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

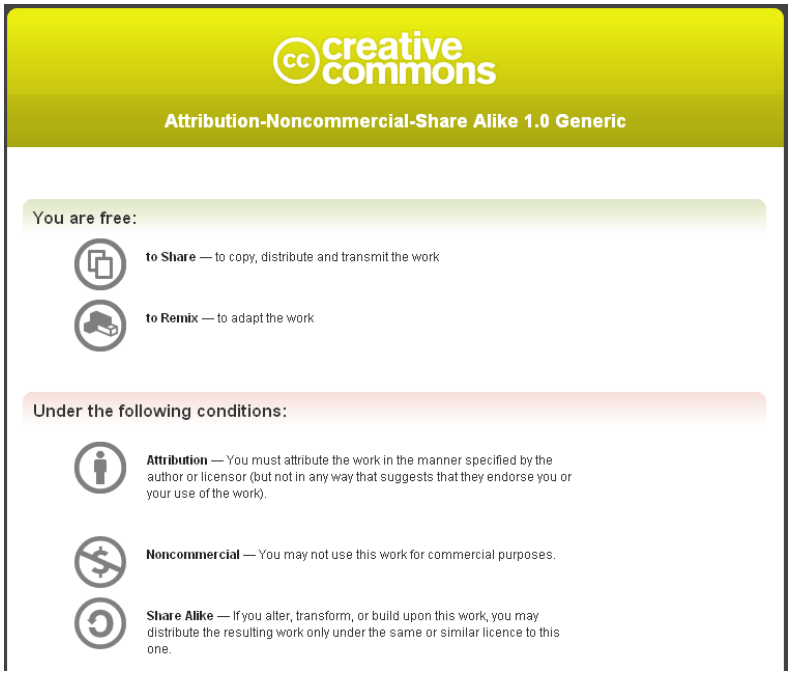
Template ini disediakan untuk orang-orang yang berencana menggunakan LaTeX untuk membuat dokumen tugas akhirnya. Mengapa LaTeX? Ada banyak hal mengapa menggunakan LaTeX, diantaranya:

1. LaTeX membuat kita jadi lebih fokus terhadap isi dokumen, bukan tampilan atau halaman.
2. LaTeX memudahkan dalam penulisan persamaan matematis.
3. Adanya automatis dalam penomoran caption, bab, subbab, subsubbab, referensi, dan rumus.
4. Adanya automatisasi 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.

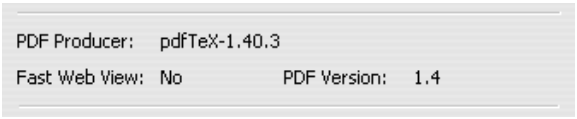
Template ini bebas digunakan dan didistribusikan sesuai dengan aturan *Creative Common License 1.0 Generic*, yang secara sederhana berisi:

Gambar 1 diambil dari http://creativecommons.org/licenses/by-nc-sa/1.0/deed.en_CA. Jika ingin mengentahui 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

demi 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 pernyatan 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

DAFTAR PUSTAKA **71**

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR ALGORITMA

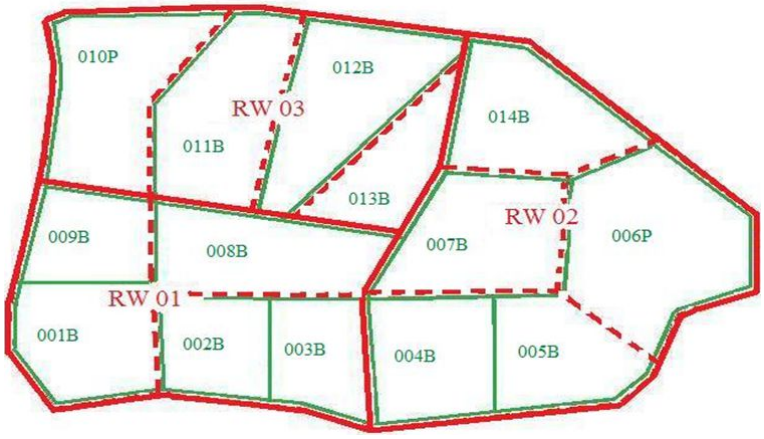
DAFTAR KODE

BAB 1
PENDAHULUAN

1.1 Latar Belakang

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

Dalam pengumpulan data primer atau yang biasa disebut dengan pencacahan, suatu wilayah administrasi dibagi menjadi beberapa Blok Sensus (BS) sebagai satuan wilayah kerja pencacahan (BPS, 2016b). Setiap petugas pencacahan dapat memiliki beban tugas berupa 1 (satu) atau lebih BS, dimana pada masing-masing BS terdapat rumah tangga yang menjadi target pencacahan. Figure 1.1 memberikan ilustrasi pembagian BS dalam desa/kelurahan.



Gambar 1.1: Pembagian Blok Sensus dalam Desa/Kelurahan

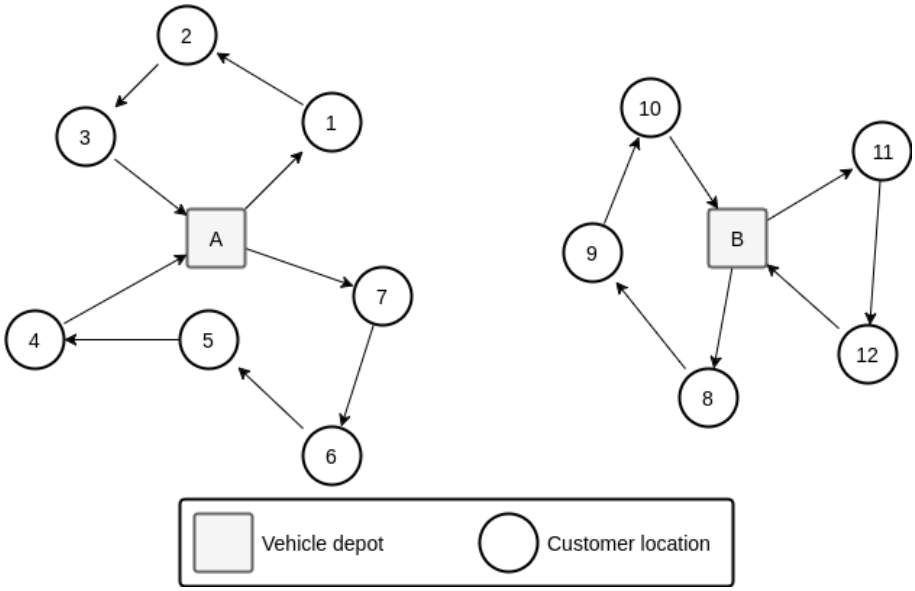
Alokasi BS kepada masing-masing petugas pencacahan bukanlah suatu perkara yang mudah. *Subject matter* selaku penanggung jawab sensus/survey harus membuat alokasi wilayah kerja yang memenuhi kriteria berikut:

1. Minimum total waktu dan biaya
2. Lokasi tugas relatif dekat dengan kantor BPS atau wilayah domisili petugas.

Alokasi wilayah kerja yang kurang cermat dapat mengakibatkan terjadinya ketimpangan dalam total waktu penyelesaian tugas antar pencacah. Kondisi ini dapat menyebabkan terlambatnya kegiatan pencacahan secara keseluruhan.

Total waktu pencacahan terdiri dari 2 (dua) komponen utama, yakni waktu tempuh antar wilayah kerja dan lama pencacahan dari seluruh wilayah kerja. Lama pencacahan dalam sebuah wilayah kerja meliputi lama perpindahan antar rumah tangga dan lama wawancara pada seluruh rumah tangga di wilayah kerja. Berdasarkan (Sudman, 1965), komposisi 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.

Permasalahan alokasi wilayah kerja pencacahan memiliki kesamaan karakteristik *Vehicle Routing Problem* (VRP), atau lebih tepatnya *Multi Depot Vehicle Routing Problem* (MDVRP). 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 varian dari VRP dimana terdapat lebih dari satu depot yang digunakan. Depot merupakan lokasi dimana kendaraan memulai dan mengakhiri perjalanan.



Gambar 1.2: Ilustrasi *Multi Depot Vehicle Routing Problem*

Penyelesaian MDVRP bertujuan untuk mendapatkan solusi dengan total biaya terkecil. Solusi yang dihasilkan berupa sejumlah rute yang terdiri dari sejumlah konsumen yang harus dikunjungi oleh masing-masing kendaraan. Penyelesaian MDVRP harus mengikuti beberapa aturan, yaitu:

1. Setiap konsumen hanya dapat dikunjungi sekali dan hanya oleh satu kendaraan.

2. Setiap kendaraan memulai dan mengakhiri perjalanan pada sebuah depot yang sama.

3. Total *demand* dari seluruh konsumen pada setiap rute tidak melebihi kapasitas dari kendaraan

Pada permasalahan pembagian wilayah kerja, pencacah dianalogikan sebagai kendaraan, wilayah kerja dimisalkan sebagai konsumen, dan lokasi dimana pencacah memulai perjalanan (rumah atau kantor) dapat diibaratkan sebagai depot.

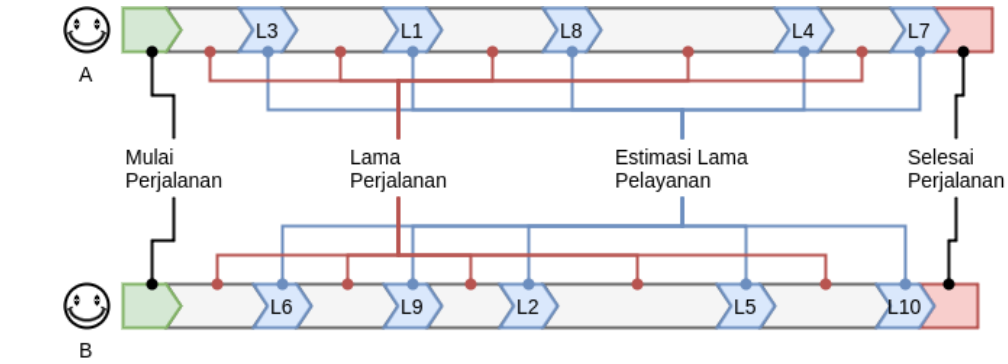
Pada permasalahan MDVRP terdapat 2 (dua) variabel yang dapat mempengaruhi solusi atau rute yang dihasilkan, yaitu biaya tempuh antar konsumen dan lama pelayanan pada setiap konsumen. Biaya tempuh umumnya didefinisikan dalam satuan waktu sehingga dapat didekati dengan lama tempuh. Kedua variabel ini analog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

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

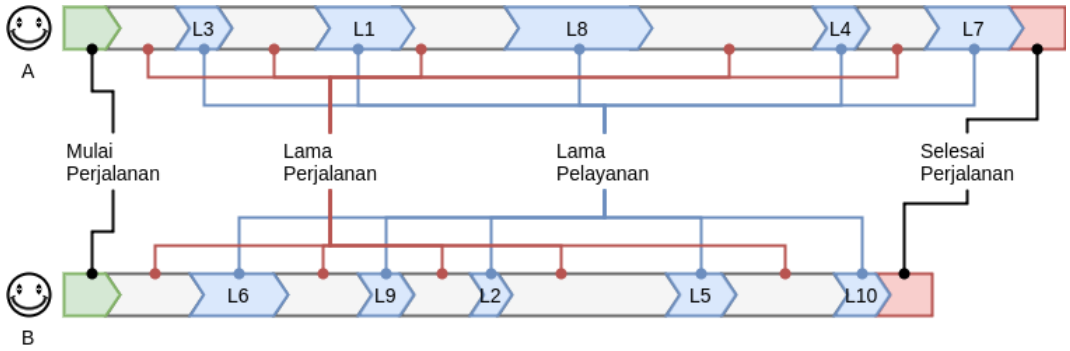
1. Solusi yang diperoleh dengan menggunakan estimasi lama pelayanan pada setiap wilayah kerja menghasilkan rute terbaik yang relatif *equal* dari segi total waktu (Figure 1.3a),

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).

Solusi yang ditawarkan agar rute yang dihasilkan memiliki total waktu pencacahan yang lebih *equal* adalah dengan menggunakan mekanisme pencarian solusi secara bertahap (Figure 1.4). Setiap tahap pencarian solusi tidak dimaksudkan untuk mendapatkan rute lengkap secara keseluruhan, tetapi hanya digunakan untuk menentukan wilayah kerja yang akan dikunjungi berikutnya.

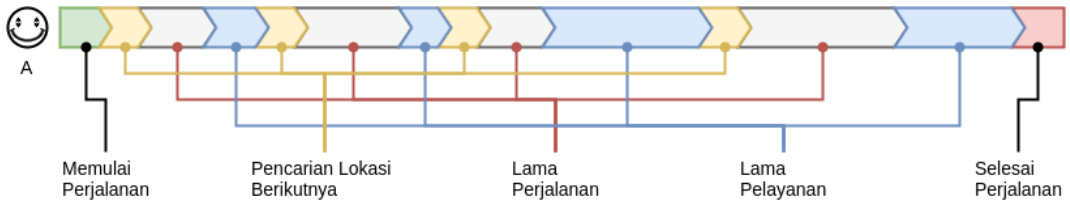


(a) *Timeline* Dengan Estimasi Waktu Pelayanan



(b) *Timeline* Setelah Dikunjungi

Gambar 1.3: *Timeline* Sebelum dan Setelah Dikunjungi



Gambar 1.4: *Timeline* Solusi Bertahap

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 peneliti sepakat bahwa *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* (pencacah yang melakukan *request* 'wilayah kerja berikutnya') dan *server* (pihak yang memproses *request* dan mencari 'wilayah kerja berikutnya'). Ada be-

ragam opsi mekanisme komunikasi *client-server* yang dapat digunakan, seperti *Web service*, *Request/Reply*, dan *Publish/Subscribe* (Weise et al., 2009; Sengoku and Yoshihara, 1998; Sarmenta et al., 2002; Mhl, 2002). Mekanisme yang tepat diperlukan agar komunikasi antara *client* dan *server* dapat dilakukan dalam berbagai kondisi jaringan komunikasi.

Berdasarkan permasalahan diatas, penelitian ini dirancang untuk menciptakan sebuah sistem yang dapat digunakan dalam membuat rekomendasi lokasi pencacahan. Peneliti mengusulkan penggunaan mekanisme pencarian solusi secara bertahap dengan melibatkan *context* dari pencacah dan mekanisme *Publish/Subscribe*.

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 sitem.

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.

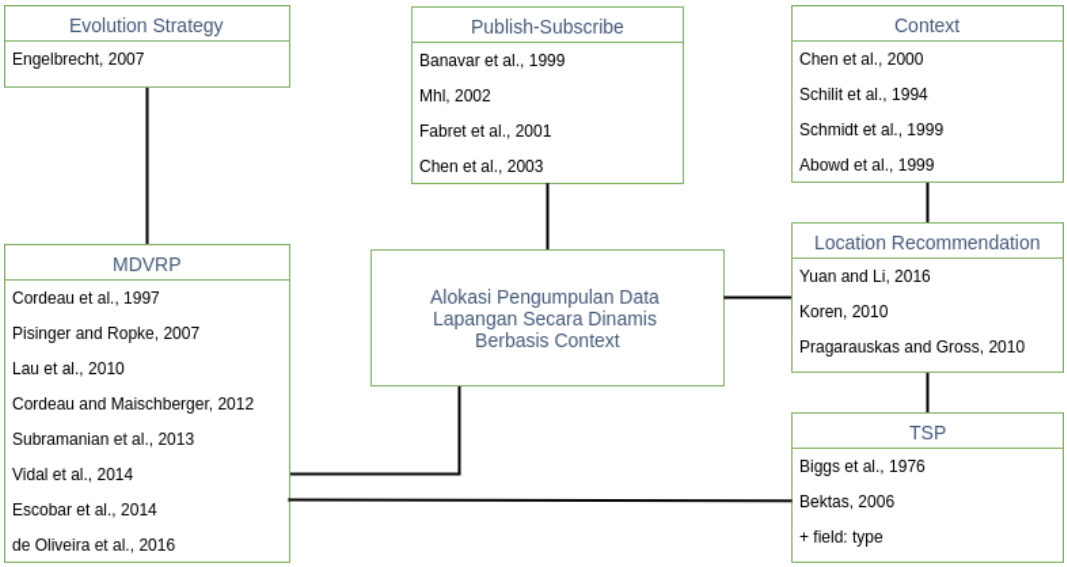
BAB V IMPLEMENTASI DAN PENGUJIAN. Proses dan hasil implementasi rancangan dijelaskan dalam bab ini. Proses pengujian diuraikan mulai dari skenario pengujian hingga hasil pengujian.

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. Sebagaian 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., 2003).

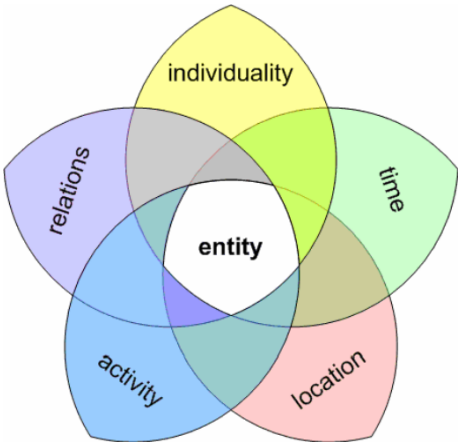
Secara umum, dapat dikatakan bahwasanya *context* adalah informasi yang dapat digunakan dalam menjelaskan situasi dari sebuah entitas. Entitas dapat dapat berupa manusia, lokasi, atau obyek yang relevan dengan interaksi antara seorang pengguna dan aplikasi, termasuk pengguna dan aplikasi itu sendiri (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 *inviduality 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 menkalkulasi 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 dapa 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 relatf, 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 Posisitioning 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., 2010b) menciptakan *drunk driving detection* dengan memanfaatkan sensor *accelerometer* yang tersemat dalam *smartphone*. Sementara (Zou et al., 2016), memanfaatkan sensor *Global Posisioning 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., 2010a; 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. Permasalah lokasi bersifat strategis, sementara permasalahan *routing* bersifat taktis. Rute dapat dikalkukasi 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 perencaan *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 *muti-depot*. 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 sebagaian 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*.

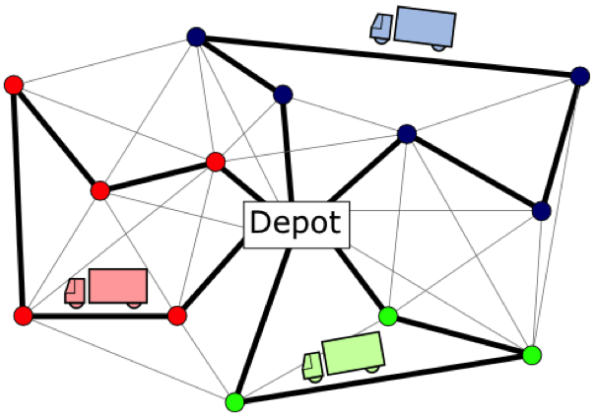
2.4.1 *Vehicle Routing Problem*

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 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 $V = 1,...,n$ 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, makan 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:

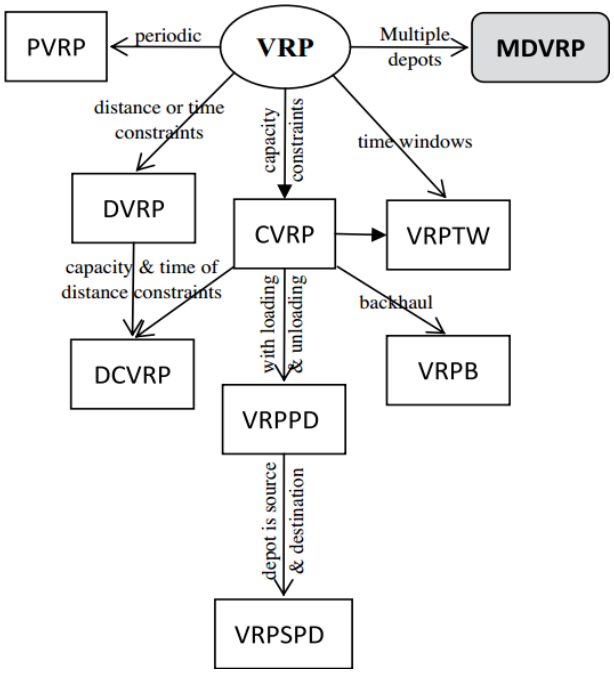
1. Setiap *customer* pada V dikunjungi hanya sekali dan oleh satu kendaraan,
2. Semua kendaraan memulai dan mengakhiri perjalanan pada satu *depot*,
3. Total *demand* dari seluruh *customer* pada setiap rute tidak melebihi Q,
4. Durasi total dari rute tidak melebihi D



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.

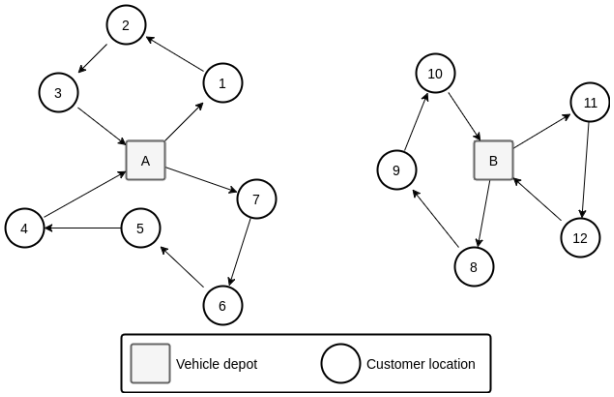
Sementara itu, *Multi-Depot Vehicle Routing Problem (MDVRP)* merupakan salah satu varian dari VRP klasik dimana terdapat lebih dari satu *depot* yang digunakan. Gambar 2.4 memberikan ilustrasi hierarki variasi dari VRP.



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., 1996b), 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 $V_C = \{1,..., N\}$, dan satu set depot $V_D = \{N+1,..., N+M\}$, dimana $V_C \cup V_D = V$ dan $V_C \cap V_D = \oslash$. 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 menkonversi *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.

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

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

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

$$\sum_{i=1}^{N+M} x_{ihk} - \sum_{j=1}^{N+M} x_{hjk} = 0 \tag{2.4}$$
$$(k = 1,...,K; h = 1,...,N+M);$$

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

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

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

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

$$y_i - y_j + (M+N)x_{ijk} \leq N+M-1; \tag{2.9}$$
$$for 1 \leq i \neq j \leq N and 1 \leq k \leq K;$$

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

$$y_i \in \{0,1\} \forall i; \tag{2.11}$$

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., 1996b) 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., 1996a), 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 karakterisik 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.

$$\chi(t) = (x(t), \sigma(t)) \tag{2.12}$$

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

$$\chi'(t) = (x'(t), \sigma'(t)) \tag{2.13}$$

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

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

dan

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

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 μ individuals;
4: for each individual, χ_i(t) ∈ C(t) do
5:    Evaluate the fitness, f(x_i(t));
6: end for
7: while stopping condition(s) not true do
8:    for i = 1,...,λ do
9:       Choose ρ ≥ 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
```

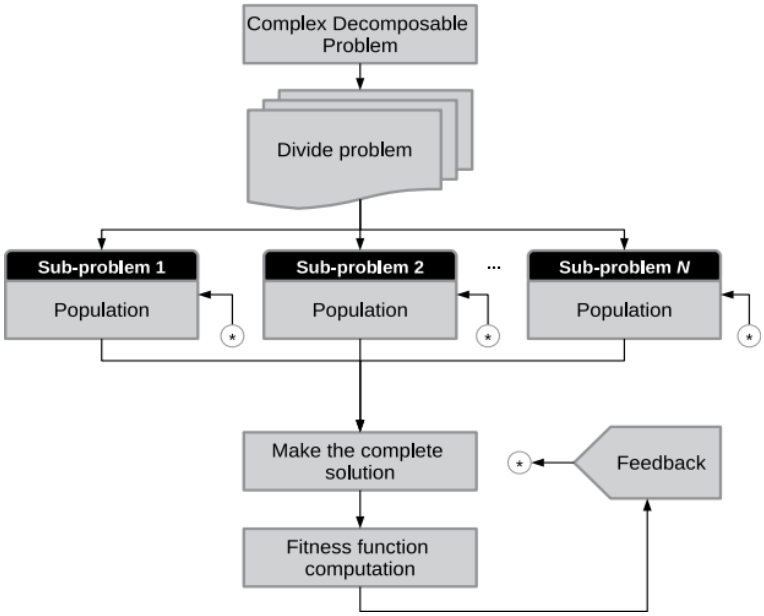
Jika pada algoritma evolusi (EA), evolusi biasanya dipandang sebagai usaha populasi untuk beradaptasi dengan lingkungan yang tetap, terdapat algoritma terkait yang lebih bersifat natural, yaitu algoritma *coevolution* (CoEA). Algoritma *coevolution* adalah algoritma yang bersifat komplemen (saling melengkapi) 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 memdapatkan 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 coevolution* 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*

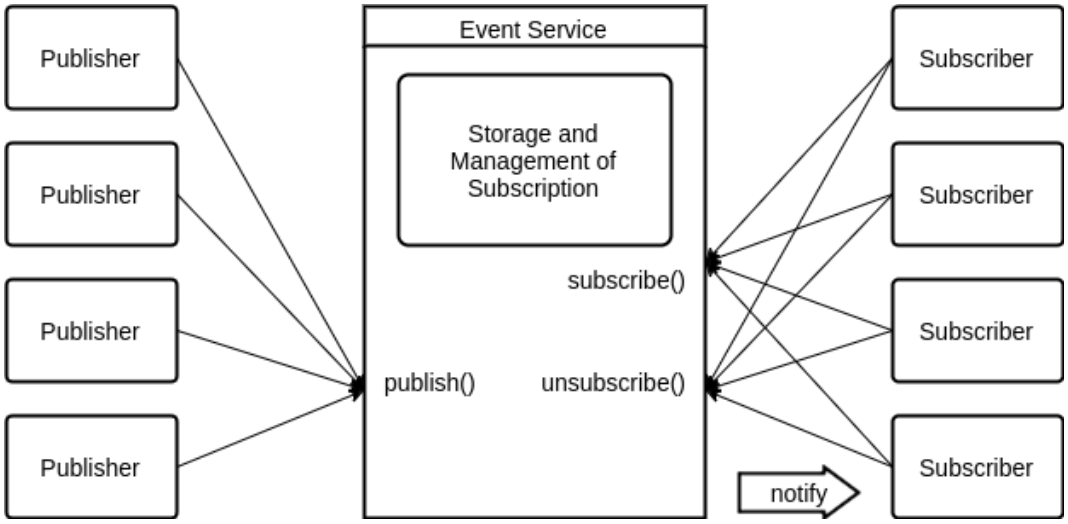
Publish/subscribe interaction merupakan salah satu alternatif untuk komunikasi antara *client* dan *server*. Paradigma interaksi pada *publish/subscribe* adalah adanya *subscriber* yang memiliki ketertarikan pada suatu *event* atau *pattern of event*, agar dapat dikirimkan notifikasi tentang sebuah *event*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

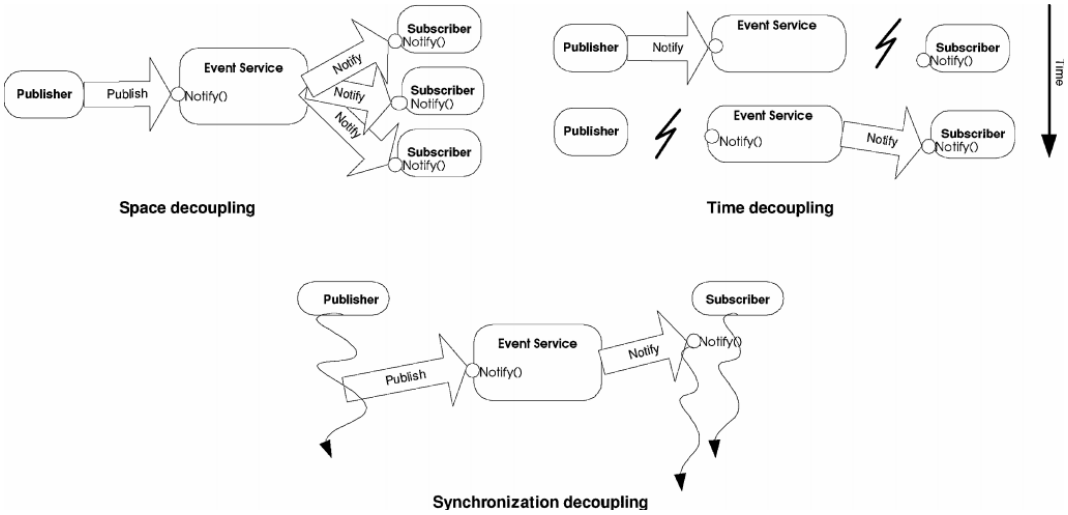


Gambar 2.7: Arsitektur dasar pada Pub/Sub (Eugster et al., 2003)

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.

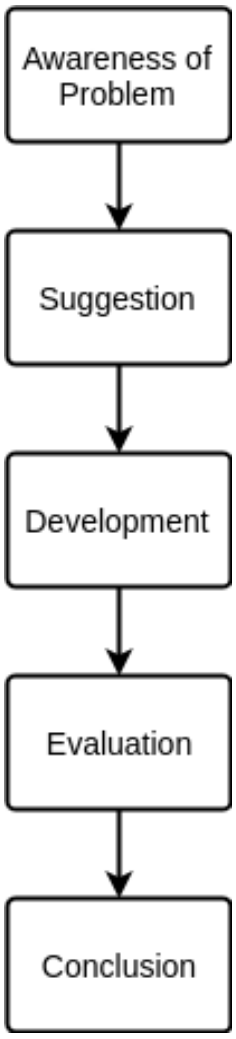
3. *Synchronizing decoupling.*
   Publisher tidak diblok ketika memproduksi *event*, dan juga *subscriber* tetap dapat memperoleh informasi meskipun sedang mengerjakan tugas yang lain. *Publisher* dan *subscriber* tidak berada dalam *main flow*, sehingga tidak berinteraksi secara *synchronous*.



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* (Vaishnavi and Kuechler, 2007). 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 konsekuesinya, terdapat resiko hasil kalkukasi 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. Jspirit bekerja secara terstruktur, mulai dari pendefinisan 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()
        .setId(line[0])
        .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 Jspirit 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

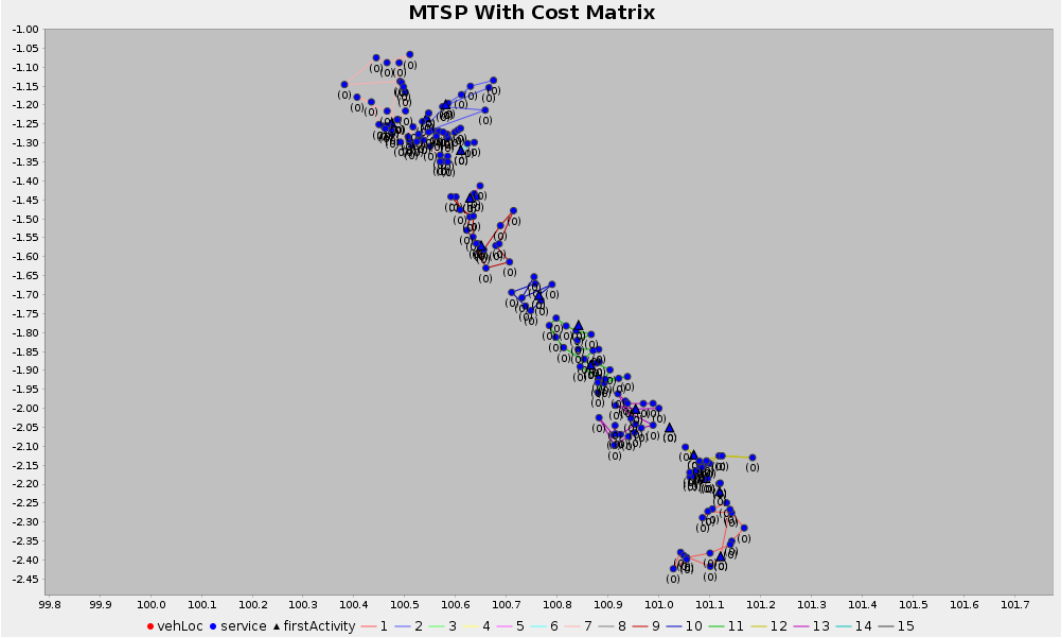
4.1.3 Hasil dan Analisis

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:

$$\mu = 82.24\,jam$$

$$\sigma = 88.95\,jam$$

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 men-*generate* solusi berupa *precalculated routes* untuk petugas A dan B, dimana

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

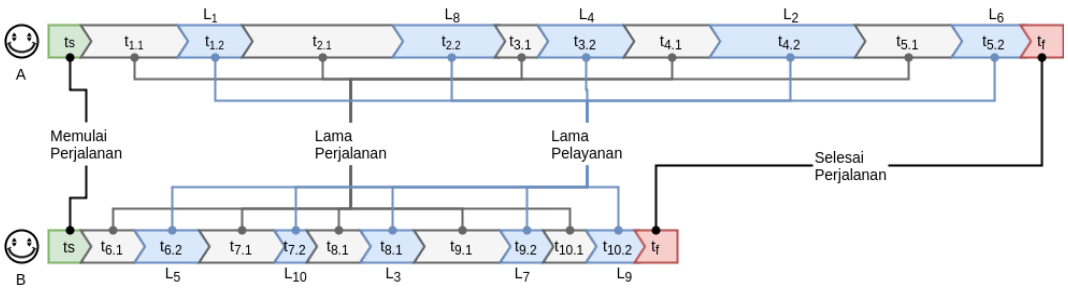
2. Petugas pencacahan A dan B sama-sama memulai tugas pada waktu t_s (*time start*) yang digambarkan sebagai segmen berwarna hijau.

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: Ilustrasi *timeline* Rute yang Dikalkulasi Tanpa Melibatkan *Service Time*

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*. Idenya 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).

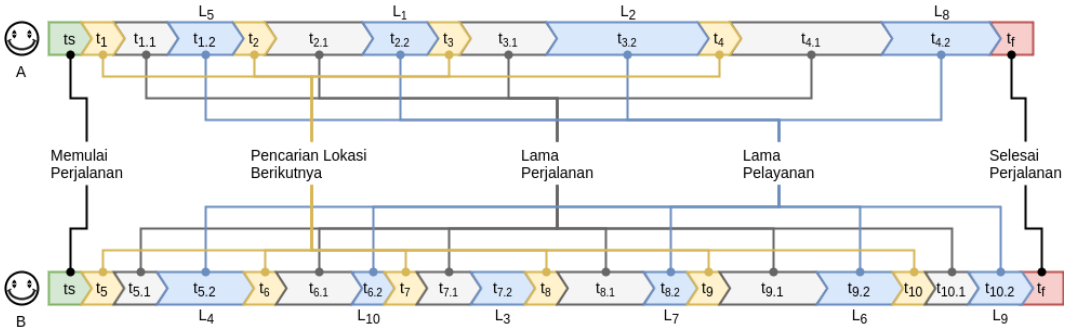
5. Setelah menyelesaikan tugas di lokasi pertama, A dan B mengirimkan *request* ke server untuk mengetahui lokasi berikutnya yang harus dikunjungi. Server kemudian melakukan rekalkulasi rute dengan melakukan pengecekan terhadap 8 lokasi yang belum dikunjungi dan mencocokkannya dengan *current location* 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*.

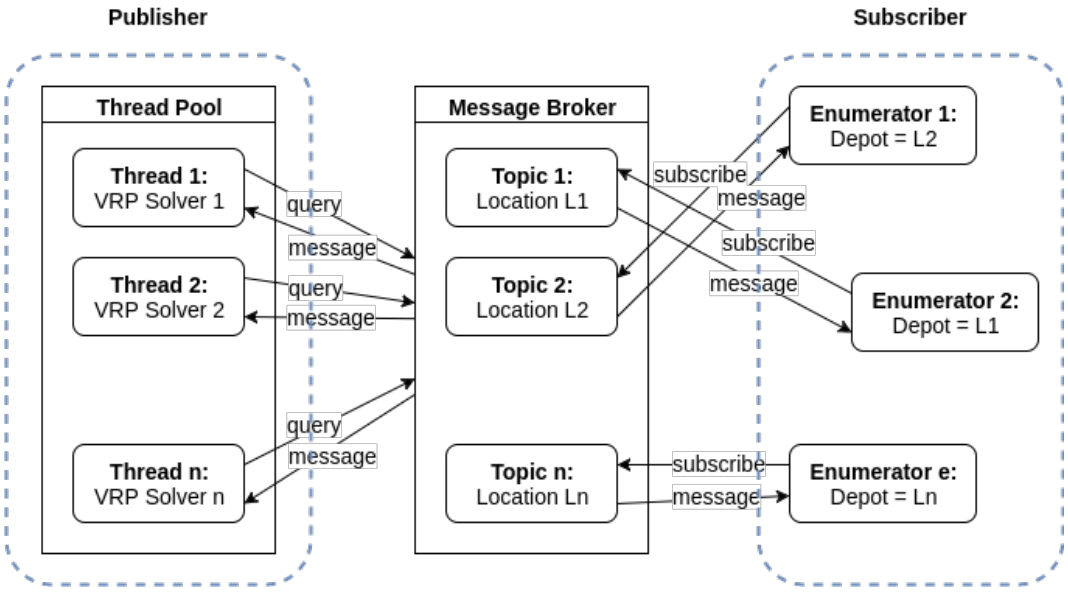


Gambar 4.4: Ilustrasi *timeline* Penerapan MDVRP Secara *Real Time*

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.

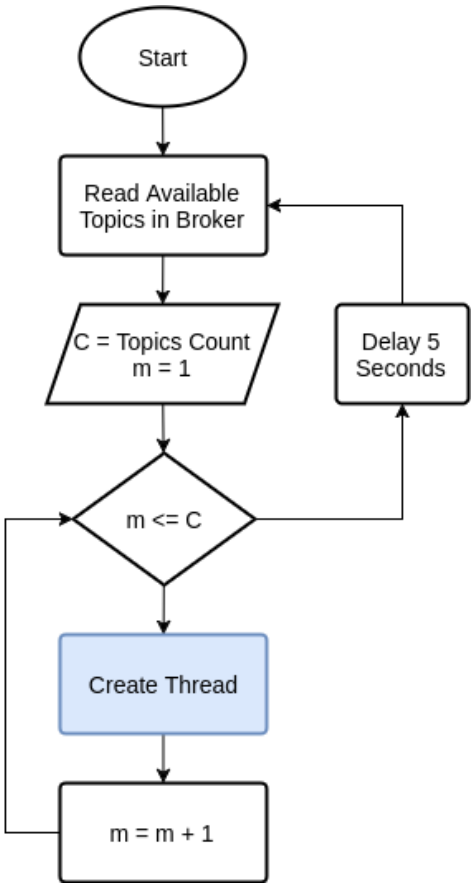
Untuk setiap *request* yang diterima, *publisher* akan menyiapkan sebuah *thread* baru dengan menggunakan *current location* dari *subscriber*/petugas sebagai ID. *Thread* ini dilengkapi dengan sebuah *VRP solver procedure* (Algoritma 4.2) yang akan melakukan pencarian solusi/rute. Seluruh *thread* dari 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

Algoritma 4.1 TopicWatcher

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

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



Gambar 4.6: *Flowchart Topic Watcher*

kedatangan. Setiap eksekusi akan menjalankan *VRPSolver procedure* yang melibatkan *M subscribers* (petugas pencacahan) dan *unassigned* N lokasi pencacahan.

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

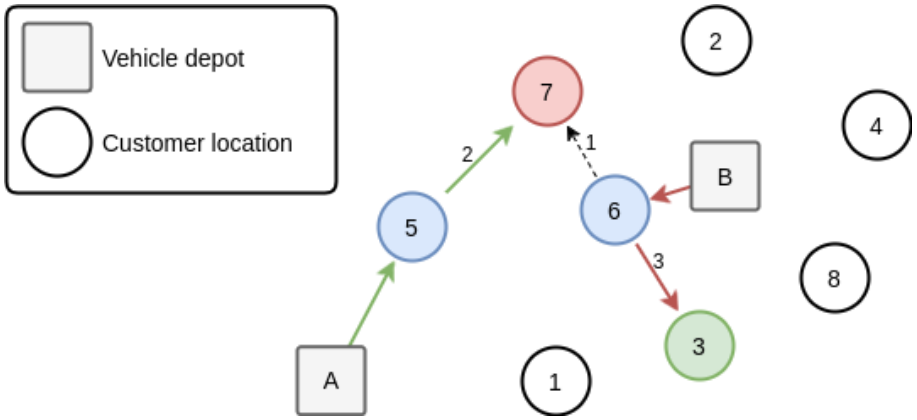
4. Sebagai gambaran, misalkan saat A selesai melakukan pencacahan dan me-*request* 'lokasi berikutnya', lokasi 7 sudah tidak *available* lagi bagi A. Rekomendasi terbaik yang bisa diberikan oleh sistem adalah lokasi 1 yang memiliki jarak yang cukup jauh dari A.

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 seluruh petugas pencacahan yang diikutsertakan dalam kalkulasi rute. Setiap rute R_m $(1 \leq m \leq M)$ yang dihasilkan oleh *VRPSolver* akan di-*publish* kepada petugas m yang merupakan *subscriber* topik (lokasi) C_n $(1 \leq n \leq N)$. *Message broker* kemudian akan melaporkan status pengiriman informasi kepada *publisher* (*'success'* atau *'failed'*). Mekanisme kerja *recommendation publisher* diilustrasikan dengan menggunakan *flowchart* pada Figure 4.8.

Figure 4.9 memberikan gambaran umum tentang proses yang terjadi sejak *request* dikirim oleh petugas hingga solusi/rute diperoleh. Petugas B mengirimkan



Gambar 4.7: Ilustrasi *Global Best Solution*

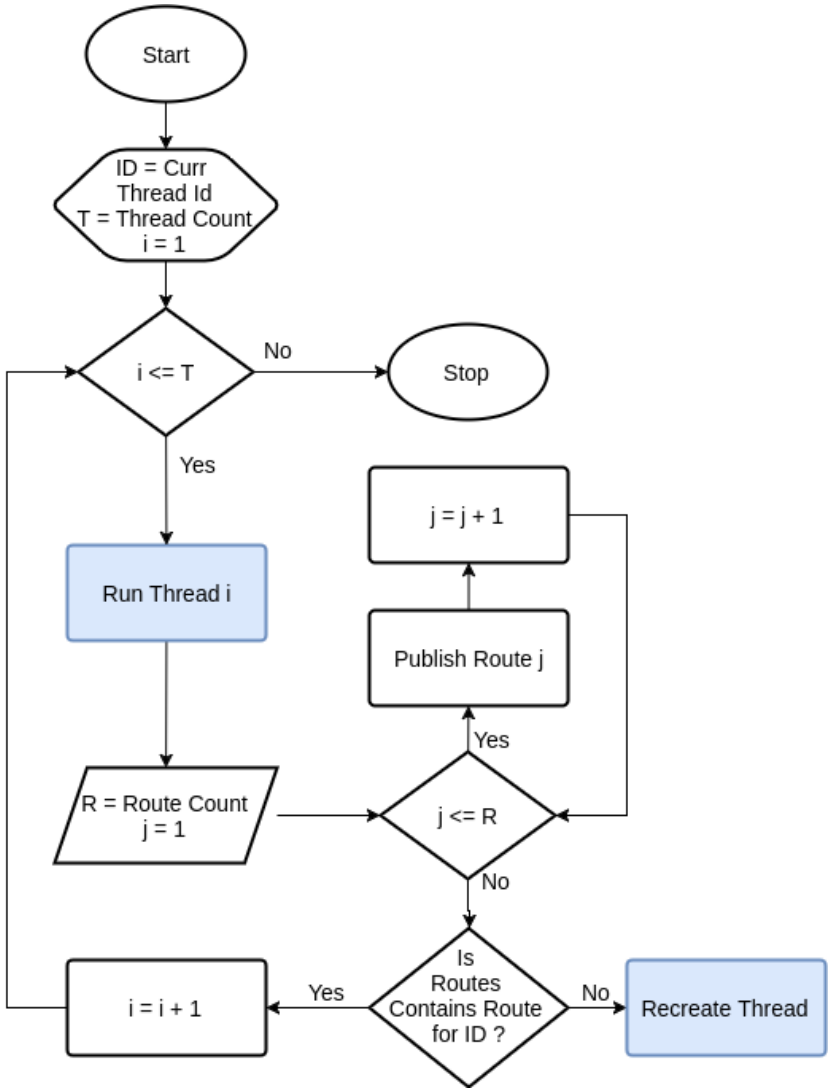
Algoritma 4.2 VRPSolver Procedure

Input: TP // threadpool
Output: *None*

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

request yang diterima oleh *message broker*. *Topic Watcher* yang memantau *message broker* mengetahui keberadaan *request* baru ini dan kemudian melaporkannya kepada *publisher*. *Publisher* kemudian menciptakan sebuah *thread* yang mengandung *VRPSolver procedure*. Saat *thread* dieksekusi, *VRPSolver procedure* akan melakukan kalkulasi solusi/rute terbaik dengan mengikutsertakan seluruh M petugas dan lokasi yang masih berstatus *unassigned*. *Thread* hanya dapat dijalankan secara berurutan berdasarkan prinsip *once at a time*. Sebagai konsekuensinya, ketika petugas A mengirimkan *request* baru, *thread* untuk *request* A tetap akan diciptakan, namun dibiarkan dalam status *idle*. *Thread* A harus menunggu hingga *thread* sebelumnya (*thread* B) selesai diproses.

Saat *thread* B selesai diproses, thread B tidak hanya menghasilkan rute untuk petugas B, tapi juga untuk sejumlah m petugas lainnya ($1 \leq m \leq M$). Rute-rute ini akan di-*publish* kepada petugas yang bersesuaian. Jika suatu rute R_m berhasil

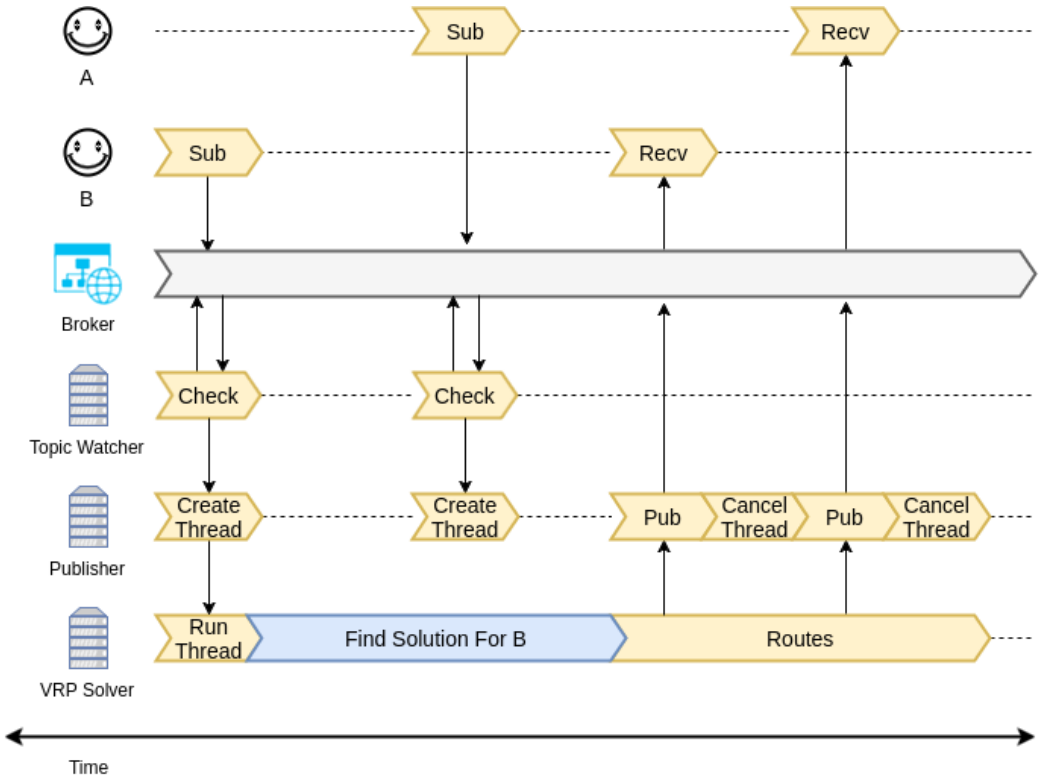


Gambar 4.8: *Flowchart recommendation publisher*

diterima oleh petugas *m*, maka *thread* yang berasosiasi dengan rute tersebut akan dimatikan. Dalam Figure 4.9, *thread* B menghasilkan solusi/rute untuk petugas A dan B dan kedua rute ini berhasil diterima dengan baik oleh masing-masing petugas. Dalam hal ini, *thread* A dan *B* dinilai tidak lagi dibutuhkan sehingga kedua *thread* tersebut akan dihentikan. Penghentian ini dilakukan untuk mencegah terjadinya penghitungan ganda terhadap rute yang sudah diterima oleh *subscriber* (petugas).

Pada praktiknya, dapat terjadi suatu kondisi dimana *VRPSolver procedure* gagal mendapatkan rute untuk petugas B. Pada kondisi demikian, *thread* B akan diproses ulang dengan hanya melibatkan 1 (satu) petugas B saja dan sejumlah lokasi yang masih berstatus *unassigned*. Ini dilakukan untuk memberikan jaminan bahwa akan selalu ada solusi/rute untuk setiap *request*.

Pada subsection 2.5.1 dijelaskan bahwa karakteristik *loose coupling* pada



Gambar 4.9: Ilustrasi *Timeline* Proses Pencarian Solusi Berdasarkan

mekanisme *publish/subscribe* membuat *publisher* dan *subscriber* mampu berkomunikasi secara fleksibel, tanpa perlu mengetahui identitas masing-masing. Namun, ini mengakibatkan informasi mengenai *current location* dari subscriber tidak dapat diperoleh. Untuk itu, sistem dirancang dengan menyertakan *shared memory*, dimana *subscriber* dapat 'menaruh' *current location*-nya untuk kemudian diakses oleh *publisher*.

4.2.3 *VRPSolver Procedure*

VRPSolver procedure merupakan sebuah modul yang terdapat di dalam *thread* dan digunakan untuk melakukan kalkulasi rute yang harus dikunjungi oleh petugas. *VRPSolver procedure* dibangun berdasarkan algoritma penyelesaian 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 *coevolutionay algorithms*

(CoEAs) dipilih karena menghasilkan rute dengan *total cost* yang minimal dan waktu pemrosesan yang relatif singkat dibandingkan algoritma yang lain.

Langkah-langkah yang digunakan dalam implementasi algoritma CoEAs pada *VRPSolver procedure* adalah sebagai berikut:

1. Definisi masalah
   Masalah didefinisikan sebagai kumpulan informasi tentang sejumlah M petugas, N lokasi pencacahan, dan D *initial location/depot*. Setiap lokasi i, dimana $i \in (D \cup N)$, dilengkapi dengan koordinat lokasi (x_i, y_i). Setiap pasangan lokasi i dan j, dimana $i, j \in (D \cup N)$, memiliki *cost* sebesar c_{ij} yang merepresentasikan waktu yang diperlukan untuk melakukan perjalanan dari i ke j.

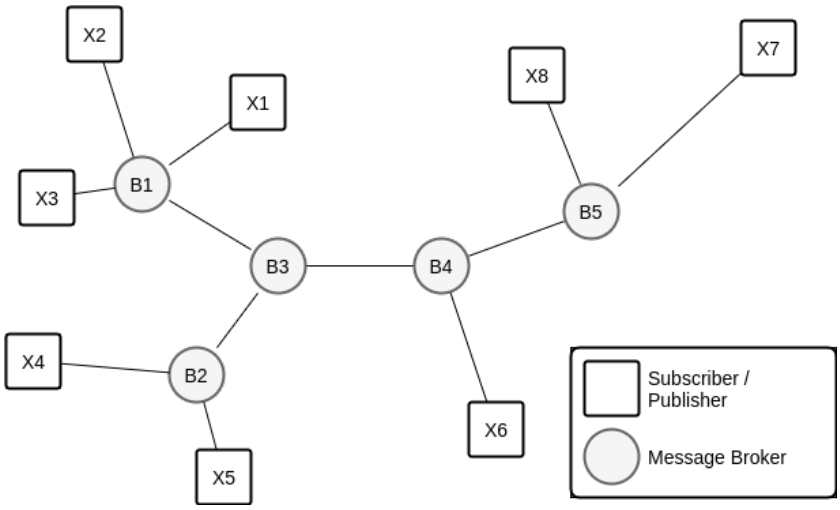
2. Dekomposisi masalah
   Masalah yang telah didefinisikan sebelumnya akan didekomposisi (dipecah) menjadi beberapa *chunk* submasalah agar proses kalkulasi rute dapat dilakukan secara paralel.

3. Evolusi
   Setiap submasalah berisi kombinasi pasangan petugas dan lokasi yang akan dikunjungi (*enumerator-locations pair*). Setiap kombinasi akan mengalami proses evolusi, dimana susunan formasi pasangan petugas dan lokasi mengalami perubahan untuk menemukan formasi yang lebih baik. Evolusi terjadi secara iteratif dan setiap iterasi akan menghasilkan formasi baru (*new best formation*). *New best formation* ini akan dievaluasi dengan cara membandingkannya dengan formasi terbaik yang diperoleh dari iterasi sebelumnya (*current best formation*). Jika *new best formation* lebih baik daripada *current best formation*, maka *current best formation* akan di-*overwrite* dengan *new best formation*. Iterasi akan terus berlangsung hingga tercapai suatu *time limit* yang telah ditentukan. Dalam penelitian ini, *time limit* yang digunakan adalah 60 detik atau 40 detik jika tidak ada *new best formation* yang terbentuk. Formasi yang diperoleh pada akhir keseluruhan iterasi akan dipilih oleh *VRPSolver* sebagai solusi final.

4.2.4 *Message Broker*

Message broker merupakan subsistem yang bertanggung jawab dalam menyalurkan (*routing*) pesan dari *publisher* ke *subscriber* sesuai dengan topik yang di-*subscribe* (Banavar et al., 1999). Suatu mekanisme *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*, *subscriber* maupun *publisher* dapat terkoneksi pada *broker* terdekat. Arsitektur *multi broker* ini disebut juga dengan istilah *distributed publish/subscribe system* (Mhl, 2002), seperti ilustrasi pada Figure 4.10. Sistem yang diusulkan akan menerapkan arsitektur terdistribusi agar dapat menangani lokasi pencacahan yang tersebar secara geografis.



Gambar 4.10: Ilustrasi *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 penggunaaan 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 keringkasan 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 maksumum 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.

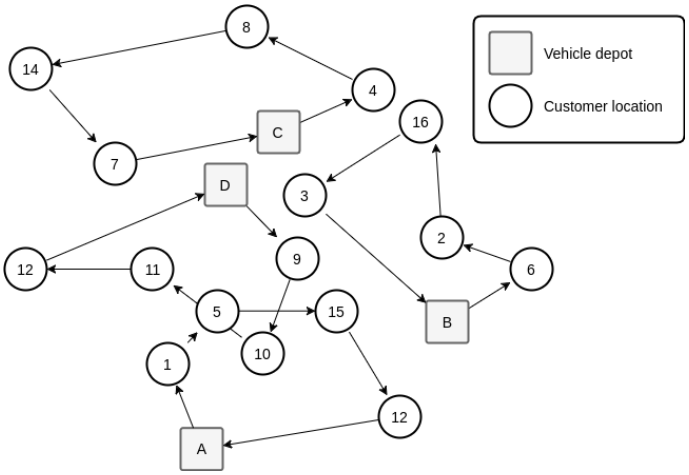
5.2.2.2 *Metric*

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 $\rightarrow$ Loc5 $\rightarrow$ Loc15 $\rightarrow$ Loc12
- *Vehicle* B = Loc 6 $\rightarrow$ Loc2 $\rightarrow$ Loc16 $\rightarrow$ Loc3
- *Vehicle* C = Loc4 $\rightarrow$ Loc8 $\rightarrow$ Loc14 $\rightarrow$ Loc 7
- *Vehicle* D = Loc9 $\rightarrow$ Loc10 $\rightarrow$ Loc11 $\rightarrow$ 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, 2017b). 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, 2017a), 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*. Configurasi 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: c5f6764d82ed10793c0c81e54830b4f68b1eacd7 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: c5f6764d82ed10793c0c81e54830b4f68b1eacd7 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 di*publish* 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
```

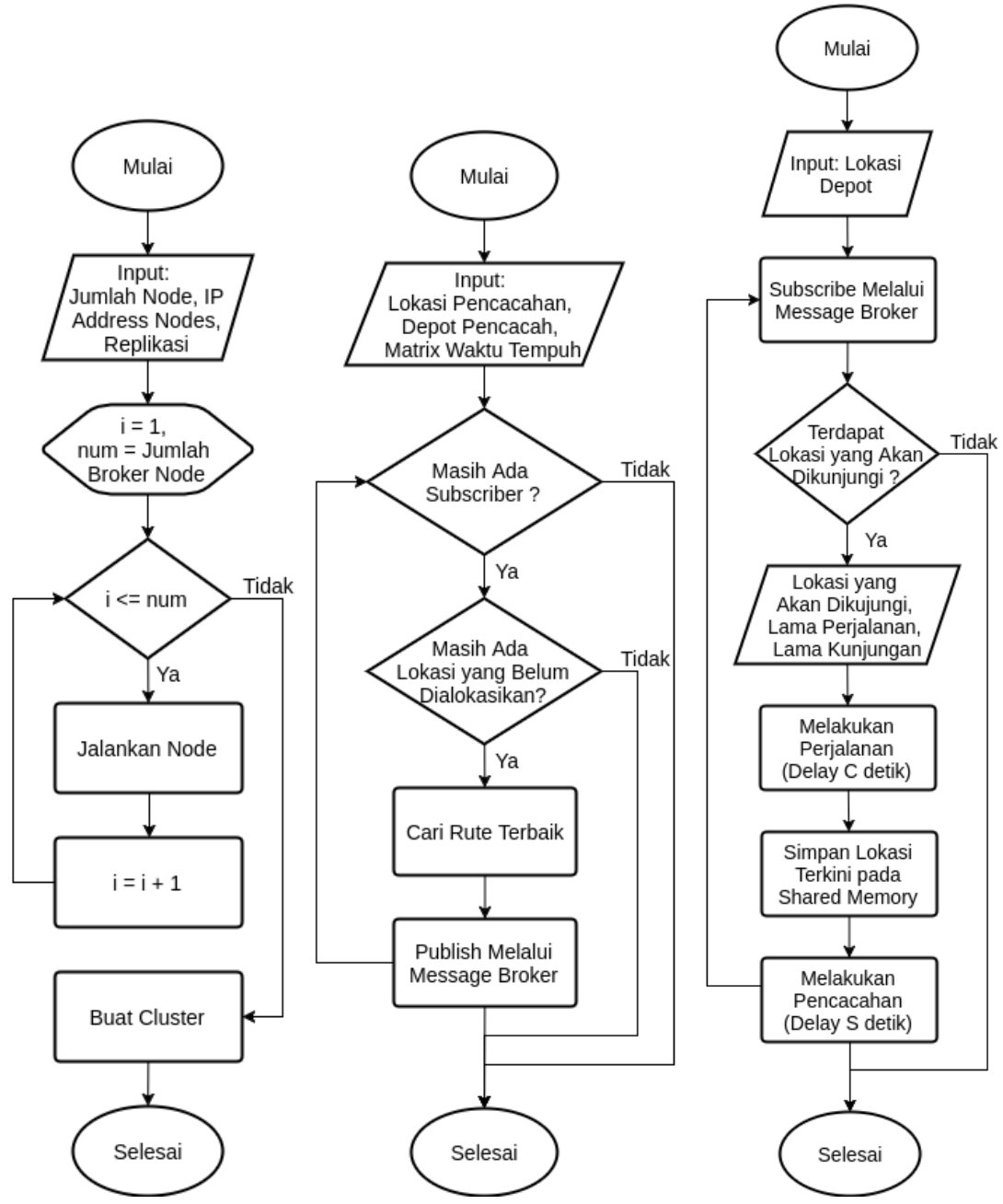
Kode 5.4: Format penggunaan Publisher

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 *customer*nya. 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, dengan masing-masing komposisi sebagai berikut:

1. Instance P01 terdiri dari 50 *customer* dan 5 kendaraan,
2. Instance P02 terdiri dari 50 *customer* dan 4 kendaraan,
3. Instance P03 terdiri dari 75 *customer* dan 5 kendaraan,
4. Instance P04 terdiri dari 100 *customer* dan 2 kendaraan,
5. Instance P05 terdiri dari 100 *customer* dan 2 kendaraan,
6. Instance P06 terdiri dari 100 *customer* dan 3 kendaraan,
7. Instance P07 terdiri dari 100 *customer* dan 4 kendaraan,



(a) *Flowchart Message Broker* **(b)** *Flowchart Publisher* **(c)** *Flowchart Subscriber*

Gambar 5.2: *Flowchart* Subsistem pada Pengujian

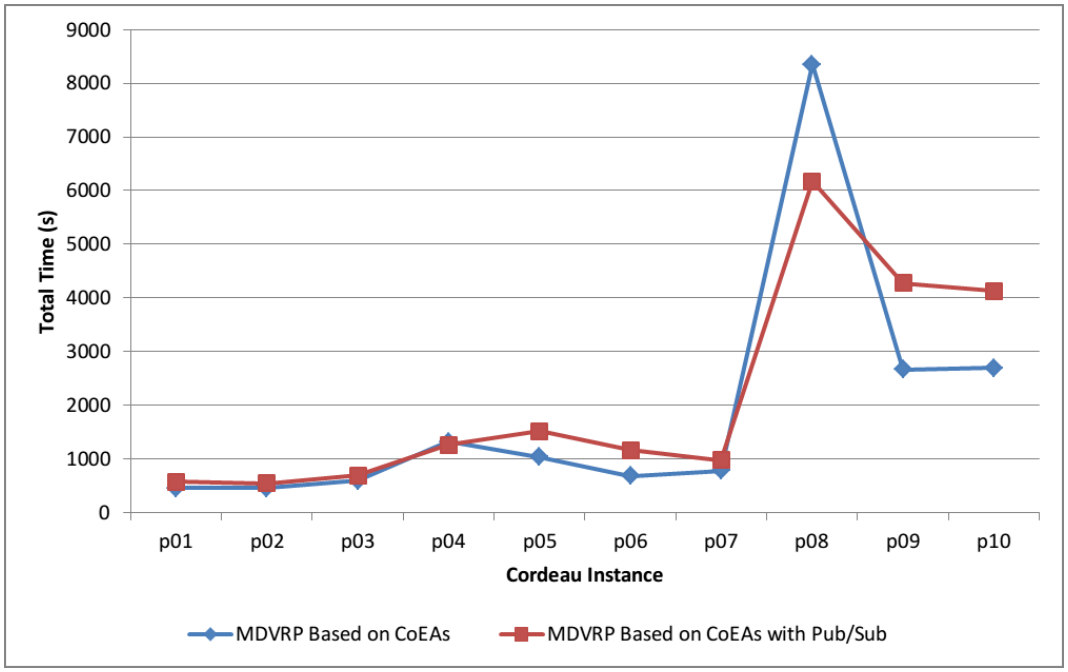
8. Instance P08 terdiri dari 249 *customer* dan 2 kendaraan,
9. Instance P09 terdiri dari 249 *customer* dan 3 kendaraan,
10. Instance P10 terdiri dari 249 *customer* dan 4 kendaraan.

Terminologi kendaraan dan *customer* pada data Cordeau analog dengan pencacah dan lokasi pencacahan pada permasalahan rekomendasi lokasi pencacahan.

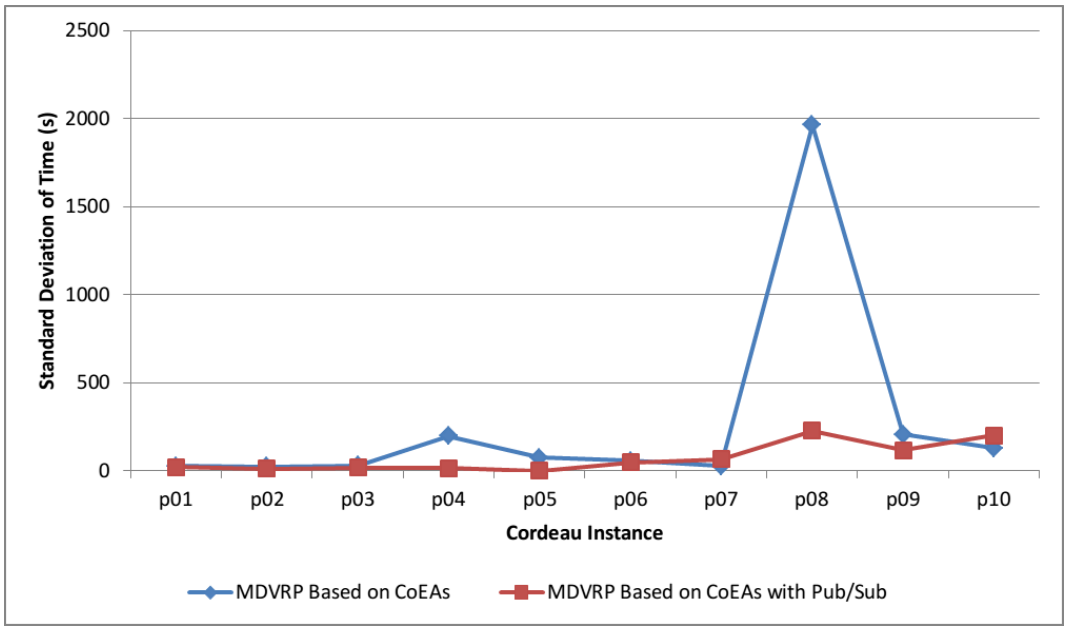
Dari hasil simulasi, diperoleh rute untuk masing-masing *instance* Cordeau sebagaimana digambarkan pada Lampiran 1. Kemudian dari seluruh rute yang diperoleh, dikalkulasi *total time* untuk masing-masing rute, sebagaimana cara yang dijelaskan pada subsubsection 5.2.2.2. Berdasarkan Table 5.1, diperoleh 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.

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

Instance	Total Waktu CoES MDVRP (det)	Total Waktu CoES MDVRP + Pub/Sub (det)	Stdev Waktu CoES MDVRP (det)	Stdev Waktu CoES MDVRP + Pub/Sub (det)
p01	**454.91**	573.77	27.24	**20.84**
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 sesuatupun yang menyebabkan penundaaan. Pengujian kondisi ini akan dilakukan satu kali untuk masing-masing data set, Cordeau

P01 sampai P10 dan data lapangan.

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. Meskipun komposisi waktu (Sudman, 1965) dinilai tidak lagi relevan, akan tetapi pada penelitian ini hanya digunakan sebagai gambaran. Adapun penggunaan komposisi waktu yang lebih relevan terdapat pada subsubsection 5.2.3.6.

Berikut adalah komposisi dari masing-masing *instance* Cordeau:

1. Instance P01 terdiri dari 50 *customer* dan 5 kendaraan, lama wawancara 27.58 menit dengan standar deviasi 12.13 menit.
2. Instance P02 terdiri dari 50 *customer* dan 4 kendaraan, lama wawancara 27.58 menit dengan standar deviasi 12.13 menit.
3. Instance P03 terdiri dari 75 *customer* dan 5 kendaraan, lama wawancara 27.58 menit dengan standar deviasi 12.13 menit.
4. Instance P04 terdiri dari 100 *customer* dan 2 kendaraan, lama wawancara 27.58 menit dengan standar deviasi 12.13 menit.
5. Instance P05 terdiri dari 100 *customer* dan 2 kendaraan, lama wawancara 27.58 menit dengan standar deviasi 12.13 menit.
6. Instance P06 terdiri dari 100 *customer* dan 3 kendaraan, lama wawancara 27.58 menit dengan standar deviasi 12.13 menit.
7. Instance P07 terdiri dari 100 *customer* dan 4 kendaraan, lama wawancara 27.58 menit dengan standar deviasi 12.13 menit.
8. Instance P08 terdiri dari 249 *customer* dan 2 kendaraan, lama wawancara 27.58 menit dengan standar deviasi 12.13 menit.

9. Instance P09 terdiri dari 249 *customer* dan 3 kendaraan, lama wawancara 27.58 menit dengan standar deviasi 12.13 menit.

10. Instance P10 terdiri dari 249 *customer* dan 4 kendaraan, lama wawancara 27.58 menit dengan standar deviasi 12.13 menit..

Adapun data lapangan memiliki komposisi sebagai berikut:

1. Instance TW1 terdiri dari 182 *customer* dan 15 kendaraan, lama wawancara 27.58 menit dengan standar deviasi 4,13 menit.

2. Instance TW2 terdiri dari 182 *customer* dan 15 kendaraan, lama wawancara 71.58 menit dengan standar deviasi 12.16 menit.

3. Instance TW3 terdiri dari 182 *customer* dan 15 kendaraan, lama wawancara 27.35 menit dengan standar deviasi 25.34 menit.

4. Instance TW4 terdiri dari 182 *customer* dan 15 kendaraan, lama wawancara 71.58 menit dengan standar deviasi 56.87 menit..

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 subsubsection 5.2.2.2. Berdasarkan Table 5.1, diperoleh hasil bahwasannya dengan menggunakan sistem usulan, seluruh *instance* menghasilkan **total waktu** yang lebih kecil. Sementara dari sisi **standar deviasi**, diperoleh hasil bahwa seluruh *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.

Tabel 5.2: Hasil Pengujian Kondisi Normal Pada Data Cordeau Dengan *Service Time*

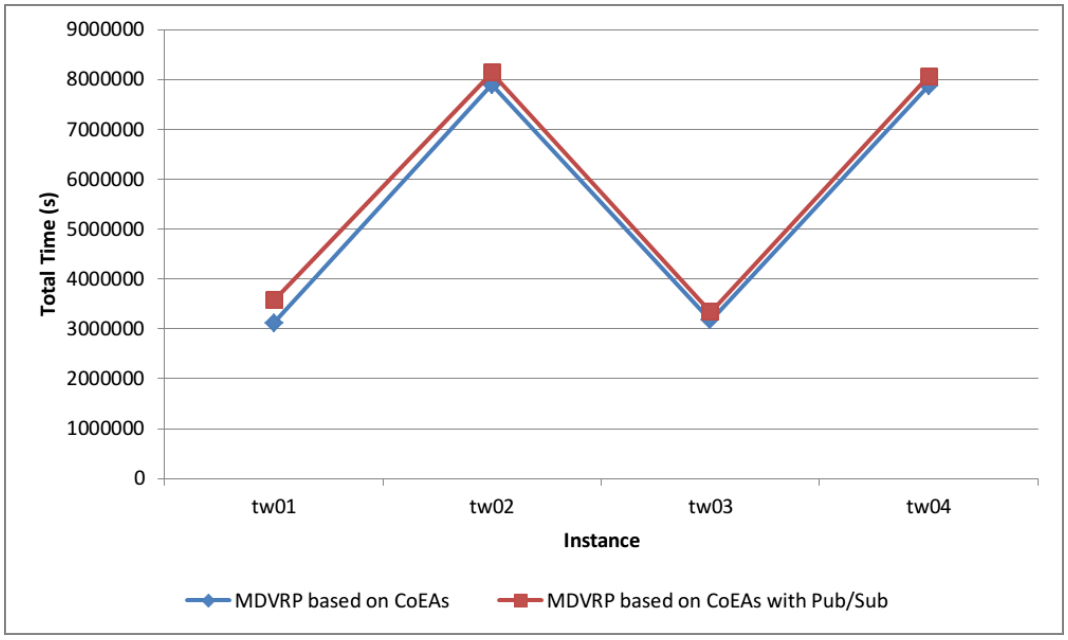
Instance	Total Waktu CoES MDVRP (det)	Total Waktu CoES MDVRP + Pub/Sub (det)	Stdev Waktu CoES MDVRP (det)	Stdev Waktu CoES MDVRP + Pub/Sub (det)
p01	**824,513.17**	824,649.98	44,165.10	**12,539.43**
p02	**805,990.03**	806,090.97	37,666.96	**12,322.08**
p03	**1,256,883.28**	1,257,044.56	57,486.44	**13,271.86**
p04	**1,642,834.05**	1,643,051.50	148,762.94	**4,719.26**
p05	**1,654,118.92**	1,654,511.94	154,028.63	**22,469.85**

Instance	Total Waktu CoES MDVRP (det)	Total Waktu CoES MDVRP + Pub/Sub (det)	Stdev Waktu CoES MDVRP (det)	Stdev Waktu CoES MDVRP + Pub/Sub (det)
p06	**1,634,803.49**	1,635,193.62	103,266.13	**10,160.24**
p07	**1,639,779.18**	1,639,996.92	81,233.61	**5,492.17**
p08				
p09				
p10	**4,115,466.02**	4,117,128.75	223,253.75	**13,662.48**

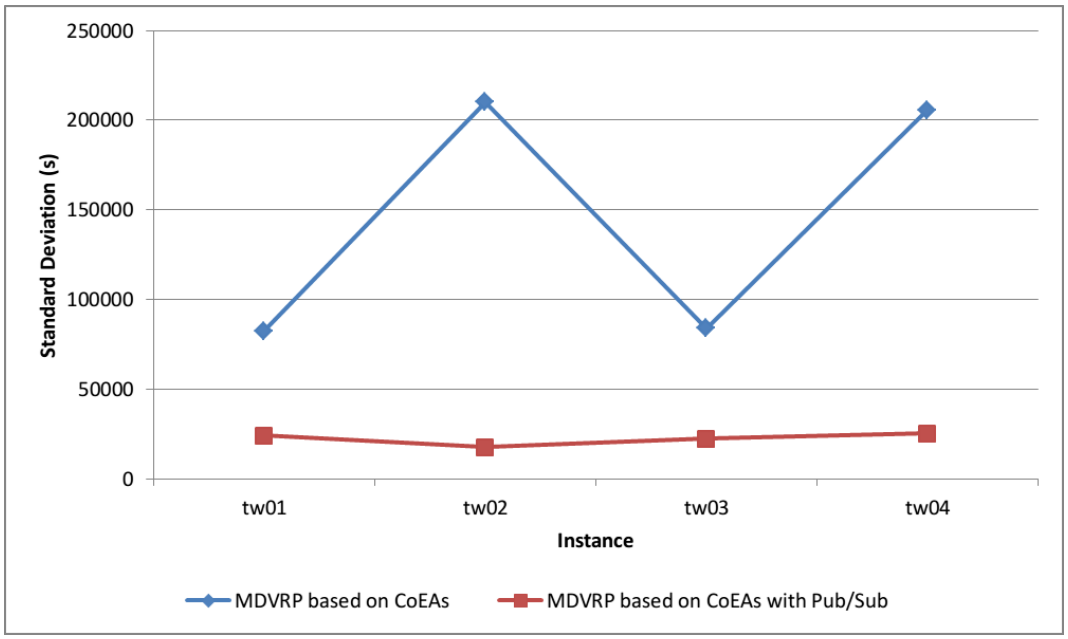
Sementara itu, pada pengujian dengan menggunakan data lapangan, diperoleh hasil sebagaimana digambarkan pada Lampiran 1. Kemudian dari seluruh rute yang diperoleh, dikalkulasi *total time* untuk masing-masing rute, sebagaimana cara yang dijelaskan pada subsubsection 5.2.2.2. Berdasarkan Table 5.3, diperoleh hasil bahwasannya **total waktu** yang dihasilkan dengan menggunakan sistem usulan keseluruhannya lebih besar dibandingkan dengan menggunakan aplikasi pembanding. Sementara dari sisi **standar deviasi**, keseluruhan *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 5.5 dan Figure 5.6 menggambarkan perbandingan hasil pengujian dari 4 *instance* data lapangan.

Tabel 5.3: Hasil Pengujian Kondisi Normal Pada Data Lapangan Dengan *Service Time*

Instance	Total Waktu CoES MDVRP (det)	Total Waktu CoES MDVRP + Pub/Sub (det)	Stdev Waktu CoES MDVRP (det)	Stdev Waktu CoES MDVRP + Pub/Sub (det)
tw01	**3,119,907.52**	3,574,996.52	82,529.44	**24,134.25**
tw02	**7,892,350.51**	8,136,451.51	210,200.68	**17,801.33**
tw03	**3,166,715.70**	3,346,362.70	83,961.70	**22,296.01**
tw04	**7,868,523.23**	8,060,837.23	205,597.64	**25,430.96**



Gambar 5.5: Perbandingan Waktu Total Pengujian Kondisi Normal Pada Data Lapangan Dengan *Service Time*



Gambar 5.6: Perbandingan Standar Deviasi Waktu Pengujian Kondisi Normal Pada Data Lapangan Dengan *Service Time*

5.2.3.6 Pengujian Kondisi *Delay* dengan *Service Time*

Skenario pengujian kondisi *delay* dimaksudkan untuk membandingkan program yang dijalankan pada kondisi terjadi hal-hal yang menghambat jalannya pencacahan. Pengujian ini digunakan sebagai cerminan kodisi lapangan dimana pada bebe-

rapa lokasi tidak terdapat koneksi yang stabil, sehingga untuk melakukan *subscription* lokasi berikutnya perlu berpindah ke lokasi lain. Pengujian dilakukan sebanyak 7 (tujuh) kali, dimana masing-masing mencerminkan kondisi yang berbeda, yaitu:

1. Pengujian dengan kode *instance* d01 menggambarkan kondisi normal. Parameter yang digunakan pada *instance* ini adalah sebagai berikut:
    - Jumlah responden pada setiap segmen/blok sensus adalah 10 rumah tangga
    - Rata-rata wawancara pada setiap rumah tangga 27,58 menit dengan standar deviasi 12,13 menit.
    - Rata-rata waktu yang diperlukan untuk mendapatkan koneksi internet sebesar 5 menit dengan standar deviasi 3 menit.
2. Pengujian dengan kode *instance* d02 menggambarkan kondisi dimana anggota rumah tangga sedikit dan relatif homogen, dengan koneksi internet relatif mudah didapatkan. Parameter yang digunakan pada *instance* ini adalah sebagai berikut:
    - Jumlah responden pada setiap segmen/blok sensus adalah 10 rumah tangga
    - Rata-rata wawancara pada setiap rumah tangga 27,58 menit dengan standar deviasi 4,16 menit.
    - Rata-rata waktu yang diperlukan untuk mendapatkan koneksi internet sebesar 5 menit dengan standar deviasi 3 menit.
3. Pengujian dengan kode *instance* d03 menggambarkan kondisi dimana anggota rumah tangga sedikit dan relatif homogen, dengan koneksi internet secara umum sulit didapatkan. Parameter yang digunakan pada *instance* ini adalah sebagai berikut:
    - Jumlah responden pada setiap segmen/blok sensus adalah 10 rumah tangga
    - Rata-rata wawancara pada setiap rumah tangga 27,58 menit dengan standar deviasi 4,16 menit.
    - Rata-rata waktu yang diperlukan untuk mendapatkan koneksi internet sebesar 60 menit dengan standar deviasi 3 menit.

4. Pengujian dengan kode *instance* d04 menggambarkan kondisi dimana anggota rumah tangga banyak dan relatif homogen, dengan koneksi internet relatif mudah didapatkan. Parameter yang digunakan pada *instance* ini adalah sebagai berikut:

    - Jumlah responden pada setiap segmen/blok sensus adalah 10 rumah tangga
    - Rata-rata wawancara pada setiap rumah tangga 71,58 menit dengan standar deviasi 12,16 menit.
    - Rata-rata waktu yang diperlukan untuk mendapatkan koneksi internet sebesar 5 menit dengan standar deviasi 3 menit.

5. Pengujian dengan kode *instance* d05 menggambarkan kondisi dimana anggota rumah tangga banyak dan relatif homogen, dengan koneksi internet secara umum sulit didapatkan. Parameter yang digunakan pada *instance* ini adalah sebagai berikut:

    - Jumlah responden pada setiap segmen/blok sensus adalah 10 rumah tangga
    - Rata-rata wawancara pada setiap rumah tangga 71,58 menit dengan standar deviasi 12,16 menit.
    - Rata-rata waktu yang diperlukan untuk mendapatkan koneksi internet sebesar 60 menit dengan standar deviasi 3 menit.

6. Pengujian dengan kode *instance* d06 menggambarkan kondisi dimana rata-rata anggota rumah tangga sedikit tetapi memiliki variasi tinggi, dengan koneksi internet secara umum sulit didapatkan. Parameter yang digunakan pada *instance* ini adalah sebagai berikut:

    - Jumlah responden pada setiap segmen/blok sensus adalah 10 rumah tangga
    - Rata-rata wawancara pada setiap rumah tangga 27,58 menit dengan standar deviasi 25,16 menit.
    - Rata-rata waktu yang diperlukan untuk mendapatkan koneksi internet sebesar 60 menit dengan standar deviasi 3 menit.

7. Pengujian dengan kode *instance* d07 menggambarkan kondisi dimana jarak antar rumah tangga jauh, rata-rata anggota rumah tangga banyak dan memiliki

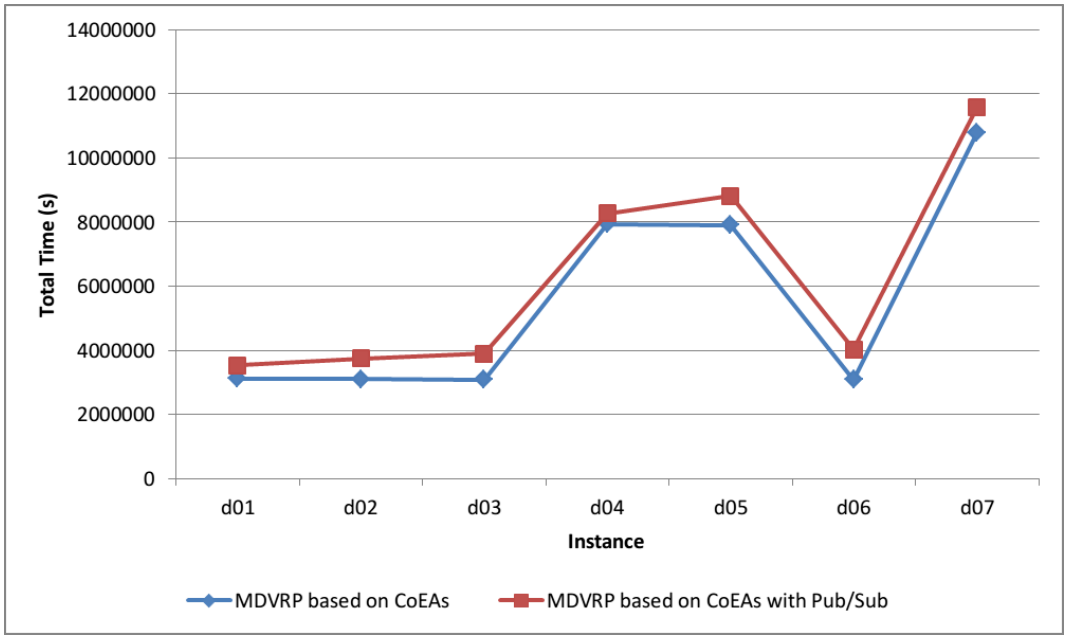
variasi tinggi, dan koneksi internet secara umum sulit didapatkan. Parameter yang digunakan pada *instance* ini adalah sebagai berikut:

- Jumlah responden pada setiap segmen/blok sensus adalah 10 rumah tangga
- Rata-rata wawancara pada setiap rumah tangga 71,58 menit dengan standar deviasi 25,65 menit.
- Rata-rata waktu tempuh antar rumah tangga 30,45 menit dengan standar deviasi 22,12 menit.
- Rata-rata waktu yang diperlukan untuk mendapatkan koneksi internet sebesar 60 menit dengan standar deviasi 3 menit.

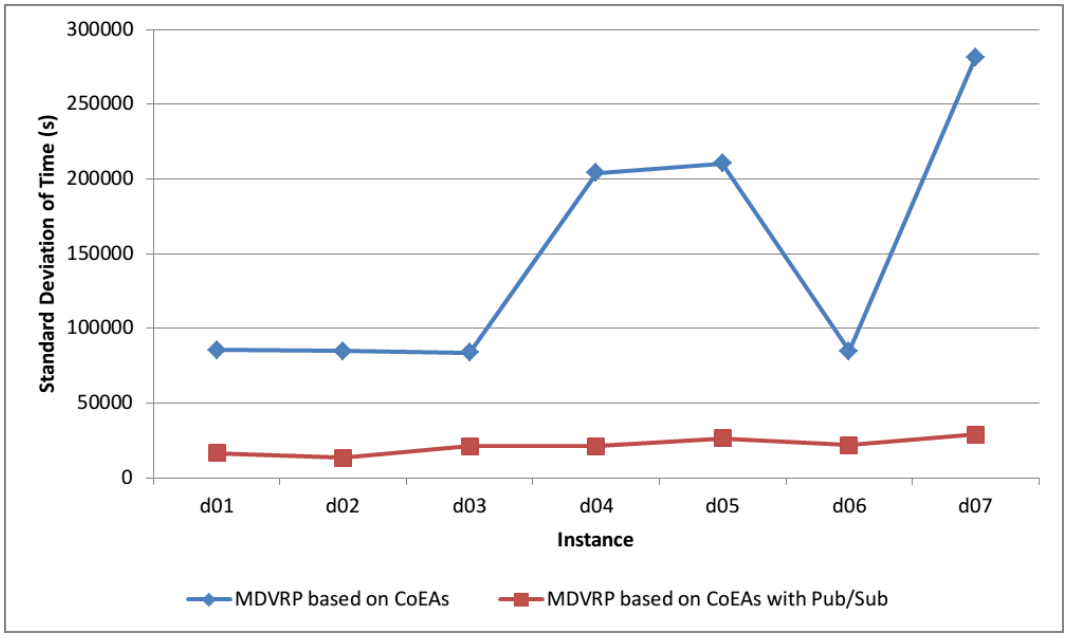
Dari hasil simulasi, diperoleh rute untuk masing-masing *instance* sebagaimana digambarkan pada Lampiran 2. Kemudian dari seluruh rute yang diperoleh, dikalkulasi *total time* untuk masing-masing rute, sebagaimana cara yang dijelaskan pada subsubsection 5.2.2.2. Berdasarkan Table 5.4, diperoleh hasil bahwasannya dengan menggunakan sistem usulan keseluruhan *instance* menghasilkan **waktu total** yang lebih besar. Sementara dari sisi **standar deviasi** menunjukkan hasil 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 5.7 dan Figure 5.8 menggambarkan perbandingan hasil pengujian dari 7 *instance*.

Tabel 5.4: Hasil Pengujian Kondisi Delay Dengan *Service Time* pada Data Lapangan

Instance	Total Waktu CoES MDVRP (det)	Total Waktu CoES MDVRP + Pub/Sub (det)	Stdev Waktu CoES MDVRP (det)	Stdev Waktu CoES MDVRP + Pub/Sub (det)
d01	**3,126,361.44**	3,532,199.80	85,427.10	**16,565.69**
d02	**3,107,775.85**	3,743,549.71	84,873.08	**13,445.49**
d03	**3,085,573.84**	3,901,417.79	83,869.48	**21,175.88**
d04	**7,937,588.57**	8,262,690.25	204,164.67	**21,147.45**
d05	**7,903,175.28**	8,820,620.43	210,500.85	**26,542.12**
d06	**3,083,461.58**	4,019,585.94	84,959.86	**22,071.07**
d07	**10,792,753.12**	11,590,204.96	280,996.99	**29,048.13**



Gambar 5.7: Perbandingan Waktu Total dari 7 *Instance*



Gambar 5.8: Perbandingan Standar Deviasi Waktu dari 7 *Instance*

5.2.3.7 Pengujian Kondisi Pencacah Berhenti dengan *Service Time*

Tabel 5.5: Hasil Pengujian Kondisi Normal Pada Data Cordeau Dengan *Service Time*

Instance	Total Waktu CoES MDVRP (det)	Total Waktu CoES MDVRP + Pub/Sub (det)	Stdev Waktu CoES MDVRP (det)	Stdev Waktu CoES MDVRP + Pub/Sub (det)
q01		3,464,733.76		37,997.03
q02		3,635,752.13		68,239.51
q03		3,611,394.03		98,900.33

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 hasil bahwasannya sistem usulan yang menggunakan algoritma MDVRP berbasis CoEAs yang dikombinasikan dengan mekanisme Publish/-Subscribe menghasilkan total waktu 80 persen lebih buruk dari program pembanding yang menggunakan algoritma MDVRP berbasis CoEAs tanpa mekanisme Publish/Subscribe. Sementara dari sisi *standar deviasi*, sistem usulan menghasilkan 80 persen lebih baik dari aplikasi pembanding. Dengan demikian dapat disimpulkan algoritma MDVRP berbasis CoEAs dengan mekanisme Publish/Subscribe menghasilkan total waktu yang lebih merata antar rute yang dihasilkan, sementara algoritma MDVRP berbasis CoEAs tanpa mekanisme Publish/Subscribe menghasilkan total waktu yang lebih efisien.

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 pembanding, 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 pembanding, tetapi 59,11 persen dari sisi standar deviasi. Pada pengujian kondisi normal, dapat disimpulkan algoritma MDVRP berbasis CoEAs dengan mekanisme Publish/Subscribe lebih sesuai digunakan pada permasalahan rekomendasi lokasi pencacahan karena menghasilkan total waktu yang lebih merata antar pencacah.

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. Pada pengujian kondisi dimana terdapat *delay* pada koneksi, dapat disimpulkan bahwa meskipun memiliki perbedaan yang tipis, algoritma MDVRP berbasis CoEAs

dengan mekanisme Publish/Subscribe masih lebih sesuai digunakan pada permasalahan rekomendasi lokasi pencacahan karena menghasilkan total waktu yang lebih merata antar pencacah.

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. Pada pengujian kondisi dimana terdapat *packet loss* pada koneksi, dapat disimpulkan bahwa algoritma MDVRP berbasis CoEAs dengan mekanisme Publish/Subscribe lebih sesuai digunakan pada permasalahan rekomendasi lokasi pencacahan karena menghasilkan total waktu yang lebih merata antar pencacah.

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. Mengkasi penggunaan algoritma MDVRP yang lain, misalnya Tabu Search, Particle Swarm Optimization, dan lain lain, sehingga diperoleh algoritma MDVRP yang paling sesuai.

DAFTAR PUSTAKA

Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., and Steggles, P. (1999). Towards a better understanding of context and context-awareness. In *International Symposium on Handheld and Ubiquitous Computing*, pages 304–307. Springer.

Atzmueller, M. and Hilgenberg, K. (2013). 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. ACM.

Averbakh, I. and Berman, O. (1994). Technical NoteRouting and Location-Routing p-Delivery Men Problems on a Path. *Transportation Science*, 28(2):162–166.

Averbakh, I. and Berman, O. (1995). Probabilistic Sales-Delivery Man and Sales-Delivery Facility Location Problems on a Tree. *Transportation Science*, 29(2):184–197.

Balakrishnan, A., Ward, J. E., and Wong, R. T. (1987). Integrated Facility Location and Vehicle Routing Models: Recent Work and Future Prospects. *American Journal of Mathematical and Management Sciences*, 7(1-2):35–61.

Baldacci, R., Toth, P., and Vigo, D. (2010). Exact algorithms for routing problems under vehicle capacity constraints. *Annals of Operations Research*, 175(1):213–245.

Banavar, G., Chandra, T., Mukherjee, B., Nagarajarao, J., Strom, R. E., and Sturman, D. C. (1999). 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.

Bao, X. and Roy Choudhury, R. (2010). 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.

Bookbinder, J. H. and Reece, K. E. (1988). Vehicle routing considerations in distribution system design. *European Journal of Operational Research*, 37(2):204–213.

BPS, B. (2016a). Badan Pusat Statistik - Tugas, Fungsi, dan Kewenangan.

BPS, B. (2016b). Sistem Rujukan Statistik BPS RI - Blok Sensus.

Branco, I. M. and Coelho, J. D. (1990). The hamiltonian p-median problem. *European Journal of Operational Research*, 47(1):86–95.

Brown, P. J. (1995). The stick-e document: a framework for creating context-aware applications. *Electronic Publishing-Chichester-*, 8:259–272.

Brown, P. J., Bovey, J. D., and Chen, X. (1997). Context-aware applications: from the laboratory to the marketplace. *IEEE personal communications*, 4(5):58–64.

Bruns, A. (1998). *Zweistufige Standortplanung unter Berucksichtigung von Tourenplanungsaspekten Primale Heuristiken und Lokale Suchverfahren*. PhD Thesis, Sankt Gallen University.

Cao, P. Y., Li, G., Chen, G., and Chen, B. (2015). 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. ACM.

Chao, I.-M., Golden, B. L., and Wasil, E. (1993). 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.

Chen, G., Kotz, D., and others (2000). A survey of context-aware mobile computing research. Technical report, Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College.

Chen, H., Finin, T., and Joshi, A. (2003). An intelligent broker architecture for context-aware systems. *PhD proposal in computer science, University of Maryland, Baltimore, USA*.

Cordeau, J.-F., Gendreau, M., and Laporte, G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2):105–119.

Cordeau, J.-F., Laporte, G., Savelsbergh, M. W. P., and Vigo, D. (2007). Chapter 6 Vehicle Routing. In Laporte, C. B. a. G., editor, *Handbooks in Operations Research and Management Science*, volume 14 of *Transportation*, pages 367–428. Elsevier. DOI: 10.1016/S0927-0507(06)14006-2.

Cordeau, J.-F. and Maischberger, M. (2012). A parallel iterated tabu search heuristic for vehicle routing problems. *Computers & Operations Research*, 39(9):2033–2050.

Creswell, J. W. (2013). *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications.

Dai, J., Bai, X., Yang, Z., Shen, Z., and Xuan, D. (2010a). PerFallD: 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.

Dai, J., Teng, J., Bai, X., Shen, Z., and Xuan, D. (2010b). Mobile phone based drunk driving detection. In *2010 4th International Conference on Pervasive Computing Technologies for Healthcare*, pages 1–8. IEEE.

Dantzig, G. B. and Ramser, J. H. (1959). The Truck Dispatching Problem. *Management Science*, 6(1):80–91.

de Oliveira, F. B., Enayatifar, R., Sadaei, H. J., Guimares, F. G., and Potvin, J.-Y. (2016). A cooperative coevolutionary algorithm for the Multi-Depot Vehicle Routing Problem. *Expert Systems with Applications*, 43:117–130.

Dey, A. K. (2001). Understanding and using context. *Personal and ubiquitous computing*, 5(1):4–7.

Diaz, B. D. (2012). *Vrp web*.

Do, T. M. T. and Gatica-Perez, D. (2011). Groupus: Smartphone proximity data and human interaction type mining. In *2011 15th Annual International Symposium on Wearable Computers*, pages 21–28. IEEE.

Dueck, G. (1993). New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational physics*, 104(1):86–92.

Escobar, J. W., Linfati, R., Toth, P., and Baldoquin, M. G. (2014). A hybrid Granular Tabu Search algorithm for the Multi-Depot Vehicle Routing Problem. *Journal of Heuristics*, 20(5):483–509.

Eugster, P. T., Felber, P. A., Guerraoui, R., and Kermarrec, A.-M. (2003). The many faces of publish/subscribe. *ACM computing surveys (CSUR)*, 35(2):114–131.

Garey, M. R. and Johnson, D. S. (2002). *Computers and intractability*, volume 29. wh freeman New York.

Gendreau, M., Hertz, A., and Laporte, G. (1992). New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 40(6):1086–1094.

Gillett, B. E. and Johnson, J. G. (1976). Multi-terminal vehicle-dispatch algorithm. *Omega*, 4(6):711–718.

Golden, B. L., Magnanti, T. L., and Nguyen, H. Q. (1977). Implementing vehicle routing algorithms. *Networks*, 7(2):113–148.

Google, G. (2016). Google Maps Directions API.

Gross, T. and Specht, M. (2001). Awareness in context-aware information systems. In *Mensch & Computer 2001*, pages 173–182. Springer.

Holland, J. H. and others (1990). Echo: Explorations of evolution in a minature world.

Hull, R., Neaves, P., and Bedford-Roberts, J. (1997). Towards situated computing. In *Digest of Papers. First International Symposium on Wearable Computers*, pages 146–153.

Huning, A., Rechenberg, I., and Eigen, M. (1976). *Evolutionsstrategie. Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. JSTOR.

jsprit (2014). jsprit | java toolkit for rich VRPs and TSPs.

Kulkarni, R. V. and Bhave, P. R. (1985). Integer programming formulations of vehicle routing problems. *European Journal of Operational Research*, 20(1):58–67.

Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European journal of operational research*, 59(3):345–358.

Laporte, G. and Nobert, Y. (1981). An exact algorithm for minimizing routing and operating costs in depot location. *European Journal of Operational Research*, 6(2):224–226.

Laporte, G., Nobert, Y., and Arpin, D. (1984). *Optimal solutions to capacitated multidepot vehicle routing problems*. Universit de Montral, Centre de recherche sur les transports.

Laporte, G., Nobert, Y., and Taillefer, S. (1988). Solving a Family of Multi-Depot Vehicle Routing and Location-Routing Problems. *Transportation Science*, 22(3):161–172.

Lau, H. C. W., Chan, T. M., Tsui, W. T., and Pang, W. K. (2010). Application of Genetic Algorithms to Solve the Multidepot Vehicle Routing Problem. *IEEE Transactions on Automation Science and Engineering*, 7(2):383–392.

Lin, S. (1965). Computer solutions of the traveling salesman problem. *The Bell System Technical Journal*, 44(10):2245–2269.

Lu, H., Pan, W., Lane, N. D., Choudhury, T., and Campbell, A. T. (2009). 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.

Magara, M. B., Ojo, S., Ngwira, S., and Zuva, T. (2016). 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.

Maranzana, F. E. (1964). On the Location of Supply Points to Minimize Transport Costs. *OR*, 15(3):261–270.

Mhl, G. (2002). *Large-scale content-based publish-subscribe systems*. PhD thesis, TU Darmstadt.

Nagy, G. and Salhi, S. (2007). Location-routing: Issues, models and methods. *European Journal of Operational Research*, 177(2):649–672.

OptaPlanner (2016). Constraint satisfaction solver (Java, Open Source).

Pisinger, D. and Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435.

Psaraftis, H. N. (1995). Dynamic vehicle routing: Status and prospects. *Annals of operations research*, 61(1):143–164.

Raft, O. M. (1982). A modular algorithm for an extended vehicle scheduling problem. *European Journal of Operational Research*, 11(1):67–76.

Rechenberg, I. (1965). Cybernetic solution path of an experimental problem.

redis (2017a). Clients - Redis.

redis (2017b). Introduction to Redis Redis.

Renaud, J., Boctor, F. F., and Laporte, G. (1996a). An improved petal heuristic for the vehicle routeing problem. *Journal of the operational Research Society*, 47(2):329–336.

Renaud, J., Laporte, G., and Boctor, F. F. (1996b). A tabu search heuristic for the multi-depot vehicle routing problem. *Computers & Operations Research*, 23(3):229–235.

Rubel, P., Fayn, J., Nollo, G., Assanelli, D., Li, B., Restier, L., Adami, S., Arod, S., Atoui, H., Ohlsson, M., and others (2005). Toward personal eHealth in cardiology. Results from the EPI-MEDICS telemedicine project. *Journal of electrocardiology*, 38(4):100–106.

Ryan, N., Pascoe, J., and Morse, D. (1999). Enhanced reality fieldwork: the context aware archaeological assistant. *Bar International Series*, 750:269–274.

Said, A., Berkovsky, S., and De Luca, E. W. (2013). Introduction to Special Section on CAMRa2010: Movie Recommendation in Context. *ACM Trans. Intell. Syst. Technol.*, 4(1):13:1–13:9.

Salhi, S. and Fraser, M. (1996). An Intergrated Heuristic Approach for the Combined Location Vehicle Fleet Mix Problem. *Studies in Locational Analysis*, 8:3–22.

Sarmenta, L. F., Chua, S. J., Echevarria, P., Mendoza, J. M., Santos, R.-R., Tan, S., and Lozada, R. (2002). Bayanihan Computing. NET: Grid Computing with XML Web Services. In *ccgrid*, pages 434–435.

Schilit, B., Adams, N., and Want, R. (1994). Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, pages 85–90. IEEE.

Schmidt, A., Beigl, M., and Gellersen, H.-W. (1999). There is more to context than location. *Computers & Graphics*, 23(6):893–901.

Schwefel, H.-P. (1975). *Evolutionsstrategie und numerische Optimierung*. Technische Universitt Berlin.

Sengoku, H. and Yoshihara, I. (1998). A fast TSP solver using GA on JAVA. In *Third International Symposium on Artificial Life, and Robotics (AROB III98)*, pages 283–288.

Simchi-Levi, D. (1991). The Capacitated Traveling Salesman Location Problem. *Transportation Science*, 25(1):9–18.

Solomon, M. M. and Desrosiers, J. (1988). Survey papertime window constrained routing and scheduling problems. *Transportation science*, 22(1):1–13.

Subramanian, A., Uchoa, E., and Ochi, L. S. (2013). A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research*, 40(10):2519–2531.

Sudman, S. (1965). Time Allocation in Survey Interviewing and in Other Field Occupations. *The Public Opinion Quarterly*, 29(4):638–648.

Tillman, F. A. (1969). The multiple terminal delivery problem with probabilistic demands. *Transportation Science*, 3(3):192–204.

Tillman, F. A. and Cain, T. M. (1972). An upperbound algorithm for the single and multiple terminal delivery problem. *Management Science*, 18(11):664–682.

Tillman, F. A. and Hering, R. W. (1971). A study of a look-ahead procedure for solving the multiterminal delivery problem. *Transportation Research*, 5(3):225–229.

Toth, P. and Vigo, D. (2002). The vehicle routing problem, volume 9 of SIAM Monographs on Discrete Mathematics and Applications. *SIAM, Philadelphia, PA*.

Tsai, M. J., Wu, C. L., Pradhan, S. K., Xie, Y., Li, T. Y., Fu, L. C., and Zeng, Y. C. (2016). 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.

Vaishnavi, V. and Kuechler, W. (2007). *Design Science Research Methods and Patterns: Innovating Information and Communication Technology*. Auerbach Publications. DOI: 10.1201/9781420059335.

Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. (2014). Implicit depot assignments and rotations in vehicle routing heuristics. *European Journal of Operational Research*, 237(1):15–28.

Weise, T., Podlich, A., and Gorldt, C. (2009). Solving real-world vehicle routing problems with evolutionary algorithms. In *Natural intelligence for scheduling, planning and packing problems*, pages 29–53. Springer.

Wren, A. and Holliday, A. (1972). 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.

Zimmermann, A., Lorenz, A., and Oppermann, R. (2007). An operational definition of context. In *International and Interdisciplinary Conference on Modeling and Using Context*, pages 558–571. Springer.

Zou, X., Gonzales, M., and Saeedi, S. (2016). A Context-aware Recommendation System using smartphone sensors. In *2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 1–6.

LAMPIRAN