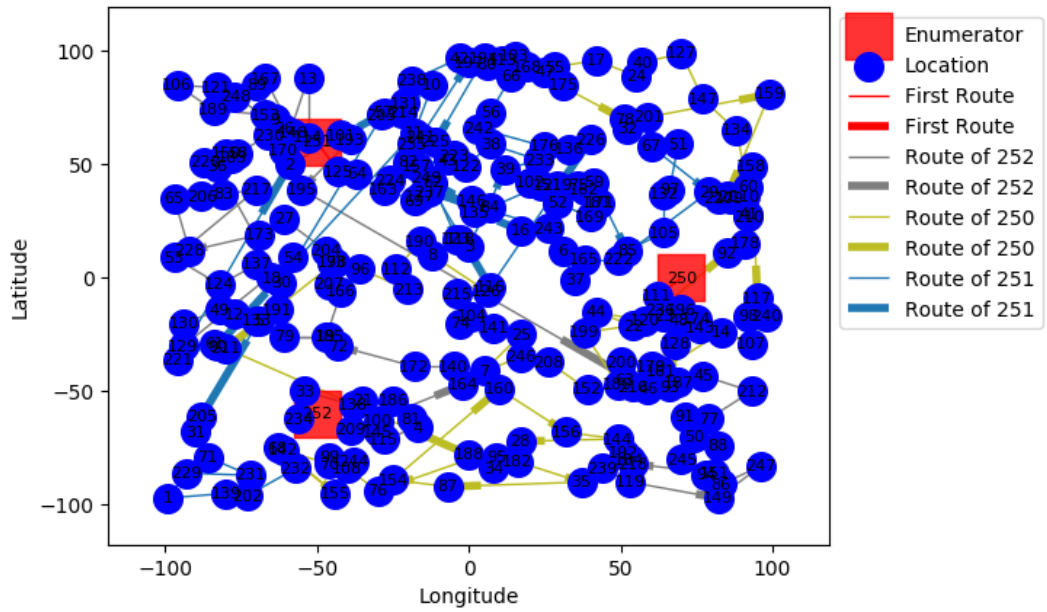


(a) MDVRP berbasis CoEAs

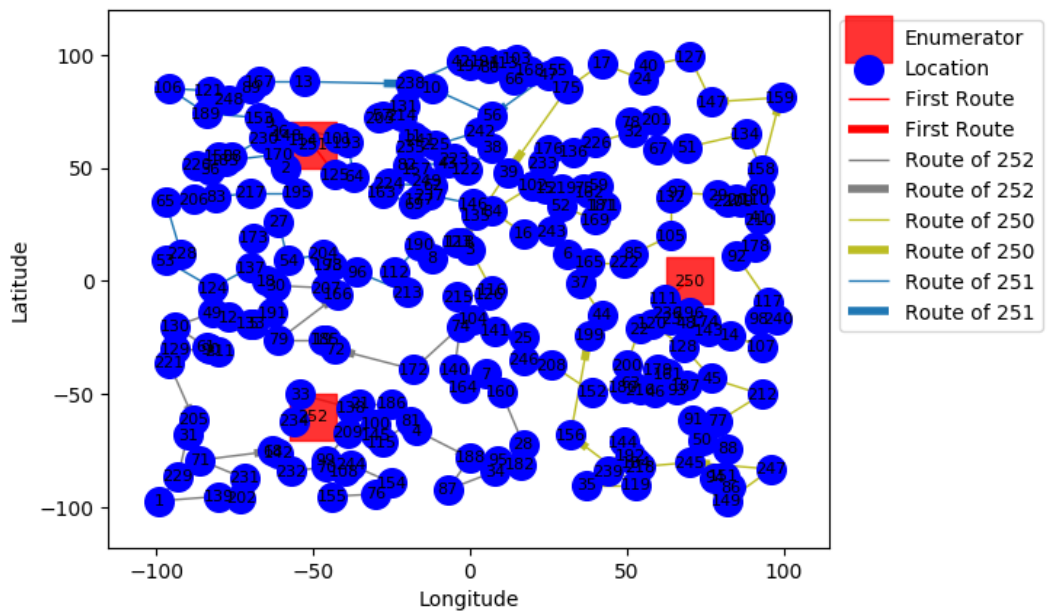


(b) MDVRP berbasis CoEAs dengan Publish/Subscribe

Gambar 8: Perbandingan Rute Hasil Pengujian Data Cordeau P08 Tanpa *Service Time*

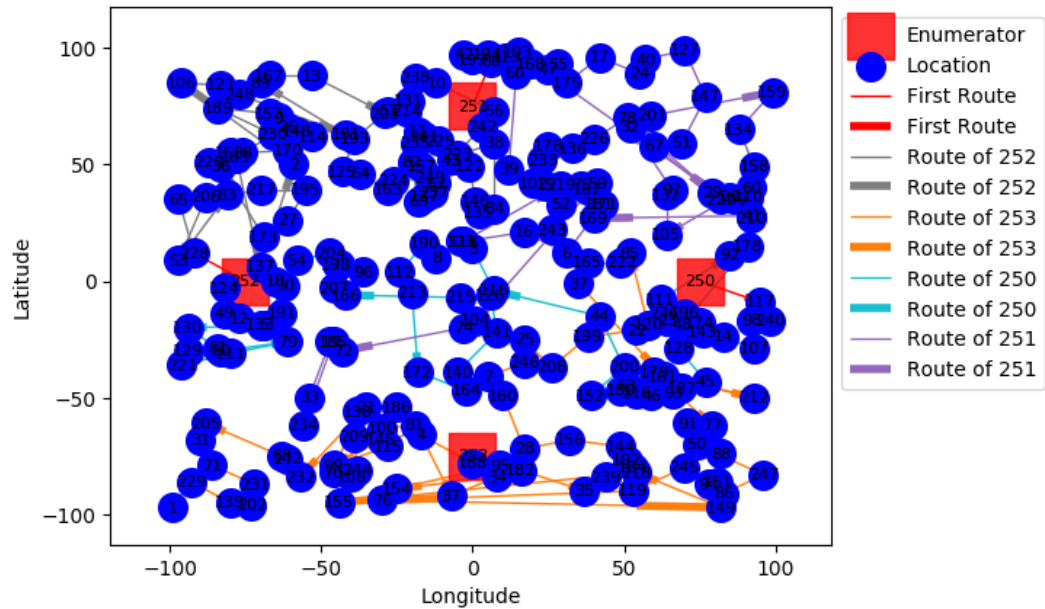


(a) MDVRP berbasis CoEAs

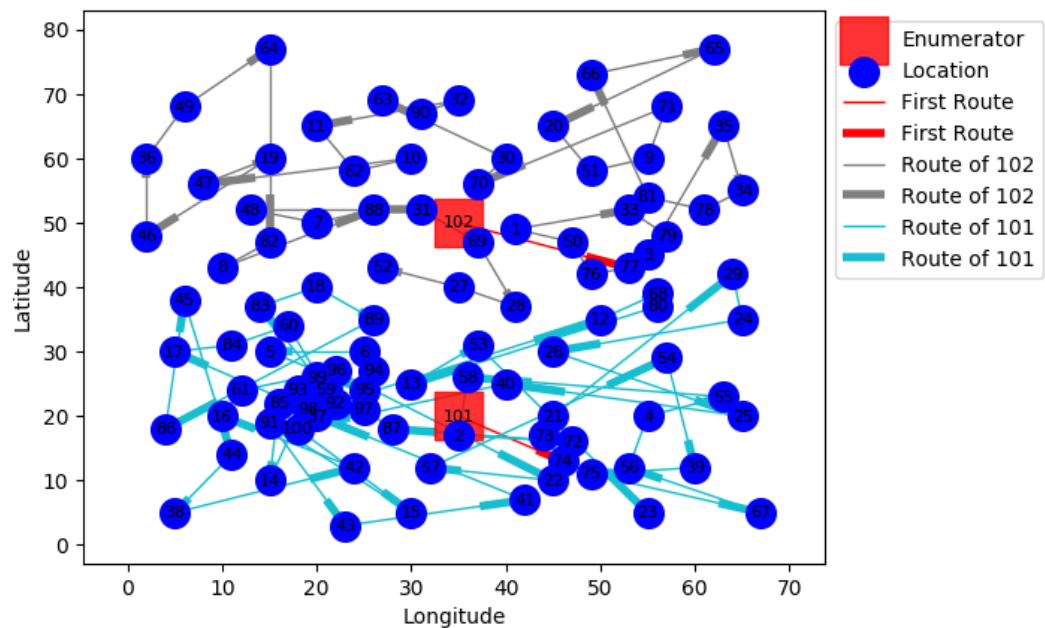


(b) MDVRP berbasis CoEAs dengan Publish/Subscribe

Gambar 9: Perbandingan Rute Hasil Pengujian Data Cordeau P09 Tanpa *Service Time*



(a) MDVRP berbasis CoEAs



(b) MDVRP berbasis CoEAs dengan Publish/Subscribe

Gambar 10: Perbandingan Rute Hasil Pengujian Data Cordeau P10 Tanpa *Service Time*