TESIS – TI 185401

PEMODELAN PENENTUAN RUTE TRUK TANGKI UNTUK OPTIMALISASI PERMASALAHAN PENYIRAMAN TAMAN KOTA DI SURABAYA

Nuskha Ilma Arini
02411650032006

DOSEN PEMBIMBING
Nurhadi Siswanto, S.T., M.S.I.E., Ph.D.
Dr.Eng.Ir. Ahmad Rusdiansyah, M.Eng., CSCP

Departemen Teknik Sistem dan Industri
Fakultas Teknologi Industri dan Rekayasa Sistem
Institut Teknologi Sepuluh Nopember
2020

TESIS – TI185401

PEMODELAN PENENTUAN RUTE TRUK TANGKI UNTUK OPTIMALISASI PERMASALAHAN PENYIRAMAN TAMAN KOTA DI SURABAYA

NUSKHA ILMA ARINI
02411650032006

Dosen Pembimbing
Nurhadi Siswanto S.T., MSIE.,Ph.D
Dr.Eng.Ir. Ahmad Rusdiansyah, M.Eng.,CSCP

Departemen Teknik Sistem dan Industri
Fakultas Teknologi Industri dan Rekayasa Sistem
Institut Teknologi Sepuluh Nopember
2020

TESIS – TI185401

MODELING OF DETERMINATION OF TANK TRUCK ROUTE FOR OPTIMIZATION OF CITY PARK WATERING PROBLEMS IN SURABAYA

NUSKHA ILMA ARINI
02411650032006

Dosen Pembimbing
Nurhadi Siswanto S.T., MSIE.,Ph.D
Dr.Eng.Ir. Ahmad Rusdiansyah, M.Eng.,CSCP

Departement Of Systems and Industrial Engineering
Faculty Of Industrial Technology and Systems Engineering
Institut Teknologi Sepuluh Nopember
2020

LEMBAR PENGESAHAN TESIS

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

NUSKHA ILMA ARINI
NRP: 02411650032006

Tanggal Ujian : 23 Januari 2020
Periode Wisuda: Maret 2020

Disetujui oleh:
Pembimbing:

1. Nurhadi Siswanto, ST., M.S.I.E., Ph.D ..... NIP: 197005231996011001

2. Dr. Eng. Ir. Ahmad Rusdiansyah, M.Eng., CSCP ..... NIP: 196811091995031003

Penguji:

1. Prof. Dr. Ir. Budisantoso Wirjodirdjo, M.Eng ..... NIP: 195503081979031001

2. Prof. Dr. Ir. Udisubakti Ciptomulyono, M.Eng.Sc ..... NIP: 195903181987011001

Kepala Departemen Teknik Sistem dan Industri
Fakultas Teknologi Industri dan Rekayasa Sistem

Nurhadi Siswanto, ST., M.S.I.E., Ph.D
NIP: 197005231996011001

LEMBAR PERNYATAAN KEASLIAN TESIS

Saya yang bertanda tangan di bawah ini:

Nama : Nuskha Ilma Arini
NRP : 02411650032006
Program Studi : Magister Teknik Industri - ITS

Menyatakan bahwa tesis dengan judul

"PEMODELAN PENENTUAN RUTE TRUK TANGKI UNTUK OPTIMALISASI PERMASALAHAN PENYIRAMAN TAMAN KOTA DI SURABAYA"

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

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, Januari 2020
Yang membuat pernyataan

Nuskha Ilma Arini
NRP. 02411650032006

PEMODELAN PENENTUAN RUTE TRUK TANGKI UNTUK OPTIMALISASI PERMASALAHAN PENYIRAMAN TAMAN KOTA DI SURABAYA

Nama Mahasiswa : Nuskha Ilma Arini
NRP : 02411650032006
Pembimbing : Nurhadi Siswanto, S.T., M.S.I.E., Ph.D.
Co Pembimbing : Dr. Eng. Ir. Ahmad Rusdiansyah, M.Eng., CSCP, CLTD

ABSTRAK

Berdasarkan UU No 26 Tahun 2007, proporsi minimal Ruang Terbuka Hijau (RTH) adalah 30% dari luas wilayah kota yang terdiri dari 20% RTH publik dan 10% RTH privat. Untuk menjalankan peraturan tersebut, Pemerintah Kota Surabaya sedang giat menambah jumlah RTH publik dengan membuat taman kota dan dikelola oleh Dinas Kebersihan dan Ruang Terbuka Hijau (DKRTH) Kota Surabaya. Dengan jumlah RTH yang semakin besar, maka dibutuhkan perawatan dan pemeliharaan, salah satunya adalah dengan melakukan penyiraman yang rutin dilakukan setiap hari. Permasalahan yang sering dihadapi oleh DKRTH Surabaya adalah menentukan rute truk yang dapat mengoptimalkan waktu tempuh. Banyak variabel yang dapat mempengaruhi, antara lain *demand*, jumlah kendaraan, kapasitas kendaraan, *time windows*, jam kerja operasi dan rute angkut kendaraan. Namun pada praktiknya, banyaknya jumlah lokasi penyiraman, rute dan penugasan area *service* yang tidak merata seringkali mengakibatkan *overtime*. *Overtime* ini berimbas pada jam kerja *shift* berikutnya.

Penugasan dan penentuan rute pada truk tangki penyiraman termasuk dalam sebuah permasalahan NP-*hard*, yaitu sulit diselesaikan dengan menggunakan metode konvensional dan membutuhkan waktu komputasi yang lama. Permasalahan optimasi rute truk penyiraman secara matematis termasuk dalam *Vehicle Routing Problem* (VRP). Prinsip dasar VRP berkaitan dengan kunjungan setiap titik hanya dilakukan satu kali dirasa kurang cocok untuk menyelesaikan permasalahan ini, sehingga perlu dimodifikasi dengan tambahan *split service* agar pembagian tugas antar kendaraan merata. Sehingga memenuhi batasan *constraint*, terutama permasalahan batasan waktu (*time windows*).

Untuk dapat menyelesaikan suatu permasalahan pada proses penjadwalan dan penentuan rute dapat menggunakan metode optimasi dengan pendekatan metaheuristik menggunakan algoritma *Ant Colony Optimization*. Tujuan dari penelitian ini adalah meminimasi waktu total penyiraman dan menghasilkan model yang terkait dengan jumlah minimum kendaraan dan rute dalam melakukan proses pengisian dan penyiraman dari depot menuju ke taman.

Penelitian ini membangun model dengan empat skenario dalam proses penyiraman taman. Skenario pertama dengan menggunakan *split service* untuk melakukan penyiraman taman, skenario kedua menggunakan *split service* dan *saving value*, skenario ketiga menggunakan *saving value* tanpa *split service* (*Hard No Split*), dan skenario keempat menggunakan *saving value* tanpa *split service* (*Soft No Split*). Keempat skenario memberikan rekomendasi penggunaan 8 truk dari 9 truk yang tersedia. Jika dilihat dari fungsi tujuan, skenario 4 lebih unggul dari sisi total waktu tempuh dan menghasilkan penghematan sebesar 31,18% dari total waktu tempuh rute *existing*.

Kata kunci: *Manajemen Transportasi, Vehicle Routing Problem, Ant Colony Optimization*

(Halaman ini sengaja dikosongkan)

MODELING OF DETERMINATION OF TANK TRUCK ROUTE FOR OPTIMIZATION OF CITY PARK WATERING PROBLEMS IN SURABAYA

Name	: Nuskha Ilma Arini
NRP	: 02411650032006
Supervisor	: Nurhadi Siswanto, S.T., M.S.I.E., Ph.D.
Co Supervisor	: Dr. Eng. Ir. Ahmad Rusdiansyah, M.Eng., CSCP, CLTD

ABSTRACT

Based on Law No. 26/2007, the minimum proportion of Green Open Space is 30% of the city area consisting of 20% of public green open space and 10% private green open space. To carry out these regulations, the Surabaya City Government is actively increasing the number of public green space by creating city parks and is managed by Dinas Kbersihan dan Ruang Terbuka Hijau (DKRTH) Surabaya. With the ever-increasing amount of green open space, care and maintenance are needed, one of which is to do routine watering every day. The problem often faced by DKRTH Surabaya is determining truck routes that can optimize travel time. Many variables can affect, among others, demand, number of vehicles, vehicle capacity, time windows, operating hours and vehicle transport routes. But in practice, a large number of watering locations, irregular routes, and uneven service area assignments often result in overtime. This overtime impacts the next shift work hours.

Assigning and determining the route on the watering tank truck is included in an NP-hard problem, which is difficult to solve using conventional methods and requires a long computational time. Problems with optimizing the route of the watering truck mathematically included in the Vehicle Routing Problem (VRP). The basic principle of VRP about visiting each point only once is not suitable to solve this problem, so it needs to be modified with the addition of split service so that the division of tasks between vehicles is evenly distributed. So that it meets the constraints, especially the problem of time constraints (time windows).

To be able to solve a problem in the process of scheduling and determining routes can use the optimization method with a metaheuristic approach using the Ant Colony Optimization algorithm. The purpose of this study is to minimize the total watering time and produce a model related to the minimum number of vehicles and routes in the process of filling and watering from the depot to the park.

This research builds a model with four scenarios in the process of watering the park. The first scenario uses split service to do garden watering, the second scenario uses split service and saving value, the third scenario uses saving value without split service (Hard No Split), and the fourth scenario uses saving value without split service (Soft No Split). The four scenarios provide recommendations for using 8 of the 9 available trucks. If viewed from the objective function, scenario 4 is the best in terms of total travel time and results in savings of 31.18% of the total travel time of existing routes.

Keywords: *Transportation Management, Vehicle Routing Problem, Ant Colony*

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT atas limpahan rahmat dan rezeki-Nya sehingga penulis dapat menyelsaikan penulisan Tesis yang berjudul "Pemodelan Penentuan Rute Truk Tangki Untuk Optimalisasi Permasalahan Penyiraman Taman Kota Di Surabaya" dengan baik dan tepat waktu. Tak lupa juga shalawat serta salam tercurahkan kepadan junjungan Nabi Muhammad SAW yang telah menyampaikan petunjuk kepada umatnya.

Laporan tesis ini diajukan sebagai syarat untuk menyelesaikan studi Strata 2 (S2) di Jurusan Teknik Industri. Selama pelaksanaan dan penyusunan tesis ini, penulis telah menerima bantuan dari berbagai pihak. Oleh sebab itu, pada kesempatan ini, penulis tidak lupa mengucapkan terima kasih dan peghargaan kepada:

1. Allah SWT yang telah memberikan kemudahan, kelancaran dan keyakinan kepada penulis, bahwa penulis mampu menyelesaikan Tesis ini;
2. Kedua orang tua penulis yaitu Bapak HM. Yunus Thohir dan Ibu Khusnul Khotimah, kakak penulis yaitu Mohammad Helmi Nur Fikri, S.Pd, serta adik penulis yaitu Moh. Iqbal Zamzamy , yang senantiasa mendoakan dan memotivasi penulis;
3. Bapak Nurhadi Siswanto, S.T., M.S.I.E., Ph.D dan Bapak Dr. Eng. Ir. Ahmad Rusdiansyah, M.Eng., CSCP, CLTD selaku dosen pembimbing penulis yang telah memberikan pengarahan, bimbingan serta dukungan penuh selama penulis menyelesaikan Tesis ini;
4. Bapak Prof. Dr. Ir. Budi Santosa, M. Sc. dan Ibu Dyah Santhi Dewi, ST., M.Eng.Sc. selaku penguji Seminar Proposal penulis untuk penelitian ini yang telah memberikan saran-saran perbaikan serta bimbingan yang mendukung penelitian ini;
5. Bapak Prof. Dr. Ir. Budisantoso Wirjodirdjo, M. Eng. dan Bapak Prof. Dr. Ir. Udisubakti Ciptomulyono, M. Eng.Sc selaku penguji Seminar Hasil penulis untuk penelitian ini yang telah memberikan saran-saran perbaikan serta bimbingan yang mendukung penelitian ini;
6. Dinas Kebersihan dan Ruang Terbuka Hijau yang telah memberikan kesempatan penulis untuk melakukan penelitian ini dan memberikan bimbingan serta arahan yang dibutuhkan oleh penulis;
7. Calon pendamping penulis yang telah senantiasa memberikan semangat dan dukungan penuh demi terselesaikannya penelitian ini dengan baik.
8. Teman – teman Pasca Sarjana Teknik Industri Program Magister dan Doktoral Teknik Industri ITS, yang telah memberikan bantuan dan memotivasi penulis dalam menyelesaikan Tesis ini.

Dalam penulisan Tesis ini, penulis merasa masih banyak kekurangan pada teknis penulisan dan materi laporan. Untuk itu, kritik dan saran dari semua pihak sangat diharapkan demi penyempurnaan pembuatan Tesis ini. Penulis berharap semoga Tesis ini dapat bermanfaat bagi objek amatan dan rekan – rekan di Teknik Industri ITS pada khususnya.

Surabaya, Januari 2020

Penulis

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

BAB 1
PENDAHULUAN

Pada bab pendahuluan ini akan dijelaskan mengenai dasar-dasar dalam melakukan penelitian yaitu latar belakang permasalahan, identifikasi permasalahan penelitian, tujuan penelitian, manfaat penelitian, ruang lingkup penelitian dan sistematika penulisan penelitian.

1.1 Latar Belakang

Berdasarkan UU No 26 Tahun 2007, proporsi minimal Ruang Terbuka Hijau (RTH) adalah 30% dari luas wilayah kota yang terdiri dari 20% RTH publik dan 10% RTH privat. Untuk menjalankan peraturan tersebut, Pemerintah Kota Surabaya telah membuat rencana pengaturan RTH dalam Rencana Detil Tata Ruang (RDTR) Kota Surabaya yang memiliki jangka waktu pelaksanaan selama 20 tahun. Selama tahun 2014-2015, Pemerintah Kota Surabaya sedang giat memperbaiki dan menambah jumlah RTH publik dengan membuat taman kota. Taman kota ini kemudian dikelola oleh masyarakat dan Dinas Kebersihan dan Ruang Terbuka Hijau (DKRTH) Kota Surabaya.

Pengelolaan yang dilakukan oleh DKRTH Kota Surabaya salah satunya adalah penyiraman yang rutin dilakukan setiap hari. Penyiraman dilakukan dengan menggunakan dua cara, yaitu dengan *sprinkle* dan truk tangki. *Sprinkle* digunakan untuk penyiraman taman aktif, sedangkan truk tangki digunakan untuk penyiraman taman pasif. Taman aktif adalah taman yang memiliki fungsi sebagai tempat bermain yang dilengkapi elemen-elemen pendukung taman bermain, misalnya seperti Taman Bungkul dan Taman Apsari. Lain halnya dengan taman pasif yang hanya dilengkapi elemen estetis saja dan berfungsi sebagai ruang terbuka hijau, misalnya pulau jalan di Jalan Kertajaya.

Menurut Peraturan Menteri Perhubungan Republik Indonesia Nomor PM 34 Tahun 2014 Tentang Marka Jalan, pulau jalan (atau bisa disebut juga pulau lalu lintas) merupakan bagian jalan yang tidak dapat dilalui oleh kendaraan, dapat berupa marka jalan atau bagian jalan yang ditinggikan dan biasanya ditanami pepohonan sebagai ruang terbuka hijau. Pulau jalan berfungsi untuk meningkatkan keselamatan lalu lintas pada ruas jalan ataupun di persimpangan jalan melalui pemisahan arus.

(a)

(b)

Gambar 1. 1 (a) Taman Pasif (b) Taman Aktif

Menurut data Dinas Kebersihan dan Ruang Terbuka Hijau (DKRTH) Kota Surabaya terdapat 70 taman kota dan 400 titik ruang terbuka hijau termasuk pulau jalan. Dengan banyaknya jumlah tersebut, maka pengelolaan taman pasif oleh DKRTH Surabaya dibagi menjadi per rayon. Terdapat 5 rayon pertamanan, yaitu Rayon Pusat, Rayon Barat, Rayon Timur, Rayon Utara, dan Rayon Selatan. Rayon Timur memiliki wilayah penyiraman paling luas yang meliputi 9 kecamatan, antara lain Gunung Anyar, Rungkut, Tenggilis, Sukolilo, Keputih, Mulyorejo, Kalijudan, Tambaksari, dan Gubeng dengan masing-masing kecamatan dialokasikan 1 kendaraan, sehingga kendaraan yang dimiliki Rayon Timur berjumlah 9 truk dengan kapasitas 5000 Liter. Untuk kegiatan operasional penyiraman taman, terdapat 2 *shift* kerja yaitu *shift* pagi dan *shift* malam. *Shift* pagi dimulai pada pukul 06.00 – 15.00 WIB, sedangkan untuk *shift* malam mulai dari pukul 15.00-23.00 WIB.

Truk tangki berangkat dari DKRTH Surabaya Rayon Timur yang berada di Jl. Tenggilis Tengah No 1 Surabaya dan akan kembali ke lokasi yang sama jika waktu *shift* berakhir. Truk tangki berangkat dari Kantor DKRTH Rayon Timur dalam keadaan kosong menuju lokasi penyiraman. Sebelum sampai di lokasi, dilakukan pengisian tangki di sungai atau selokan yang memenuhi kriteria air untuk penyiraman. Pengisian tangki dilakukan di sungai atau selokan yang terdekat dari lokasi penyiraman.

Gambar 1. 2 Proses Pengisian Tangki

Setelah mengisi air, truk tangki akan menyiram taman sesuai rute yang telah ditentukan oleh DKRTH Surabaya Rayon Timur. Truk tangki akan menyiram sepanjang jalan hingga seluruh ruas di sepanjang jalur pulau jalan tercukupi kebutuhan airnya. Proses penyiraman dilakukan oleh dua pekerja. Satu orang akan bertugas sebagai pengemudi truk tangki dan satu orang lainnya akan melakukan penyiraman dengan menggunakan selang seperti yang ditunjukan pada Gambar 1.2.

Gambar 1. 3 Proses Penyiraman Taman Pasif

Permasalahan yang sering dihadapi oleh DKRTH Surabaya Rayon Timur dalam melaksanakan kegiatan penyiraman antara lain mengoptimalkan waktu operasional dalam memenuhi kebutuhan air di semua taman. Banyak variabel yang dapat mempengaruhi jumlah kendaraan yang digunakan supaya dapat memenuhi kebutuhan penyiraman tanaman pada masing-masing titik. Kota Surabaya sementara ini memiliki 70 taman kota dan 400 titik ruang terbuka hijau termasuk pulau jalan masing-masing memiliki *demand* yang berbeda. Selain *demand* pada beberapa titik terdapat variabel lainnya misalnya: jumlah kendaraan, kapasitas kendaraan, batasan waktu *shift*, jam kerja operasional dan rute angkut kendaraan. Namun pada praktiknya, banyaknya jumlah lokasi penyiraman, rute yang tidak teratur, dan penugasan area *service* yang tidak merata

seringkali mengakibatkan *overtime* atau kendaraan terlambat pulang ke kantor rayon. *Overtime* ini mengakibatkan terganggunya *shift* kerja berikutnya (*shift* sore) karena alokasi kendaraan yang tersedia hanya 1 kendaraan per kecamatan. Sedangkan pada masing-masing kecamatan mempunyai beban kerja yang tidak rata, dimana terdapat wilayah yang memiliki area ruang terbuka hijau yang luas dan sebaliknya ada yang wilayah ruang terbuka hijaunya sedikit. Sehingga perlu dilakukan pembagian beban kerja kembali dengan parameter pembagian berdasarkan luasan taman dan bukan lagi berdasarkan kecamatan, agar semua wilayah mempunyai beban kerja yang seimbang. Dari deskripsi sebelumnya menggambarkan betapa kompleks permasalahan dalam penentuan jumlah kendaraan sehingga perlu adanya optimasi dengan metode tertentu yang dapat membantu menyelesaikan permasalahan tersebut.

Penjadwalan pada truk tangki penyiraman termasuk dalam sebuah permasalahan NP-*hard*, yaitu sulit diselesaikan dengan menggunakan metode konvensional dan waktu komputasi yang dibutuhkan untuk mendapatkan solusi yang optimal dan meningkat eksponensial seiring besarnya suatu permasalahan. Proses penjadwalan truk berusaha mengalokasikan sumber daya yang dimiliki oleh perusahaan dalam kasus ini jumlah truk yang dibutuhkan ke dalam suatu kegiatan penyiraman taman memiliki sebuah batasan. Batasan-batasan ini dikelompokkan menjadi dua yaitu, *hard constraint* dan *soft constraint. Hard constraint* memiliki prioritas lebih tinggi dibandingkan dengan *soft constraint.* Permasalahan pada penjadwalan terjadi di bagian transportasi DKRTH Kota Surabaya Rayon Timur, dimana masalah penjadwalan ini berkaitan dengan *time windows* dan kapasitas truk.

Permasalahan optimasi rute truk penyiraman secara matematis termasuk dalam permasalahan yang disebut *Vehicle Routing Problem* (VRP). Prinsip dasar VRP berkaitan dengan kunjungan setiap titik hanya dilakukan satu kali dirasa kurang cocok untuk menyelesaikan permasalahan ini, sehingga perlu dimodifikasi dengan tambahan *split service* agar pembagian tugas antar kendaraan merata. Sehingga memenuhi batasan *constraint*, terutama permasalahan batasan waktu (time windows). Oleh karena itu untuk menyelesaikan permasalahan pada penelitian ini dilakukan pengembangan model *Split Delivery Vehicle Routing Problem with Time Windows (SDVRPTW).* Bianchessi (2017) mengembangkan model VRP yang melayani pengiriman secara terpisah (*split*), untuk menghasilkan *saving cost* pada total *travelling cost.* Hal ini bisa diterapkan pula untuk meminimasi total *travelling time* yang ada permasalahan ini.

Untuk dapat menyelesaikan suatu permasalahan pada proses penjadwalan dan penentuan rute dapat menggunakan metode optimasi dengan pendekatan heuristik. Akan tetapi hal tersebut sulit jika menggunakan cara manual untuk menyelesaikan permasalahan, karena akan memakan

banyak waktu dengan memperhitungkan banyaknya *constraint* yang harus diselesaikan. Penelitian terhadap penjadwalan dan penentuan rute sebelumnya telah dilakukan dengan menggunakan metode *eksak*, dimana pada penelitian tersebut telah dihasilkan sebuah rute yang memenuhi seluruh *constraint* yang telah ditentukan. Akan tetapi penelitian tersebut menggunakan beberapa asumsi yang membuat hasil tidak optimal secara total, dan juga proses *running* dari metode eksak tersebut membutuhkan waktu yang lama. Sedangkan pada penelitian kali ini akan digunakan metode *Ant Colony Optimization* (ACO) yang sebelumnya telah dikembangkan oleh (Dorigo M. a., 1999) yang terinspirasi dari karakteristik perilaku serangga sosial, yaitu koloni semut. Salah satu perilaku yang menarik dari koloni semut adalah kemampuannya berkoordinasi dalam memgumpulkan makanan. Semut pekerja bertugas untuk mencari makan dan mencari sumber makanan secara acak. Ketika sumber makanan ditemukan, semut tersebut akan membawa makanan kembali ke sarangnya. Dengan sebuah mekanisme tertentu, semut-semut yang lain akan mengetahui jalur untuk menguju ke sumber makanan tersebut untuk kemudian diangkut menuju sarangnya. Bahkan meskipun jalur yang telah tercipta diberi penghalang, semut-semut tersebut tetap mampu membuat jalur baru yang menghubungkan sumber makanan dengan sarangnya. Algoritma ACO termasuk dalam kategori *population based metaheuristic*, sehingga relatif dapat memberikan waktu komputasi lebih cepat dibandingkan *single based solution metaheuristic* karena pencarian solusi langsung berdasarkan populasi dan grup-grup tertentu.

Penelitian ini berfokus pada pengembangan model optimasi untuk penyiraman taman kota di Surabaya. Kendaraan melakukan *loading* di setiap taman dengan permintaan *deterministic* dan melakukan *unloading* pada sungai. Kendaraan akan berangkat dan kembali ke depot dalam keadaan muatan kosong. Dalam pencarian rute optimal yang memiliki waktu minimal, pendekatan yang digunakan adalah metode metaheuristik dengan algoritma ACO dengan mempertimbangkan banyaknya taman yang harus dikunjungi sebelum kembali ke depot dengan batasan waktu *shift*. Penggunaan algoritma ACO diharapkan dapat membantu pencarian nilai optimal dengan waktu komputasi yang lebih cepat.

1.2 Perumusan Masalah

Perumusan masalah dalam penelitian ini adalah bagaimana menentukan jadwal dan rute kendaraan yang optimal untuk melakukan penyiraman taman pasif di beberapa lokasi dengan mempertimbangkan batasan waktu dan kapasitas kendaraan untuk meminimasi waktu operasional *shift* pada semua rute kendaraan.

1.3 Tujuan Penelitian

Berikut ini adalah tujuan yang ingin dicapai dalam penelitian tesis ini.

1. Menentukan rute penyiraman baru untuk mempersingkat waktu total penyiraman dan mengestimasi waktu operasional kendaraan pada semua rute dalam satu hari untuk melayani semua taman.
2. Menghasilkan model yang dapat membantu dalam keputusan yang terkait dengan:
    a. Jumlah minimum kendaraan yang tersedia yang dapat melayani semua taman kota dalam satu kali operasional harian.
    b. Jadwal keberangkatan kendaraan pada rute tujuan dalam melakukan proses pengisian dan penyiraman dari depot menuju ke taman.

1.4 Manfaat Penelitian

Berikut ini adalah manfaat yang akan didapatkan dari penelitian tesis ini.

1 Memberikan solusi baru terkait rute penyiraman truk tangki dan mempersingkat total waktu kerja untuk melakukan penyiraman taman di Kota Surabaya

2 Memberikan estimasi mengenai jumlah kendaraan yang optimal untuk seluruh rute dalam melayani semua permintaan penyiraman taman dalam waktu operasional harian kendaraan.

1.5 Ruang Lingkup Penelitian

Ruang lingkup penelitian terbagi atas dua bagian yaitu batasan penelitian dan asumsi penelitian. Pada penelitian ini dilakukan batasan antara lain:

1. Lokasi yang digunakan dalam penelitian ini adalah lokasi yang dapat dilalui oleh truk tangki.
2. Penelitian ditujukan pada proses pengisian dan penyiraman taman pasif wilayah Rayon Timur pada hari kerja *shift* pagi.
3. Penjadwalan dan penentuan rute dilakukan pada musim kemarau

Berikut ini merupakan asumsi yang digunakan dalam penelitian tesis ini.

1. Diasumsikan pengisian tangki sebagai *pick up* dan penyiraman taman sebagai *delivery*.
2. Satu *node* merupakan titik akhir penyiraman dengan suatu luasan taman tertentu
3. Kecepatan kendaraan saat berpindah dari satu lokasi ke lokasi lain tergantung waktu sibuk

1.6 Sistematika Penelitian

Laporan penelitian ini terdiri dari tujuh bab. Penjelasan lebih rinci ketujuh bab tersebut diuraikan melalui sistematika penulisan sebagai berikut.

BAB I PENDAHULUAN

Pada Bab 1 Pendahuluan dijelaskan mengenai hal-hal yang mendasari dilakukannya penelitian serta identifikasi permasalahan penelitian yang meliputi latar belakang, perumusan masalah, tujuan penelitian, manfaat penelitian, serta ruang lingkup penelitian yang berisikan batasan dan asumsi yang digunakan dalam penyusunan penelitian ini.

BAB II TINJAUAN PUSTAKA

Pada Bab 2 Tinjuauan Pustaka berisi tentang penjelasan teori dari permasalahan yang diangkat dalam penelitian ini. Serta metode yang digunakan dalam melakukan penelitian yang diperoleh dari berbagai referensi yang digunakan sebagai landasan dalam penulisan penelitian ini. Hal yang dijelaskan pada bab ini adalah Manajemen Transportasi, *Vehicle Routing Problem*, Metaheuristik.

BAB III METODOLOGI PENELITIAN

Pada bab 3 Metodologi Penelitian akan dijelaskan secara rinci mengenai tahapan-tahapan yang dilakukan dalam melakukan penelitian tesis. Metodologi penelitian ini menggambarkan alur pelaksanaan penelitian dan kerangka berpikir yang digunakan peneliti selama pelaksanaan penelitian. Metodologi penelitian ini meliputi tahap identifikasi dan perumusan masalah, tahap pengumpulan data, tahap pengolahan data, tahap analisis dan interpretasi data, dan yang terakhir adalah tahap penarikan kesimpulan dan penyusunan saran.

BAB IV PENGUMPULAN DAN PENGOLAHAN DATA

Pada bab ini membahas mengenai data-data relevan yang diperoleh dan pengolahan data yang digunakan sebagai *input* pada model. Pengolahan data dijelaskan mengenai bagaimana data yang telah dikumpulkan diolah. Pada penelitian ini pengolahan data menggunakan bahasa pemrograman *Python* sebagai *model framework*-nya dan *Jupyter Notebook* sebagai *software excecutor* yang menjalankan program tersebut.

BAB V PENGOLAHAN DATA

Pada bab ini dibahas mengenai formulasi model, validasi dan verifikasi model, *running* model, dan analisis hasil *running* yang diperoleh serta analisis ketepatan dan keterbatasan model.

BAB VI KESIMPULAN DAN SARAN

Pada bab ini akan dilakukan penarikan kesimpulan dari hasil pelaksanaan penelitian sesuai dengan tujuan yang ingin dicapai serta saran-saran yang dapat diberikan untuk perbaikan pada penelitian selanjutnya.

BAB 2
TINJAUAN PUSTAKA

Pada bab tinjauan pustaka diuraikan teori, temuan dan bahan penelitian dari sumber lain yang akan dijadikan landasan untuk melakukan kegiatan penelitian. Uraian dalam tinjauan pustaka diarahkan untuk menyusun kerangka pemikiran atau konsep yang akan digunakan dalam penelitian ini.

2.1 Ruang Terbuka Hijau

Berdasarkan Undang-Undang nomor 26 tahun 2007 tentang penataan ruang, Ruang Terbuka Hijau (RTH) didefinisikan sebagai area memanjang atau jalur, yang penggunaannya lebih bersifat terbuka, tempat tumbuh tanaman, baik yang tumbuh secara alamiah maupun yang sengaja ditanam. Definisi lain dari ruang terbuka hijau adalah suatu pemanfaatan lahan pada satu kawasan yang diperuntukkan untuk penghijauan (Febrianti, N. & Sofan, P., 2014). RTH sebagai infrastruktur hijau perkotaan yang diisi oleh tumbuhan, tanaman, dan vegetasi.

Fungsi RTH menurut Permendagri Nomor 1 Tahun 2007 dapat berfungsi secara ekologis, sosial/budaya, arsitektural, dan ekonomi. Berkaitan dengan fungsi secara ekologi misalnya, ruang terbuka hijau berfungsi sebagai pengendali iklim yakni sebagai produsen oksigen, peredam kebisingan, dan juga berfungsi sebagai visual control / kontrol pandangan yaitu dengan menahan silau matahari atau pantulan sinar yang ditimbulkan. Adapun dalam aspek sosial budaya, salah satu fungsi dari ruang terbuka hijau (RTH) diantaranya adalah sebagai ruang komunikasi dan interaksi sosial bagi masyarakat. Hal ini dapat diwujudkan melalui RTH yang bersifat publik. Selain sebagai ruang interaksi masyarakat, RTH publik baiknya juga memenuhi fungsi sebagai sarana rekreasi, olahraga, sarana pendidikan, bahkan sebagai pusat kuliner. Selain kedua aspek tersebut, RTH juga dapat berfungsi secara estetika diantaranya meningkatkan kenyamanan, memperindah lingkungan kota, serta menstimulasi kreativitas dan produktivitas warga kota. Agar suatu RTH publik dapat berfungsi secara optimal, tentunya perlu diperhatikan pula apakah sudah memenuhi kriteria penyediaan sebagai ruang publik yang ideal seperti lokasi yang mudah dijangkau, nyaman, dan memberikan rasa aman bagi penggunanya. (Imansari, N. & Khadiyanta P., 2015)

Pengelolaan RTH merupakan kebijakan otonomi masing-masing daerah dengan proporsi luasan paling sedikit 30% dari luas wilayah kota yang terdiri dari 20% RTH publik dan 10% RTH privat. RTH publik merupakan RTH yang dimiliki dan dikelola oleh pemerintah daerah kota yang digunakan untuk kepentingan masyarakat secara umum. RTH publik meliputi taman kota, taman pemakaman umum, dan jalur hijau sepanjang jalan (pulau jalan), sungai, dan pantai. Sedangkan

ruang terbuka hijau privat meliputi kebun atau halaman rumah/gedung milik masyarakat/swasta yang ditanami tumbuhan.

Perawatan RTH dilakukan oleh Dinas Kebersihan dan Ruang Terbuka Hijau (DKRTH). Salah satu perawatan rutin yang dilakukan adalah penyiraman. Untuk penyiraman RTH terdapat 3 cara tergantung jenis dan lokasi RTH, yaitu menggunakan truk tangki penyiraman untuk pulau jalan, *sprinkle* untuk taman kota aktif, dan pompa alkon untuk taman di pinggiran sungai.

Gambar 2. 1 Penyiraman Ruang Terbuka Hijau

2.2 Transportasi

Transportasi adalah proses pemindahan barang (muatan) dan penumpang dari suatu tempat ke tempat lain dengan menggunakan wahan yang digerakkan oleh manusia atau mesin (Andriansyah, 2015). Sistem transportasi merupakan suatu bentuk keterikatan dan keterkaitan antara penumpang, barang, prasarana dan sarana yang berinteraksi dalam rangka perpindahan orang atau barang yang tercakup dalam suatu tatanan, baik secara alami maupun buatan/rekayasa (Hadihardaja dkk, 1997).

Tarif transportasi dipengaruhi berbagai faktor. Faktor utama yang mempengaruhi tarif transportasi adalah jarak (*distance*), berat (*weight*) dan densitas (*density*). Jarak merupakan faktor utama yang menetukan biaya transportasi. Umumnya biaya-biaya transportasi dipicu oleh jarak. Jarak transportasi akan berkontribusi secara langsung terhadap biaya varaiabel seperti tenaga sopir, biaya bahan bakar dan minyak (*fuel*) dan biaya pemeliharaan kendaraan (Zaroni,2015)

2.3 *Vehicle Routing Problem* (VRP)

Vehicle Routing problem (VRP) merupakan masalah penentuan rute kendaraan yang memegang peranan penting dalam dunia industri yaitu pada masalah manajemen logistik dan transportasi. VRP pertama kali diperkenalkan oleh Dantzig dan Ramser pada tahun 1959 dan semenjak itu telah dipelajari secara luas. Oleh Fisher, VRP didefinisikan sebagai sebuah cara pencarian atas penggunaan yang efisien dari sejumlah kendaraan yang harus melakukan perjalanan untuk mengunjungi sejumlah tempat untuk mengantar dan menjemput orang atau barang. Istilah konsumen menunjukkan pemberhentian untuk mengantar dan menjemput orang/barang. Setiap konsumen harus dilayani oleh satu kendaraan saja. Penentuan pasangan kendaraan-konsumen ini dilakukan dengan mempertimbangkan kapasitas kendaraan dalam satu

kali angkut, untuk meminimalkan biaya yang diperlukan. Biasanya penentuan biaya minimal erat kaitannya dengan jarak yang minimal. (Fisher, 1995).

VRP merupakan pengembangan dari *Travelling Salesman Problem* (TSP). TSP dapat dijelaskan sebagai suatu masalah dimana seorang *salesman* harus berangkat dari sebuah depot untuk mengunjungi n *node*/kota kemudian kembali ke depot semula dengan memilih *feasible tour* yang terpendek. Tujuan dari TSP ini adalah mencari biaya per mulasi atau tour minimum dari semua kota sehingga mendapatkan rute dengan lintasan minimum atau meminimumkan biaya. Dengan kata lain mendesain suatu rute perjalanan terpendek dimana setiap *node* harus dikunjungi oleh *salesman* tersebut. (Balas, 2001)

Jika ada *m salesman* pada TSP, maka kita sebut masalah tersebut sebagai m-TSP. Prinsipnya sama dengan TSP, seperti halnya TSP pada m-TSP setiap *node* hanya boleh dikunjungi tepat satu kali dan oleh satu orang *salesman* saja. Tiap *salesman* harus memulai dan mengakhiri tour nya pada depot yang sama, sehingga jika ada m *salesman* maka pasti ada m rute tour yang berbeda pula. (Raden Prana A, 2007).

VRP dapat ditranformasi dengan dua cara. Transformasi pertama adalah dengan cara memperluas dasar prinsip pengepakan struktur BPP (*Bin Packing Problem*). Transformasi kedua dengan berlandaskan dasar perutean pada struktur TSP (*Travelling Salesman Problem*). (Raden Prana A, 2007).

BPP dapat dideskripsikan sebagai sejumlah angka yang melambangkan ukuran dari sejumlah item dan sebuah konstanta K yang melambangkan kapasitas dari bin. Jumlah bin minimum yang diperlukan adalah satu item hanya dapat berada dalam satu bin saja dan total kapasitas item pada setiap bin tidak boleh melebihi kapasitas dari bin tersebut. Disamping itu, TSP adalah sebuah permasalahan tentang seorang *salesman* yang ingin mengunjungi tiap kota sekali saja, dimulai dan diakhiri dari kota awal. Inti permasalahan adalah untuk menentukan jalur terpendek melalui semua kota yang ada. Hubungan keduanya dengan VRP adalah, vehicle dapat dihubungkan dengan customer menggunakan BPP, dan urutan kunjungan vehicle terhadap tiap customer diselesaikan menggunakan TSP. (Falkenauer, 1996)

VRP merupakan sebuah cakupan masalah yang didalamnya terdapat sejumlah rute untuk sejumlah kendaraan yang berada pada satu atau lebih depot yang harus ditentukan jumlahnya agar tersebar secara geografis supaya bisa melayani konsumen-konsumen yang tersebar. Setiap kendaraan memiliki kapasitas angkut, dan setiap pelanggan memiliki *demand*. Tujuan umum VRP menurut Toth dan Vigo (2002) adalah: meminimalkan jarak dan biaya tetap yang berhubungan dengan penggunaan kendaraan, meminimalkan jumlah kendaraan yang dibutuhkan untuk melayani permintaan seluruh pelanggan, menyeimbangkan rute-rute dalam hal waktu perjalanan

dan muatan kendaraan, dan meminimalkan pinalti sebagai akibat dari pelayanan yang kurang memuaskan terhadap pelanggan, seperti keterlambatan pengiriman dan lain sebagainya. Pada umumnya, batasan yang terdapat dalam VRP adalah kapasitas kendaraan, waktu pengiriman dan pemenuhan permintaan pelanggan.

Berikut ini merupakan komponen-komponen yang terdapat pada *Vehicle Routing Problem* (VRP) :

a. Jaringan Kerja (*Link*)

Dalam transportasi pada suatu rute, setiap jalan yang tersedia merupakan jaringan kerja (link) dan setiap lokasi merupakan *node*. Link dapat dijalani dalam satu arah (directed) atau dua arah (undirected). Setiap link berkaitan dengan panjang atau waktu perjalanan, jenis kendaraan dan periode waktu perjalanan yang dilakukan pada link tersebut sehingga link dapat dikatakan berhubungan dengan biaya.

b. Konsumen (*Customer)*

Karakteristik khusus dari *customers* adalah sebagai berikut:

1. Jumlah permintaan (*demand*) dari *customers* berbeda-beda, ada *customers* yang jumlah permintaannya diketahui secara pasti (deterministik) tetapi ada juga jumlah permintaannya tidak pasti (stokastik).
2. Ada *customers* yang mempunyai *time windows*, yaitu periode waktu yang menunjukkan jangka waktu *customers* dapat dilayani.

c. Depot

Depot merupakan awal dan akhir dari suatu rute yang akan dilewati oleh kendaraan dalam melakukan pengiriman barang ke *customers*. Setiap depot dicirikan berdasarkan tipe dan banyak kendaraan yang berkaitan dengan depot tersebut serta banyaknya barang yang tersedia disana.

d. Kendaraan (*vehicle*)

Karakteristik khusus dari kendaraan (*vehicle*) adalah sebagai berikut:

1. Mempunyai kapasitas kendaraan maksimum (berat dan volume maksimum) dalam mengangkut barang.
2. Mempunyai total waktu kerja dari awal keberangkatan dari depot sampai kembali ke depot, sesuai peraturan yang diberlakukan oleh perusahaan untuk jam kerja pengemudi (waktu *loading*) dan sejumlah periode waktu yang tidak ikut diperhitungkan (waktu *non-loading*), misalnya waktu istirahat pengemudi.
3. Memerlukan biaya untuk melakukan pengiriman, biaya penggunaan kendaraan dihitung berdasarkan per unit jarak, per unit waktu, dan per rute.

e. Pengemudi (*driver*)

Pengemudi yang mengoperasikan kendaraan harus memenuhi semua kendala yang ditetapkan dalam kontrak kerja dan aturan dari perusahaan. (Dorigo L. M., 2000)

Penelitian VRP mengalami perkembangan hingga diklasifikasikan dalam berbagai variasi sesuai dengan berbagai kendala atau batasan seperti kapasitas, jarak, *time window*, *pick up* dan *delivery* dan lainnya. Berbagai metode juga dipakai untuk memecahkan VRP dan variasinya. Berikut ini terdapat beberapa jenis atau variasi masalah utama dalam VRP menurut Toth dan Vigo (2002).

1. *Capacitated Vehicle Routing Problem* (CVRP)

   CVRP merupakan jenis VRP yang setiap kendaraannya memiliki kapasitas terbatas.

2. *Distance Constrained Vehicle Routing Problem* (DCVRP)

   DCVRP merupakan jenis VRP dengan kendala batasan panjang rute.

3. *Vehicle Routing Problem with Pick up and Delivery* (VRPPD)

   VRPPD merupakan jenis VRP dengan pelayanan jemput dan pelayanan antar dalam setiap permintaan pelanggan.

4. *Vehicle Routing Problem with Multiple Depot* (MDVRP)

   MDVRP merupakan jenis VRP yang memiliki banyak depot dalam melakukan pelayanan terhadap pelanggan.

5. *Split Delivery Vehicle Routing Problem* (SDVRP)

   SDVRP merupakan jenis VRP dimana pelayanan terhadap pelanggan dilakukan dengan menggunakan kendaraan yang berbeda-beda.

6. *Vehicle Routing Problem with Time Windows* (VRPTW)

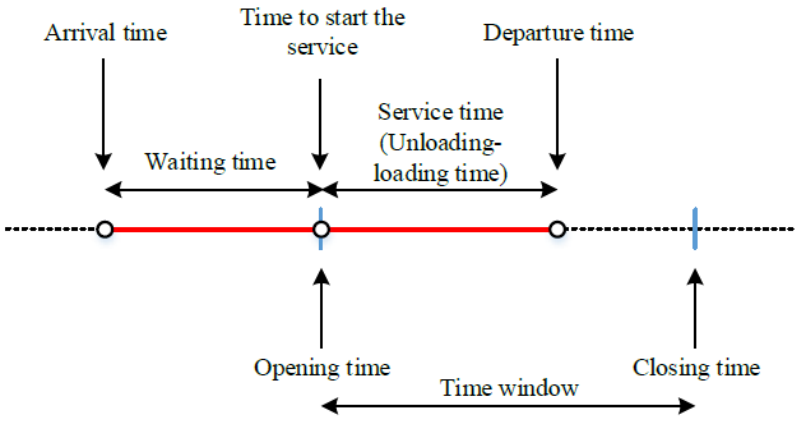
   VRPTW merupakan jenis VRP dengan kendala kapasitas kendaraan dan batasan waktu (*time windows*) pada setiap pelanggan dan depot.

2.3.1 *Vehicle Routing Problem with Time Windows* (VRPTW)

Setiap pelanggan yang memiliki batasan *time window* menunjukkan waktu buka dan tutup ketika diasumsikan bahwa *time window* konstan dan lebih besar dari atau sama dengan waktu layanan. Jika kendaraan tiba di pelanggan tertentu sebelum waktu bukanya, maka kendaraan harus menunggu dan layanan (*loading unloading*) dimulai pada waktu bukanya. Waktu keberangkatan kendaraan pada pelanggan harus kurang dari atau sama dengan waktu tutup. Gambar 2.1 menunjukkan ilustrasi ketika kendaraan tiba di pelanggan sebelum waktu pembukaannya (Suprayogi & Priyandari, 2017).

Pada VRPTW pelayanan setiap pelanggan harus dilayani dalah suatu interval waktu yang disebut dengan *time window*. *Time window* bisa menjadi *hard* dan *soft*, *hard* apabila pelanggan

datang terlalu awal maka pelanggan perlu menunggu hingga jam pelayanan buka, waktu menunggu tidak dikenakan biaya. Sedangkan untuk *soft* setiap pelanggaran akan mendapatkan biaya pinalti.



Gambar 2. 2 Ilustrasi Kedatangan Berdasarkan *Time Window*
(Sumber: Suprayogi & Priyandari, 2017)

Sebuah *time window* $[a_0, b_0] = [a_{n+1}, b_{n+1}]$ dimana a_0 dan b_0 waktu paling awal berangkat dari depot dan paling akhir kembali ke depot. Pada sbuah *arc node* $(i, j) \in A$ dapat dihilangkan dengan pertimbangan *time window* $a_i + s_i + t_{ij} > b_j$ atau dengan batasan kapasitas $q_i + q_j > Q$ atau dnegan faktor lainnya. Dalam pertimbangan *time window* waktu awal pelayanan pada suatu *node* ditambahkan *service time* pada *node* tersebut ditambahkan dengan waktu perjalanan dari *node* tersebut menuju ke *node* selanjutnya harus lebih besar dari b_j diluar itu maka *node* tidak dapat dikunjungi (Toth & Vigo, 2002).

Menurut (Guy Desaulniers, 2002) formulasi *Vehicle Routing Problem* with *Pickup and Delivery* adalah sebagai berikut:

Fungsi Objektif

Minimasi

$$min \sum_{k \epsilon K} \sum_{(i,j) \epsilon A_k} C_{ijk} X_{ijk} \qquad (2.1)$$

Batasan :

$$\sum_{k \epsilon K} \sum_{j \epsilon N_k \cup \{d(k)\}} X_{ijk} = 1 \qquad \forall\, i \in P, \qquad (2.2)$$

$$\sum_{j \epsilon N_k} x_{ijk} - \sum_{j \epsilon N_k} x_{j,n+i,k} = 0 \qquad \forall\, k \in K, i \in P_k, \qquad (2.3)$$

$$\sum_{j \epsilon P_k \cup \{d(k)\}} x_{o(k),j,k} = 1 \qquad \forall\, k \in K, \qquad (2.4)$$

$$\sum_{i \epsilon N_k \cup \{o(k)\}} x_{ijk} - \sum_{i \epsilon N_k \cup \{d(k)\}} x_{jik} = 0 \qquad \forall\, k \in K, j \epsilon N_k, \qquad (2.5)$$

$$\sum_{i \epsilon D_k \cup \{o(k)\}} x_{i.d(k).k} = 1 \qquad \forall\, k \in K, \qquad (2.6)$$

(2.5)

(2.6)

$$X_{ijk}\left(T_{ik}+s_i+t_{ijk}-T_{jk}\right)\leq 0 \qquad \forall\, k\in K,(i,j)\in A_k, \qquad (2.7)$$

$$a_i\leq T_{ik}\leq b_i \qquad \forall\, k\in K, i\in V_k, \qquad (2.8)$$

$$T_{ik}+t_{i,n+i,k}\leq T_{n+i,k} \qquad \forall\, k\in K,(i,j)\in P_k, \qquad (2.9)$$

$$X_{ijk}\left(L_{ik}+l_j-L_{jk}\right)\leq 0 \qquad \forall\, k\in K,(i,j)\in A_k, \qquad (2.10)$$

$$l_i\leq L_{ik}\leq C_k \qquad \forall\, k\in K, i\in P_k, \qquad (2.11)$$

$$0\leq L_{n+i,k}\leq C_k-l_i \qquad \forall\, k\in K, n+i\in D_k, \qquad (2.12)$$

$$L_{o(k),k}=0 \qquad \forall\, k\in K, \qquad (2.13)$$

$$X_{ijk}\geq 0 \qquad \forall\, k\in K,(i,j)\in A_k, \qquad (2.14)$$

$$X_{ijk}\ binary \qquad \forall\, k\in K,(i,j)\in A_k, \qquad (2.15)$$

Dengan :

1. *Set Index*

   i = {1,2,...,n};*node index*

   j = {1,2,...,n}; *node index*

   k = {1,2,..,n};*route index*

2. Parameter

   m = jumlah rute kendaraan

   n = jumlah *node*

   Q = kapasitas kendaraan

   L = beban kendaraan

3. Variabel Keputusan

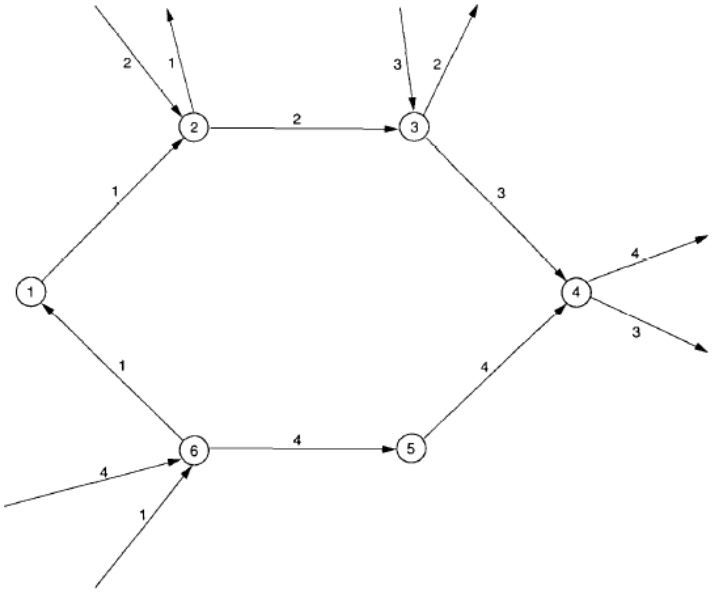
   x_{ijk} = 1 ketika *arc* (i,j) dilalui oleh rute k; 0 sebaliknya

   L_{ik} = variabel yang melambangkan jumlah beban yang diantar dari *node i* pada rute k

Berdasarkan formulasi tersebut, fungsi obyektifnya untuk meminimalkan total travel cost. Batasan (2.2) dan (2.3) bertujuan agar permintaan setiap *customer* dilayani sekali oleh satu kendaraan. Batasan (2.4) – (2.6) memastikan bahwa setiap kendaraan k mulai berangkat dari depot asal o(k) dan berakhir di depot tujuan d(k). Batasan (2.7) memastikan kompatibilitas antara rute dan jadwal, sedangkan batasan (2.8) merupakan batasan *time windows*. Untuk setiap permintaan, batasan (2.9) memastikan kendaraan mengunjungi *node pickup* sebelum *node delivery*. Batasan (2.10) menyatakan persyaratan kompatibilitas antara rute dan beban kendaraan, sedangkan batasan (2.11)-(2.12) membentuk interval kapasitas kendaraan yang bergantung pada *pickup* dan *delivery*. Sehingga beban awal diasumsikan oleh batasan (2.13). Persyaratan non negatif dan biner diberikan oleh kendala (2.14) dan (2.15).

2.3.2 *Vehicle Routing Problem with Split Service* (VRPTW)

Menurut Toth & Vigo (2002) hingga saat ini, kebanyakan VRP diasumsikan bahwa satu *node* dilayanani oleh satu kendaraan dalam satu operasi layanan, dimana layanan tersebut tidak dibagi dengan kendaraan lain. Namun kenyataannya, ada dua alasan untuk membagi beberapa layanan: Di satu sisi, jika permintaan melebihi kapasitas kendaraan, lebih dari satu kunjungan tidak dapat dihindari. Di sisi lain, membagi layanan menjadi beberapa permintaan layanan yang lebih kecil dapat menghasilkan penghematan biaya yang signifikan. Sehingga, beberapa penelitian berusaha menghapuskan asumsi tersebut dan menjadi *Vehicle Routing Split Delivery* (VRPSD). Gambar 2.3 merupakan salah satu contoh ilustrasi dari *split service* VRP. Pada gambar tersebut terdapat 4 *split cycles* yang artinya terdapat 4 *node* dimana *service* pada *node* tersebut tidak hanya dilayani dalam 1 kali kunjungan bahkan dengan kendaraan yang berbeda.



Gambar 2. 3 4 *Split Cycle*
(Sumber: Dror, *et al*., 1994)

2.4 Metaheuristik

Metaheuristik merupakan metode lanjut (*advanced*) berbasis heuristik untuk menyelesaikan permasalahan optimasi secara efisien (Talbi, 2009). Metaheuristic didefinisikan sebagai metode optimasi yang dilakukan dengan memperbaiki kandidat penyelesaian secara iterative sesuai dengan fungsi objektifnya. Menurut Blum dan Roli (Blum, 2003) metaheuristic memiliki beberapa karakteristik dasar yaitu:

a) Metaheuristik adalah strategi yang memandu proses pencarian

b) Tujuan dari metaheuristik adalah untuk menjelajahi ruang pencarian secara efisien untuk menemukan solusi optimal
c) Teknik metaheuristik berkisar dari prosedur pencarian local yang sederhana sampai proses pembelajaran yang kompleks
d) Metaheuristik dapat terdiri dari penggabungan beberapa mekanisme supaya proses pencarian tidak terjebak dalam daerah terbatas di ruang pencarian
e) Konsep dasar dari metaheuristik memungkinkan pendeskripsian secara abstrak
f) Metaheuristik bersifat general/umum sehingga dapat diterapkan dalam berbagai macam persoalan
g) Metaheuristik dapat menggunakan pengalaman yang didapat selama proses pencarian untuk menuntun proses pencarian

Dalam menentukan apakah metaheuristic adalah metode yang sesuai untuk menyelesaikan permasalahan, ada beberapa hal yang perlu diperhatikan misalnya kompleksitas permasalahan, ukuran input, struktur input, dan waktu yang diperlukan untuk menyelesaikan masalah tersebut.

Berbagai macam metode metaheuistik yang ada memiliki kesamaan dalam prosedurnya. Secara umum prosedur metaheuristic terdiri dari tiga langkah (Purnomo, 2014):

1. Langkah 1: inisialisasi

   Langkah ini meliputi inisialisasai permasalahan dan inisisalisasi algoritma. Di dalam inisialisasi permasalahan, variable keputusan yang dipakai harus bisa mempresentasikan permasalahan secara tepat. Inisialisisasi algoritma meliputi pengaturan variable keputusan di dalam algoritma yang dipakai, misalnya: jenis operasi/manipulasi yang dipakai (seleksi *roulette wheel*, mutasi random), konstanta yang dipakai dalam operasi / manipulasi (*mutation rate, crossover rate*), ukuran populasi dan kriteria untuk menghentikan proses pencarian.
2. Langkah 2: manipulasi penyelesaian

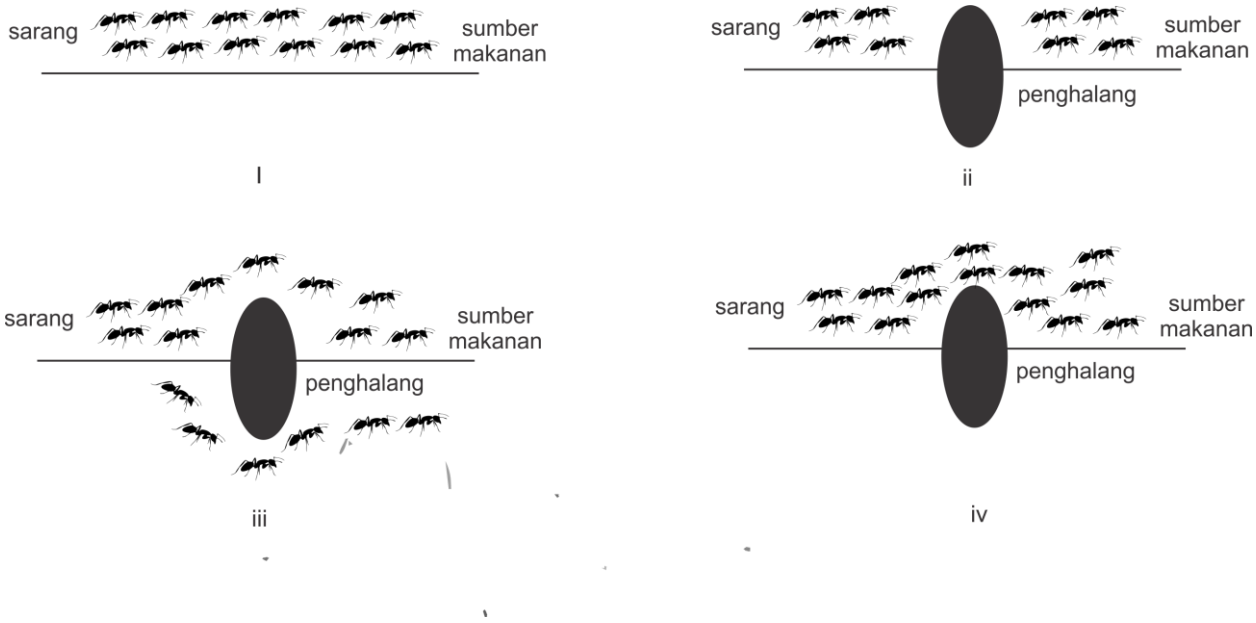
   Manipulasi penyelesaian merupakan inti dari metode metaheuristic. Perbedaan manipulasi inilah yang membedakan antara metode satu dengan metode lainnya. Teknik manipulasi ini banyak dipengaruhi oleh:

   a) Proses evolusi, misalnya algoritma genetika dan *evolution algorithm*
   b) Perilaku hewan, misalnya *ant colony optimization* dan *partical swam optimization*
   c) Proses fisika, misalnya *simulated annealing*
   d) Improvisasi musik, misalnya *harmony search*
3. Langkah 3: memperbarui / *update*

Di dalam metode metaheuristik, kandidat penyelesaian terbaik disimpan dan hanya diganti apabila di dalam iterasi berikutnya di dapatkan kandidat yang lebih baik atau memenuhi kriteria tertentu.

2.4.1 *Ant Colony Optimization*

Algoritma *Ant Colony Optimization* (ACO) diperkenalkan oleh Moyson dan Manderick dan secara meluas dikembangkan oleh Marco Dorigo. Algoritma ACO merupakan salah satu *swarm intelligence*, yang terinspirasi dari karakteristik perilaku serangga sosial, yaitu koloni semut. Salah satu perilaku yang menarik dari koloni semut adalah kemampuannya berkoordinasi dalam memgumpulkan makanan. Semut pekerja bertugas untuk mencari makan. Para semut pekerja akan mencari sumber makanan secara acak. Ketika sumber makanan ditemukan, semut tersebut akan membawa makanan kembali ke sarangnya. Dengan sebuah mekanisme tertentu, semut-semut yang lain akan mengetahui jalur untuk menguju ke sumber makanan tersebut untuk kemudian diangkut menuju sarangnya. Bahkan meskipun jalur yang telah tercipta diberi penghalang, semut-semut tersebut tetap mampu membuat jalur baru yang menghubungkan sumber makanan dengan sarangnya.



Gambar 2. 4 Ilustrasi perjalanan koloni semut mencari makanan
(Sumber: Dorigo, *et al*., 2000)

Untuk mengidentifikasi jalur kembali ke sarangnya maupun menuju ke sumber makanan, semut menggunakan suatu metode komunikasi tertentu yaitu *pheromone*. *Pheromone* merupakan suatu zat kimia yang dikeluarkan oleh semut yang berfungsi sebagai sinyal bagi semut yang lainnya. Setelah menemukan makanan, seekor semut akan mengeluarkan *pheromone* sambil bergerak menuju sarangnya. *Pheromone* inilah yang digunakan sebagai penanda jalur menuju

sumber makanan. Ketika semut lain menemukan *pheromone* tersebut, semut tersebut akan mengikuti *pheromone trail*, sebuah jalur yang telah ditandai dengan *pheromone*. Jika semut ini juga sampai ke sumber makanan, maka ia juga akan menaruh *pheromone* di jalur ketika kembali ke sarang, yang memperkuat *pheromone* yang lama. *Pheromone* akan menguap seiring berjalannya waktu, yang akan mengurangi kekuatannya dalam menarik semut-semut lain untuk mengikuti *pheromone trail* tersebut. Semakin sedikit semut yang meliwati jalur tersebut, maka semakin banyak bagian *pheromone* yang menguap. Tetapi jika semakin banyak semut yang lewat, maka semut-semut yang lewat tersebut akan menaruh *pheromone* sehingga jumlah *pheromone* di jalur tersebut akan semakin banyak.

Pada metode ACO, proses penguapan (evaporasi) dan deposit *pheromone* merupakan komponen yang penting dalam menyeimbangkan intensifikasi dan diversifikasi. Apabila tidak ada penguapan, maka jalur yang pertama kali diberi *pheromone* akan dengan cepat menarik semut-semut lain agar melewati jalur tersebut dan menyebabkan jalur yang lain terabaikan. Namun sebaliknya, jika penguapan terlau cepat, maka semut-semut yang lain akan kesulitan untuk mengidentifikasi jalur yang diberi *pheromone*, sehingga proses pencariannya menjadi acak.

Secara garis besar, prosedur ACO terdiri dari dua bagian, yaitu pembentukan solusi dan *pheromone update*. *Pheromone update* bertujuan untuk melakukan pencarian solusi yang lebih baik di dalam area pencarian dengan mengubah konsentrasi *pheromone* yang ada pada jalur jika mendapatkan solusi yang baik. Berikut merupakan langkah-langkah algoritma ACO untuk kasus TSP (Santosa, B. & Ai, T. J., 2017):

1. Langkah 1

   Asumsikan jumlah semut sejumlah N. Tentukan jumlah *pheromone* awal τ_{ij}^{1} yang sama untuk semua ruas antar kota. Agar lebih mudah untuk iterasi pertama nilai $\tau_{ij}^{1} = 1$ untuk semua ruas ij. Tentukan *visibility* antar kota sebagai dengan *invers* jarak $\frac{1}{jarak}$ dari masing-masing kota ke kota yang lain atau dengan menggunakan persamaan 2.16. Set iterasi t = 1.

   $$h = \frac{1}{jarak} \qquad (2.16)$$

2. Langkah 2

   a. Hitung probabilitas (p_{ij}) untuk memilih ruas atau nilai diskret x_{ij} menggunakan

   $$p_k(r,s) = \begin{cases} \tau(r,s)^{\alpha}\eta(r,s)^{\beta} / \sum_{j \in M_k} \tau(r,u)^{\alpha}\eta(r,u)^{\beta} \\ 0 \end{cases} \qquad (2.17)$$

   jika $s \in M^k$ dan 0 untuk lainnya.

   Nilai α merupakan bobot *pheromone* τ dan β adalah bobot yang mengontrol *visibility* terhadap tingkat *pheromone* τ.

b. Ruas tertentu akan dipilih oleh ruas *k* berdasarkan bilangan random dalam *range* (0,1). Untuk itu perlu juga menentukan *range* probabilitas komulatif yang berkaitan dengan pilihan ruas. Jadi jika ada *p* kemungkinan nilai variabel, maka akan ada *p* pilihan *range* probabilitas. Ruas yang mempunyai nilai perkalian τ dan η yang besar mempunyai peluang besar untuk terpilih.

3. Langkah 3

a. Bangkitkan bilangan random r dalam range (0,1) satu untuk setiap semut untuk setiap ruas yang akan dipilih.. Tentukan nilai diskret yang mewakili ruas untuk semut *k* dengan menggunakan bilangan randm dari langkah sebelumnya dan area probabilitis komulatif. Setiap semut akan menjalani rute tertentu.

b. Evaluasi nilai fungsi tujuan dengan cara menghitung jarak total setiap rute oleh semut k, f_k, $k = 1,2, \dots, N$. Tentukan lintasan terbaik diantara *N* ruas atau lintasan yang sudah dipilih oleh semut-semut yang berbeda dengan persamaan

$f_{best} = (_k = 1,2,..,N^{min}\{f_k\}$2.15)

Gunakan jarak total yang ditempuh oleh setiap semut untuk menghitung nilai Persamaan 2.18

$$\Delta\tau(r,s) = \frac{1}{f_k} \qquad (2.18)$$

$\Delta\tau(r,s)$ adalah *pheromone* yang ditambahkan pada setiap ruas yang dilalui seekor semut.

4. Langkah 4

Uji konvergensi dari proses. Dalam hal ini konvergensi dapat diartikan jika semua semut mengambil lintasan terbaik yang sama. Jika belum konvergen, koloni semut akan kembali ke sarang dan memulai pencarian makanan lagi. Set iterasi, $t = t + 1$, dan *update pheromone* untuk setiap ruas dengan

$$\tau_{ij}^{(t)} = \tau_{ij}^{old} + \sum_k \Delta\tau^{(k)} \qquad (2.19)$$

Dimana τ_{ij}^{old} menyatakan jumlah *pheromoene* dari iterasi sebelumnya yang tertinggal setelah penguapan

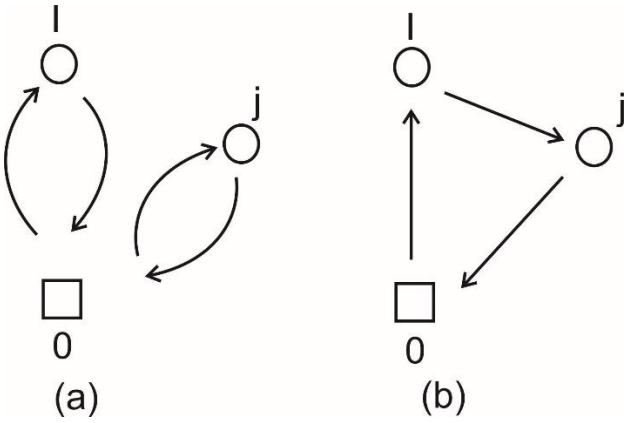
$$\tau_{ij}^{old} = (1-\rho)\tau_{ij}^{t-1} \qquad (2.20)$$

$\Delta\tau^k$ jumlah *pheromone* yang ditambahkan oleh semut terbaik *k* pada ruasnya dan dijumlahkan untuk semua semut yang menempuh ruas yang sama (jika ada lebih dari satu semut menempuh lintasan yang sama). $\rho \in (0,1)$ adalah parameter tingkat penguapan atau evaporasi. Dengan menggunakan nilai $\tau_{ij}^{(t)}$lanjutkan ke langkah 2. Langkah 2,3 dan 4

diulang sampai proses konvergen tercapai atau berhenti di sejumlah iterasi sebelum optimal tercapai.

2.5 Algoritma *Saving Value*

Algoritma *Saving Value* diperkenalkan oleh Clarke, G. & Wright, J. W. , (1964). Algoritma *saving* dirancang untuk menyelesaikan masalah rute kendaraan. Tujuan utamanya adalah menemukan solusi untuk meminimalkan total *cost* kendaraan. Dasar dari konsep *saving* ini untuk mendapatkan penghematan biaya dengan menggabungkan dua rute menjadi satu (Doyuran, 2011), seperti yang digambarkan pada Gambar 2.3.



Gambar 2.2. Ilustrasi *saving value*
(Sumber: Doyuran, 2011)

Gambar 2.3 (a) titik i dan j dikunjungi dengan rute terpisah. Oleh karena itu, dilakukan penghematan dengan dilakukan kunjungan ke titik i dan j sekaligus di rute yang sama seperti yang terlihat di gambar 2.3 (b). Rute kendaraan yang ditunjukkan diantara titik i dan j oleh c_{ij}, rute kendaraan D_a apada gambar 2.3 (a) yaitu:

$$D_a = c_{oi} + c_{io} + c_{oj} + c_{jo} \quad (2.21)$$

Ekuivalen dengan rute kendaraan D_b pada gambar 2.3 (b) adalah

$$D_a = c_{oi} + c_{ij} + c_{jo} \quad (2.22)$$

Dengan menggabungkan kedua rute diatas, maka diperoleh penghematan S_{ij}

$$S_{ij} = c_{oi} + c_{io} + c_{oj} + c_{jo} - (c_{oi} + c_{ij} + c_{jo}) \quad (2.23)$$

$$S_{ij} = c_{oi} + c_{oj} - c_{ij} \quad (2.24)$$

c_{oi} = jarak dari titik i ke depot

c_{oj} = jarak dari depot ke titik i

c_{ij} = jarak dari titik i ke titik j

s_{ij} = penghematan jarak dari titik i ke titik j

2.6 Posisi Penelitian

Untuk mengetahui perkembangan penelitian yang sejenis dan terkait dengan permasalahan yang dibahas pada penelitian ini akan dibahas dalam sub bab ini. Review penelitian terdahulu yang nantinya dapat diketahui posisi dan perbedaan dari penelitian sebelumnya. Permasalahan dalam penelitian ini berkaitan dengan permasalahan *vehicle routing problem* dengan *pick up and delivery*, *split service* dan *time window*.

Penelitian untuk pencarian rute yang terkait dengan *pick up* dan *deliveries* dilakukan oleh Desaulnierset al (2002) terkait pemenuhan permintaan customer terhadap pengiriman maupun penjemputan barang dalam satu kali perjalanan. Pembahasan permasalahan ini berkembang dengan penyelesaian menggunakan pendekatan metaheuristik penelitian yang dilakukan oleh Wassan, N. (2014). Pada penelitian tersebut dikembangkan beberapa varian *VRP Delivery and Pickup* (VRPDP) berdasarkan variasi *demand* yaitu *demand* tunggal dan kombinasi menggunakan pendekatan *Tabu Search*.

Pembahasan mengenai VRPDP berkembang seiring dengan munculnya permasalahan dan *constraint* baru yang ingin diselesaikan, salah satunya adalah mengenai *constraint* kunjungan ke masing-masing *node*. Dimana pada dasarnya kunjungan ke setiap *node* dilakukan tepat sekali. Namun dalam beberapa kasus, *node* bisa dikunjungi lebih dari satu kali (*split service*) apabila terdapat penghematan, baik jarak, waktu, biaya, dan jumlah rute yang signifikan dibandingkan dengan tanpa opsi *split service*. Hal ini dibahas dalam satu bab buku yang ditulis oleh Toth, P. (2014) mengenai multi kunjungan *node* (*multiple visit*) dengan yang disebut *Split Delivery Vehicle Routing Problem with Time Windows* (SDVRPTW). Penelitian serupa dilakukan oleh Bianchessi, N. and Irnich, S., (2016) menggunakan algoritma Branch and Cut untuk menyelesaikan permasalahan SDVRPTW yang menghasilkan output 5% lebih optimal dibandingkan dengan SDVRPTW dengan perhitungan eksak. Penelitian lainnya mengenai SDVRPTW juga dilakukan oleh Rajappa, G. P., Wilck, J. H., & Bell, J. E. (2016) yang membandingkan penyelesaian permasalahan SDVRPTW menggunakan *Ant Colony Optimization* (ACO) dan *Hybrid Metaheuristic Algorithm* yang merupakan gabungan dari ACO, *Genetic Algorithm* (GA), dan heuristik. Pada penelitian tersebut menguji ketiga algoritma metaheuristik pada beberapa dataset yang digunakan di beberapa penelitian sebelumnya yaitu Derigs et al. (2010) dan Jin et al. (2008). Dari hasil pengujian tersebut didapat hasil yang cukup menjanjikan untuk ACO, dimana 14 dari 21 dataset yang digunakan dalam penelitian Derigs et al. menghasilkan nilai yang lebih baik daripada 2 algoritma lainnya. Sedangkan untuk dataset dalam penelitian Jin et al., ACO menghasilkan nilai yang paling baik untuk semua dataset.

Sedangkan penelitian untuk penyiraman taman sendiri dilakukan oleh Maya R. (2016) terkait penentuan rute tangki penyiraman menggunakan VRP yang diselesaikan dengan metode eksak yaitu *nearest neighbour*. Pembahasan permasalahan ini berkembang dengan penyelesaian menggunakan pendekatan metaheuristik pada penelitian yang dilakukan oleh Viga A. (2017) dengan menggunakan pendekatan *Artificial Immune System* (AIS). Dalam penelitian tersebut terdapat 2 tahapan yang dilakukan, yaitu *route construction - route minimization* menggunakan *Solomon Insertion Heuristic* yang dijalnjutkan dengan prosedur *ejection pool* dan pada tahapan *distance improvement* total jarak diminimasi menggunakan AIS.

Tabel 2. 1 Penelitian Terdahulu

Tahun	Penulis	Judul	Tujuan	Kesimpulan	Future Research
2002	Desaulniers et al	*VRP with Pickup and Delivery*	Mendefinisikan dan mengklasifikasikan macam-macam model VRP Pickup and Delivery	Perkembangan lapangan dan tingkat kecanggihan metodologi VRPPD telah paralel dengan varian routing lainnya. Sebagai contoh, pendekatan heuristik. Berbagai prosedur peningkatan pencarian lokal sering diusulkan. Sementara metode yang lebih rumit, seperti metaheuristik, telah dikembangkan dari waktu ke waktu, namun belum menghasilkan manfaat yang sama seperti varian VRP lainnya.	Mempertimbangkan perkembangan infrastruktur telekomunikasi dan informasi pada aspek dinamis dari VRPPD.
2003	Gabor Nagy	*Heuristic Algorithms For Single And Multiple Depot Vehicle Routing Problems With Pickups and Deliveries*	Untuk mengembangkan metodologi solusi yang mampu memecahkan masalah VRP *Pickup* dan *Delivery* yang lebih luas daripada yang telah dipecahkan sebelumnya dalam literatur.	Heuristik terintegrasi yang diusulkan mampu memberikan solusi yang sangat baik untuk masalah VRPPD dengan satu hingga lima depot dan 50 hingga 249 pelanggan dalam beberapa detik.	Pengembangan VRPPD yang diselesaikan dengan menggunakan pembatasan kapasitas yang lebih rendah secara artifisial atau dengan mengatasi gabungan *packing-and-routing problem*

Tabel 2.1 Penelitian Terdahulu (Lanjutan)

Tahun	Penulis	Judul	Tujuan	Kesimpulan	Future Research
2012	Wasan N.A. and Gabor Nagy	*Vehicle Routing Problem with Deliveries and Pickups: Modelling Issues and Meta-heuristics Solution Approaches*	Untuk memberikan ringkasan tentang VRPDP dan penjelasan terperinci tentang beberapa bidang penelitian baru terkait dengan topik tersebut. Pada penelitian ini lebih berfokus pada permasalahan, daripada solusi.	Pada penelitian ini dibahas mengenai beberapa permasalahan VRPDP dengan beberapa pengembangan yang fokus pada permasalahan *demand*, yaitu single *demand* dan combined *demand*. Dari variasi permasalahan tersebut dapat diselesaikan dengan pendekatan yang berbeda-beda sehingga menghasilkan pengembangan VRPDP antara lain VRPB dan VRPMDP untuk *single demands* serta VRPSPD dan VRPDDP untuk *combined demands*.	Pengembangan lebih lanjut mengenai aplikasi beberapa varian VRPDP dalam dunia industri
2012	Shimizu Y. et al	*A hybrid method for solving multi-depot VRP with simultaneous pickup and delivery incorporated with Weber basis saving heuristic*	untuk mengungkapkan beberapa sifat menarik dari permasalahan VRPSPD terutama dalam perbandingan transportasi terpisah dan konfigurasi jaringan terpusat.	Untuk mengembangkan keberlanjutan dalam modern global dan perubahan sistem logistik, studi ini berkaitan dengan VRPSPD multi-depot. Sejauh ini metode apa pun belum diketahui sebelumnya karena kendala solusi yang besar. Untuk mengatasi masalah dalam praktik, maka dikembangkan permasalahan pada depot tunggal dalam kerangka yang sama. Akhirnya, dengan menerapkan pendekatan *tabu search* yang telah dimodifikasi dan algoritma grafik masalah MCF ini dibahas beberapa masalah utama yang terkait dengan karakteristik VRPSPD melalui eksperimen numerik.	Memperluas VRPPD sehingga dapat mengatasi masalah yang lebih luas dan mempertimbangkan berbagai kondisi praktis. Dengan mengembangkan model optimisasi multi-tujuan yang mengatur trade-off antara ekonomi, risiko, layanan, masalah lingkungan, dan sebagainya.

Tabel 2.1 Penelitian Terdahulu (Lanjutan)

Tahun	Penulis	Judul	Tujuan	Kesimpulan	Future Research
2012	Mahmoudi M. and Zhou X.	*Finding optimal solutions for vehicle routing problem with pickup and delivery services with time windows: A dynamic programming approach based on state–space–time network representations*	• Memformulasikan ulang VRPPDTW sebagai model diskrit waktu, multi-dimensi, multi-komoditas, dengan fungsi obyektif linier dan kendala. • Memperluas formulasi *Dynamic Programming* (DP) statis ke kerangka kerja DP yang sepenuhnya tergantung waktu untuk masalah kendaraan tunggal (VRPPDTW). • Mengembangkan prosedur solusi *Lagrangian Relaxation* (LR) untuk menguraikan masalah penjadwalan multi-kendaraan ke urutan masalah kendaraan tunggal.	Dengan merumuskan ulang PDPTW melalui jaringan ruang-waktu untuk mempertimbangkan persyaratan *time windows*, pendekatan yang diusulkan tidak hanya dapat menyelesaikan masalah routing dan penjadwalan kendaraan secara langsung dalam jaringan transportasi skala besar dengan kemacetan yang bergantung pada waktu, tetapi juga menghindari prosedur yang rumit untuk menghilangkan sub-tur. Memperkenalkan konstruksi kendaraan virtual, pendekatan yang diusulkan dapat sepenuhnya menggabungkan serangkaian faktor interaksi penuh antara permintaan penumpang dan kapasitas kendaraan yang terbatas dalam model ini untuk memperoleh solusi yang layak dan estimasi *cost-benefit* yang besar secara sistem untuk setiap permintaan melalui metode sub-gradien berbasis penentuan harga. Prosedur pengoptimalan dan penetapan harga bersama ini dapat membantu penyedia layanan jaringan transportasi untuk menghitung biaya operasi dari permintaan perjalanan yang didistribusikan secara spasial dan temporal.	Pengembangan model untuk kasus-kasus berikut: 1. Penumpang mungkin menginginkan kapasitas berbagi perjalanan yang berbeda (yaitu penumpang mungkin ingin berbagi perjalanannya dengan hingga hanya satu penumpang, sedangkan penumpang lain mungkin tidak memiliki pembatasan tentang jumlah penumpang yang berbagi perjalanan dengan dia). 2. Penumpang mungkin ingin dilayani atau tidak oleh suatu kendaraan tertentu. 3. Permintaan transportasi dapat berisi sekelompok penumpang yang memiliki asal yang sama, sementara mereka mungkin memiliki tujuan yang berbeda. Atau, permintaan transportasi bisa berisi sekelompok penumpang yang memiliki tujuan yang sama.

Tabel 2.1 Penelitian Terdahulu (Lanjutan)

Tahun	Penulis	Judul	Tujuan	Kesimpulan	Future Research
2013	Rais A. et al	*New mixed integer-programming model for the pickup-and-delivery problem with transshipment*	Menentukan rute dengan biaya kendaraan minimum untuk mengunjungi semua pelanggan dari depot kendaraan utama	Permasalahan utama untuk mendorong multimodal dan interoperabilitas dalam transportasi adalah pemindahan orang dan barang antara kendaraan dan moda transportasi. Untuk perencanaan transportasi dan logistik multimodal, *transshipment* diterjemahkan sebagai fleksibilitas untuk mengubah mode dan operator sistem transportasi. Untuk fleksibilitas tambahan, jaringan transportasi yang relevan dapat dirancang dengan *node transshipment* yang ditempatkan secara strategis untuk memenuhi transportasi jarak jauh serta jarak pendek dengan berbagai cara. Hal ini dapat membantu mengoptimalkan tujuan dengan mengurangi total biaya rute kendaraan untuk melayani semua permintaan pelanggan.	Pengembangan model *Mixed Integer Programming* (MIP) untuk mengatasi masalah yang lebih besar di masa depan
2014	Toth, P. and Vigo, D.	*VRP with Split Deliveries*	Mendefinisikan dan mengklasifikasikan model SDVRP, serta solusi penyelesaiannya.	Dari beberapa kajian dan penelitian menunjukkan bahwa penyelesaian dengan SDVRP menghasilkan *saving* yang besar ketika pelanggan diizinkan untuk dikunjungi lebih dari sekali. Namun sisi negatifnya, beberapa kunjungan mungkin tidak diinginkan jika dilihat dari sudut pandang pelanggan. Selain itu, penanganan algoritma untuk *split service* mungkin memerlukan komponen pemrograman khusus dengan menggabungkan pendekatan eksak dengan metaheuristik.	Pengembangan konsep algoritma yang paling cocok untuk merepresentasikan (sebagian) tur dan solusi sehingga operasi mendasar dalam pencarian lokal dan heuristik berbasis populasi dapat dilakukan dengan cara yang efisien dan efektif.

Tabel 2.1 Penelitian Terdahulu (Lanjutan)

Tahun	Penulis	Judul	Tujuan	Kesimpulan	Future Research
2016	Bianchessi, N. & Irnich, S.	*Branch-and-Cut for the Split Delivery Vehicle Routing Problem*	Pendekatan algoritma *Branch and Cut* untuk penyelesaian permasalahan SDVRP secara eksak dengan melakukan pemotongan atau penghentian solusi-solusi yang tidak *feasible*. Sehingga didapat rute dan jumlah pengiriman yang *feasible*.	Output dari algoritma yang diusulkan menunjukkan peningkatan optimalitas solusi sekitar 5% jika dibandingkan dengan penelitian sebelumnya yang ada di literature. Peningkatan juga terlihat pada beberapa batas bawah dan atas, dimana hasilnya tidak terpengaruh oleh jenis pra-pemrosesan yang digunakan untuk memastikan validitas untuk waktu dan biaya perjalanan.	Teknik *Branch and Cut* juga dapat membantu penyelesaian untuk pengembangan varian lain dari VRP, di mana *node* atau *arcs* tertentu dilintasi lebih dari satu kali. Misalnya, dalam VRP *with intermediate replenishment facilities* (VRPIRF) atau untuk perutean kendaraan listrik dengan baterai yang dapat diisi ulang di stasiun pengisian
2016	Rajappa, G. P., Wilck, J. H., & Bell, J. E. (2016)	*An ant colony optimization and hybrid metaheuristics algorithm to solve the split delivery vehicle routing problem*	Membandingkan hasil solusi permasalahan SDVRP dengan dua pendekatan, antara lain ACO, dan *Hybrid Combination Algorithm (*ACO, *Genetic Algorithm*, dan heuristik)	Pada penelitian tersebut menguji ketiga algoritma metaheuristik pada beberapa dataset yang digunakan di beberapa penelitian sebelumnya yaitu Derigs et al. (2010) dan Jin et al. (2008). Dari hasil pengujian tersebut didapat hasil yang cukup menjanjikan untuk ACO, dimana 14 dari 21 dataset yang digunakan dalam penelitian Derigs et al. menghasilkan nilai yang lebih baik daripada 2 algoritma lainnya. Sedangkan untuk dataset dalam penelitian Jin et al., ACO menghasilkan nilai yang paling baik untuk semua dataset.	

Tabel 2.1 Penelitian Terdahulu (Lanjutan)

Tahun	Penulis	Judul	Tujuan	Kesimpulan	Future Research
2016	Maya R., et al	Penentuan Rute Truk Tangki Penyiraman Taman Dengan Konsep Vehicle Routing Problem dan Rural Postman Problem	Mengetahui penentuan rute penyiraman untuk mempersingkat waktu total penyiraman, mengetahui sisa air yang ada dalam truk tangki setelah proses penyiraman selesai, mengetahui waktu kerja untuk proses penyiraman taman setelah dilakukan penentuan rute penyiraman.	• Didapatkan 7 rute penyiraman taman. Rute terpanjang ada pada rute 2 dengan total *completion time* sebesar 455 menit dan rute terpendek pada rute ke 7 dengan total *completion time* sebesar 256,54 menit. Rute hasil pengembangan algoritma ini lebih sedikit jika dibandingkan dengan rute *existing* sebanyak 8 rute. • Total sisa air yang ada dalam truk tangki pada rute *existing* adalah sebesar 14.120,005 dan sisa air pada rute baru dengan pengembangan algoritma adalah sebesar 13.988,7 liter. Sehingga selisih total sisa air pada kedua rute tersebut adalah sebesar 121,3 liter • Nilai *completion time* rute *existing* sebesar 2.655,206 menit, sedangkan nilai *completion time* rute baru adalah sebesar 2.616,368 menit.	Pengembangan algoritma yang dilakukan menghasilkan *completion time* yang lebih pendek, sisa air yang lebih sedikit, dan waktu lembur yang lebih singkat meskipun hasilnya tidak terlalu signifikan.
2017	Viga A. dan Eminugroho R.	Penyelesaian Masalah Rute Penyiraman Tanaman Menggunakan Algoritma *Artificial Immune System* (AIS) di Kota Yogyakarta	Untuk membentuk rute optimal yang memenuhi *demand* air untuk taman dengan kendala kapasitas dan waktu pelayanan	Rute penyiraman tanaman di kota Yogyakarta untuk 16 titik dengan menggunakan truk berkapasitas 5000 Liter diselesaikan dengan penyelesaian CVRPTW menggunakan algoritma *Artificial Immune System* (AIS) dengan tahapan *route construction, route minimization*, dan *distance improvement* menghasilkan 4 rute dengan jarak 38,53 km	Melakukan pengembangan Algoritma *Artificial Immune System* (AIS) dengan mempertimbangkan variabel jarak, waktu, dan *multiroute*. Selain itu juga pengembangan dalam hal komputasi menggunakan aplikasi, seperti *Matlab*, *SPSS*, dsb.

Tabel 2. 2 Gap Penelitian

Nama Penulis	Topik Penelitian										
	Pickup and Delivery Problem	*Split Service*	*Time Window*	*Software*	Metode Penyelesaian			Tipe kendaraan		Tipe Trip	
					Eksak	Heuristik	Metaheuristik	Homogenous	Heterogenous	Single Trip	Multi Trip
Desaulniers, et al (2002)	√	-	√	-	-	-	-	-	-	√	-
Gabor Nagy and Said Salhi (2003)	√	-	-	VAX Fortran	-	√	-	√	√	√	√
Wassan, et al (2012)	√	√	√	CPLEX 12.5	-	-	√	√	-	√	-
Shimizu, et al (2016)	√	-	√	Visual C++	-	-	√	√	-	√	-
Mahmoudi (2012)	√	-	√	Visual C++	-	√	-	√	-	√	-
Rais Et al (2013)	√	-	√	Gurobi 5.5.0	-	√	-	-	√	√	-

Tabel 2.2 Gap Penelitian (Lanjutan)

Nama Penulis	Topik Penelitian										
	Pickup and Delivery Problem	*Split Service*	*Time Window*	*Software*	Metode Penyelesaian			Tipe kendaraan		Tipe Trip	
					Eksak	Heuristik	Metaheuristik	Homogenous	Heterogenous	Single Trip	Multi Trip
Toth, P. and Vigo, D. (2014)	√	√	√		√	-	-	√	-	-	√
Bianchessi, N. & Irnich, S. (2016)	√	√	√	CPLEX 12.6.0.1	√	-	-	-	-	√	-
Rajappa, G. P., Wilck, J. H., & Bell, J. E. (2016)	√	√	√	CPLEX dan Fortran 95	-	-	√	-	-	-	√
Maya R., et al (2016)	-	-	√	Excel 2013	√	-	-	√	-	√	-
Viga A. et al (2017)	-	-	√	-	-	-	√	√	-	√	-

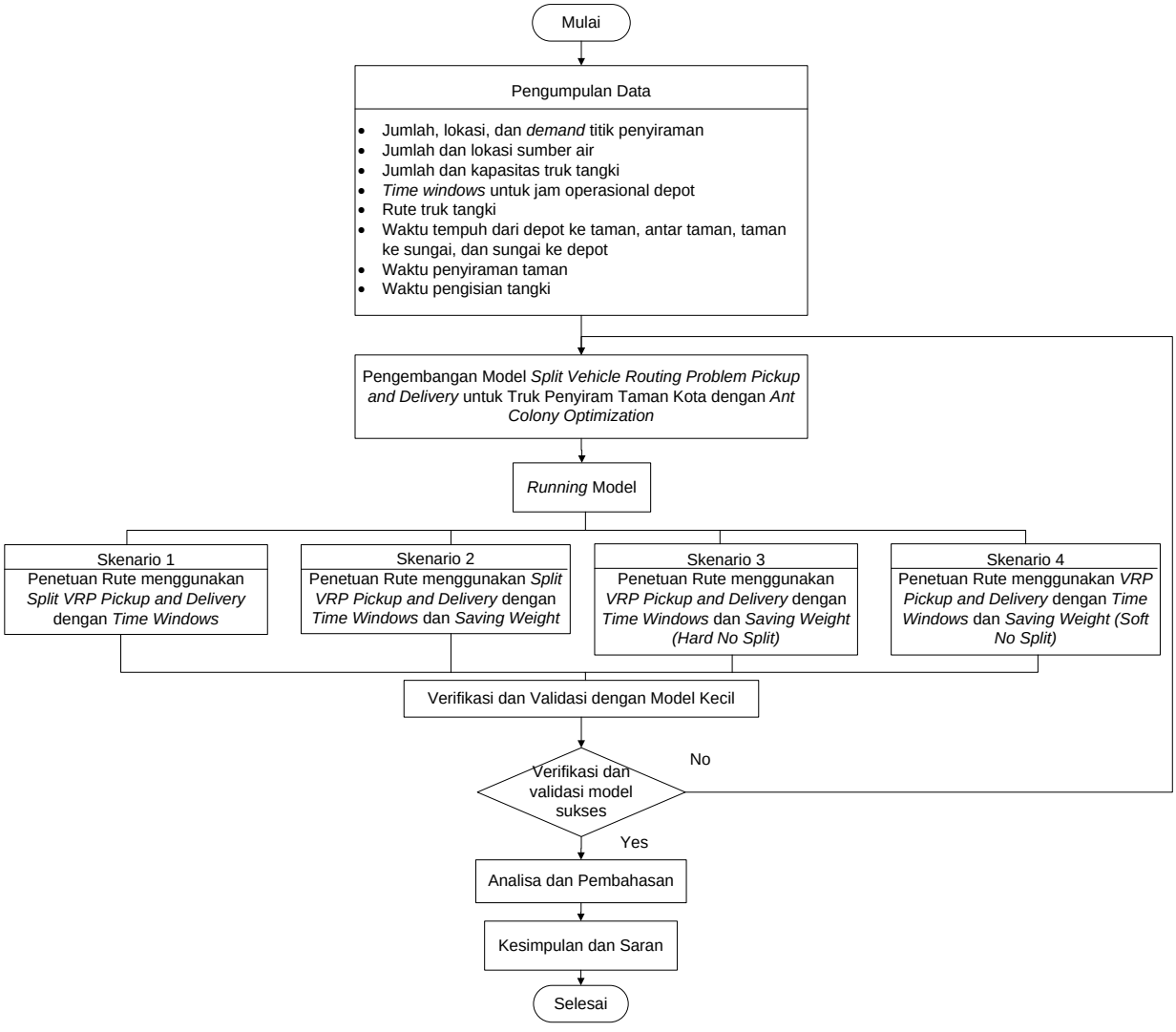
(Halaman ini sengaja dikosongkan)

BAB 3
METODOLOGI PENELITIAN

Pada bab ini akan dijelaskan mengenai metodologi penelitian berupa alur pelaksanaan penelitian dan penjelasan dari alur pelaksanaan penelitian. Alur penelitian ini meliputi beberapa tahap yaitu tahap identifikasi dan perumusan masalah, pengumpulan data, pengolahan data, analisis dan intrepretasi data, kesimpulan dan saran.

3.1.Alur Penelitian

Berikut merupakan alur pelaksanaan penelitian tesis yang menjadi dasar dalam pelaksanaan penelitian.



Gambar 3. 1 Flowchart Metodologi Penelitian

3.2.Penjelasan Flowchart Pelaksanaan Penelitian

Pada sub bab ini dijelaskan mengenai *flowchart* metodologi pelaksanaan penelitian. Metodologi pelaksanaan tersebut terdiri dari tahapan-tahapan yang dilakukan dalam penelitian untuk mencapai tujuan penelitian ini.

3.2.1 Tahap Pengumpulan Data

Pada tahap ini dilakukan pengumpulan data-data yang dibutuhkan untuk dilanjutkan ke tahap pengolahan data dan identifikasi proses penjadwalan truk tangki penyiraman DKRTH Kota Surabaya Rayon Timur. Data yang diperoleh berasal dari DKRTH Kota Surabaya Rayon Timur, observasi langsung, dan *Google Maps* yang sesuai dengan permasalahan yang diteliti. Data yang diperoleh selanjutnya akan diolah. Hasil dari pengolahan data akan digunakan sebagai *input* pada model-model yang ada pada penelitian ini. Pada penelitian ini data-data yang dibutuhkan antara lain:

1. Jumlah, lokasi, dan *demand* titik penyiraman
2. Jumlah dan lokasi sumber air
3. Jumlah dan kapasitas truk tangki
4. *Time windows* untuk jam operasional depot
5. Waktu tempuh dari depot ke taman, antar taman, taman ke tiap sungai, sungai ke depot
6. Waktu penyiraman taman
7. Waktu pengisian tangki
8. Rute dan penugasan truk tangki *existing*

3.2.2 Proses Pemodelan *Vehicle Routing Problem*

Pada tahap ini akan dilakukan proses pemodelan *Vehicle Routing Problem*. Fungsi tujuan dari model ini adalah untuk membuat rute perjalanan truk tangki penyiraman taman pasif yang optimal dengan waktu minimum. Pembuatan model akan disesuaikan dengan kebutuhan rute truk tangki penyiraman DKRTH Kota Surabaya di semua titik lokasi penyiraman dengan input berupa lokasi kantor DKRTH Rayon Timur (depot), lokasi sumber air (*pickup node*), taman pasif (*delivery node*), rute dan waktu penyiraman, kapasitas tangki, waktu tempuh antar lokasi penyiraman. Output yang dihasilkan nantinya berupa rute perjalanan truk tangki secara keseluruhan dalam satu *shift*. Adapun *constraint* yang harus dipenuhi dalam model ini, antara lain:

1. Seluruh kendaraan beroperasi dalam *time windows* yang telah ditentukan mengikuti jam operasional kantor DKRTH Kota Surabaya, yaitu untuk *shift* pagi mulai dari 06.00-15.00 WIB.
2. Semua kendaraan berangkat dan kembali ke depot dalam keadaan kosong
3. Kapasitas kendaraan adalah 5000 Liter
4. Dalam memenuhi permintaan taman, masing-masing taman dapat dikunjungi lebih dari 1 kali dengan kendaraan yang sama maupun berbeda dalam waktu yang berbeda.

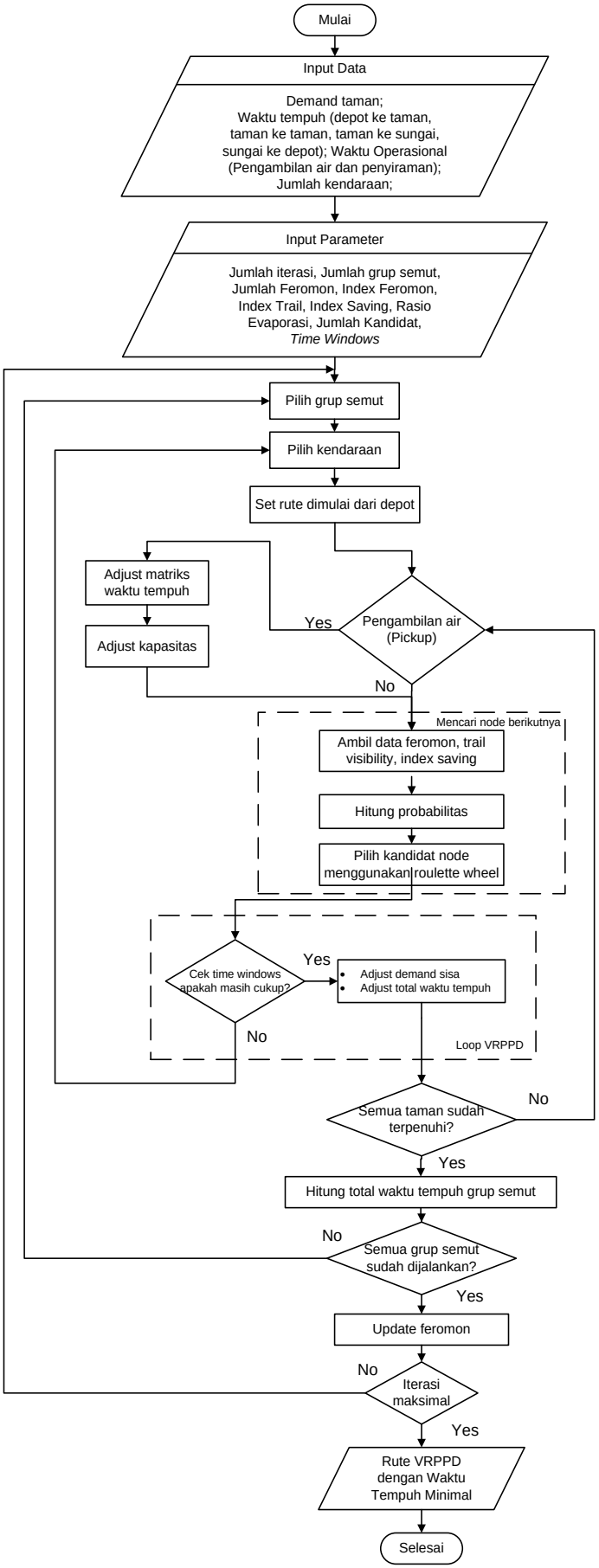
5. Waktu *loading* kendaraan ketika melakukan proses pengisian tangki diasumsikan konstan 15 menit untuk 1 tangki dan waktu *unloading* ketika melakukan melakukan kegiatan penyiraman taman diasumsikan konstan 20 menit untuk volume air 5000 Liter.

3.2.3 Pengembangan VRP dengan Algoritma *Ant Colony Optimization*

Pada penelitian ini terdapat 2 langkah utama yang akan dijalankan, yaitu pembangkitan rute dan pembagian menjadi beberapa sub rute sesuai dengan constraint yang ada.

Pada pengembangan model sesuai dengan *flowchart* pada Gambar 3.3, pemilihan rute awal dilakukan secara acak menggunakan probabilitas dan pemilihan kandidat *node* dengan *roulette wheel* (pembangkitan bilangan random) Algoritma *Ant Colony Optimization* (ACO). Penugasan kendaraan dan penentuan rute dimulai dari depot. Karena truk berangkat dari depot dalam keadaan kosong, maka dilakukan pengisian tangki (*pickup*). *Pickup node* disini merupakan *soft constraint* dan hanya dikunjungi jika kapasitas truk habis. Pemilihan lokasi *pickup* berdasarkan waktu tempuh terpendek dari *node* yang terpilih. Jika telah mengunjungi *pickup node*, maka nilai waktu tempuh (*travel time*) bertambah sejumlah waktu tempuh dari *node* terakhir ke lokasi *pickup* dan kapasitas di-*adjust* menjadi nilai maksimal sebanyak 5000 Liter.

Untuk tiap *node* yang terpilih, jika kapasitas truk tidak mampu memenuhi semua *demand* suatu taman, maka sisa *demand* yang tidak terpenuhi disimpan dalam variable *demand* sisa dan truk akan melanjutkan perjalanan menuju lokasi pengisian air (*pickup*). Dari proses ini memungkinkan untuk terjadi proses *split*. Proses *split* terjadi jika suatu truk tidak dapat melayani keseluruhan *demand* taman dan *demand* sisa dipenuhi oleh truk yang lain. Jika *demand* taman kurang dari atau sama dengan kapasitas truk, maka truk akan memenuhi semua *demand* taman. Jika waktu belum melebihi batasan *time windows*, maka truk akan mengunjungi titik-titik taman berikutnya dan melakukan penyiraman. Apabila waktu tidak cukup maka truk akan kembali ke depot.



Gambar 3. 2 Flowchart Algoritma ACO

3.2.4 Verifikasi dan Validasi Model

Validasi adalah proses pengujian untuk menentukan model yang telah dibuat apakah benar-benar merepresentasikan permasalahan *real* yang dimodelkan. Validasi dilakukan untuk menjamin model metaheuristic yang telah dikembangkan merepresentasikan permasalahan pada system nyata.

Pada verifikasi model dilakukan *running* model dengan menggunakan sampel kecil, apakah algoritma yang digunakan sudah dapat berjalan dengan baik sehingga pencarian rute optimal dapat dijalankan. Dari hasil proses *running* kemudian disesuaikan dengan hasil perhitungan manual. Apabila verifikasi dan validasi model sudah terpenuhi maka selanjutnya model siap digunakan untuk sampel yang lebih besar.

3.2.5 Proses *Running* Model

Setelah dilakukan pengembangan model, terdapat tahap pengerjaan model. Dalam tahap ini dilakukan *running* model untuk penugasan kendaraan dan pencarian rute optimal yang menghasilkan waktu tempuh minimal. *Running* model dilakukan sebanyak beberapa kali replikasi dengan uji coba berbagai parameter untuk mendapat nilai yang mendekati optimal. Parameter yang menghasilkan nilai terbaik akan dilakukan replikasi kembali agar didapat rute rekomendasi.

Untuk mengurangi variansi maka *running* model harus dilakukan sebanyak n kali replikasi. Untuk mendapatkan nilai n maka perlu dilakukan repliasi awal n0 yaitu sebanyak 10 kali replikasi. Selanjutnya untuk mendapatkan nilai n’ (n replikasi yang dibutuhkan) maka dilakukan perhitungan nilai *Half-Width* (HW) untuk menghitung nilai error dari data tersebut. Untuk menghitung *half width* dapat dihitung dengan rumus:

$$hw = \left(\frac{t_{n-1,\frac{\propto}{2}} \times s}{\sqrt{n}}\right)$$

dimana : hw : half width

$t_{n-1,\frac{\propto}{2}}$: factor yang diperoleh dari tabel T

s : standar deviasi sampel

n : jumlah sampel

dari nilai hw diperoleh nilai n’ dengan rumus:

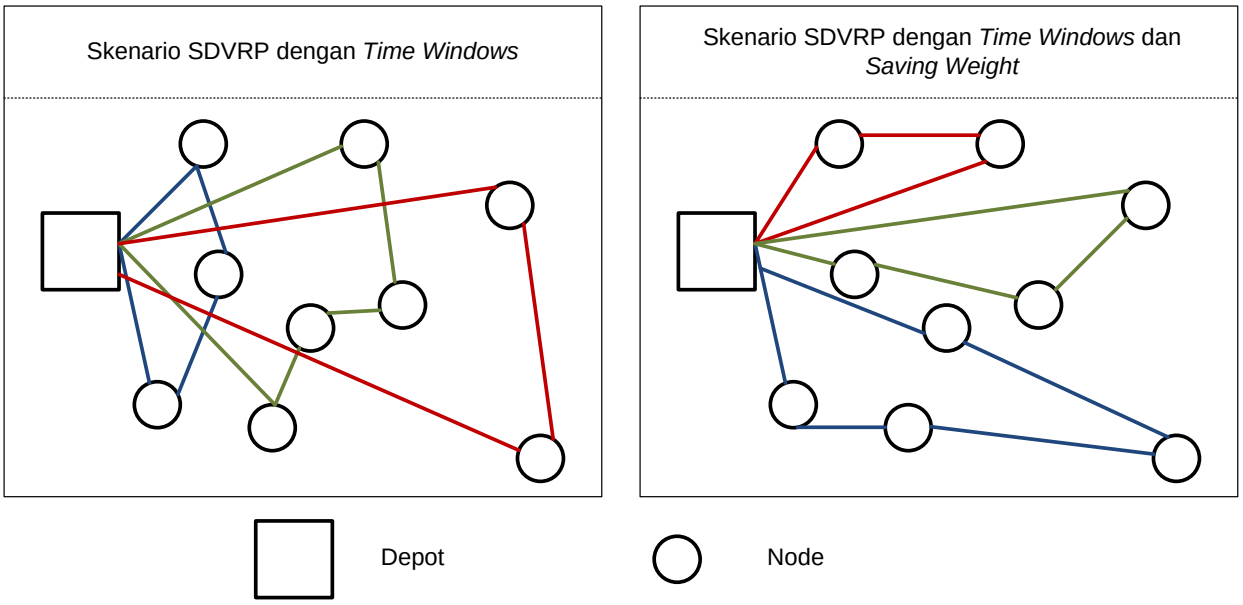
$$n' = \left[\frac{(Z\propto_{/2})s}{\left(\frac{e\%}{1+e\%}\right)\bar{x}}\right]$$

Pada tahap running model ini menggunakan bahasa pemrograman *Python*. Hasil dari running model adalah rute terbaik dengan waktu tempuh dan jumlah kendaraan minimum.

Tahap pembuatan skenario dilakukan sesuai dengan proses penentuan rute truk penyiraman. Terdapat empat skenario yang dibuat yaitu skenario *Split Delivery* VRP dengan *Time Windows*, skenario *Split Delivery* VRP dengan *Time Windows* dan *Saving Weight*, skenario VRP *Pickup and Delivery with Time Windows* dan *Saving Weight (Hard No Split)*, skenario VRP *Pickup and Delivery with Time Windows* dan *Saving Weight (Soft No Split)*.

Pada Algoritma ACO sebelum dilakukan pemilihan *node*, untuk masing-masing kendaraan dilakukan pemilihan kandidat rute berdasarkan waktu tempuh minimum. Pada skenario satu, untuk kendaraan yang mendapatkan giliran awal untuk pemilihan kandidat, akan mendapatkan rute yang terdekat dan kendaraan terakhir akan mendapatkan rute terjauh. Oleh karena itu, pada skenario kedua dibuat model dengan mempertimbangkan *time windows* dan *saving weight* dalam pemilihan kandidat rute agar pembagian rute merata untuk masing-masing kendaraan.

Lain halnya dengan dua skenario sebelumnya, untuk skenario ketiga dan keempat tidak menggunakan *split*. Skenario ini mengacu pada kondisi *real* kegiatan penyiraman, apabila kondisi untuk *split service* pada praktiknya tidak memungkinkan untuk diaplikasikan. Skenario ketiga menggunakan penyelesaian VRP *Pickup and Delivery with Time Windows* dan *Saving Weight* (*Hard No Split*), dimana untuk satu *node* dapat dikunjungi oleh 1 kendaraan yang sama lebih dari satu kali untuk memenuhi semua *demand* pada *node* tersebut. Sedangkan untuk skenario terakhir menggunakan penyelesaian VRP *Pickup and Delivery with Time Windows* dan *Saving Weight* (*Soft No Split*). Untuk skenario ini memiliki konsep yang hampir mirip dengan skenario ketiga, yaitu tanpa *split service*. Namun untuk skenario ini, batasan *split* ini tidak baku dan apabila kondisinya tidak memungkinkan, bisa terjadi *split service*.



Gambar 3. 3 Skenario Penentuan Kandidat Rute

3.2.6 Analisa dan Interpretasi Data

Pada tahap ini akan dilakukan analisis terhadap hasil pengolahan data yang dilakukan sebelumnya. Analisis yang dilakukan pada tahap ini meliputi perbandingan analisis hasil *running* menggunakan pemrograman *python* berdasarkan penjadwalan sebelumnya dan analisis data berdasarkan model yang telah dibuat.

3.2.7 Kesimpulan dan Saran

Tahap pengambilan kesimpulan bertujuan untuk menarik suatu kesimpulan dalam menjawab tujuan penelitian yang dilakukan. Penarikan kesimpulan dilakukan berdasarkan hasil analisis dan interpretasi data pada hasil penelitian yang sudah dilakukan. Saran dan rekomendasi diharapkan dapat dijadikan bahan masukan/pertimbangan yang berkaitan dengan penelitian yang telah dilakukan dan perbaikan untuk penelitian selanjutnya.

(halaman ini sengaja dikosongkan)

BAB 4
PENGUMPULAN DATA

Pada bab ini akan ditampilkan hasil pengumpulan data yang akan digunakan dalam penelitian ini. Data tersebut berkaitan dengan truk tangki penyiraman, jarak tempuh, waktu tempuh dan rute kendaraan *existing.*

4.1. Penyiraman Taman Pasif di Kota Surabaya

Penyiraman taman pasif oleh kendaraan truk tangki di Kota Surabaya Rayon Timur dimulai dari keberangkatan masing-masing kendaraan yang dimulai dari depot yang berlokasi di Kantor DKRTH Rayon Timur dan dilanjutkan dengan pengisian tangki air (disebut sebagai proses *loading*) pada sungai-sungai yang terdekat dengan lokasi penyiraman. Truk tangki memiliki kapasitas sebesar 5000 Liter. Ketika kapasitas sudah penuh, kendaraan akan melakukan proses penyiraman taman pasif (disebut sebagai *unloading*) di sepanjang jalan sesuai dengan rute yang ditentukan. Proses *loading* diasumsikan secara konstan membutuhkan waktu 15 menit dan proses *unloading* diasumsikan secara konstan membutuhkan waktu 20 menit untuk kapasitas maksimum 5000 Liter. Jam waktu keberangkatan kendaraan serentak pada pukul 06.00 WIB dan harus kembali ke depot sebelum atau tepat pada pukul 15.00 WIB.

Permasalahan rute penyiraman taman yang dimiliki DKRTH Kota Surabaya saat ini adalah jumlah kendaraan yang belum optimal untuk memenuhi permintaan penyiraman taman di seluruh taman pasif di wilayah Surabaya, khususnya di Surabaya Rayon Timur terlebih jika ada kendaraan yang rusak. Di sisi lain permintaan penyiraman taman semakin meningkat sejalan dengan meningkatnya taman-taman kota dan ruang terbuka hijau di Surabaya, sehingga diperlukan optimasi sebagai alat bantu yang digunakan sebagai pertimbangan dalam penentuan rute secara cepat dan tepat.

4.2. Pengumpulan Data

Pada penelitian ini menggunakan data yang didapat dari Dinas Kebersihan dan Ruang Terbuka Hijau (DKRTH) Kota Surabaya Rayon Timur dan dari hasil pengamatan jarak dan waktu menggunakan bantuan *Google Maps*. Data yang didapat dari Google Maps diambil pada waktu jam operasional dan kedepannya akan diasumsikan bernilai konstan.

4.3. Data Kendaraan Truk Tangki Penyiraman

Pada data kendaraan penelitian ini berfokus pada jenis kendaraan truk tangki. Data tersebut adalah data yang menunjukkan jumlah kendaraan yang dimiliki oleh DKRTH Kota Surabaya Rayon Timur yang berada di depot. Truk tangki memiliki kapasitas yang sama yaitu

5000 Liter. Data ini selanjutnya akan digunakan untuk melihat utilitas kendaraan yang dimiliki oleh seluruh kendaraan yang akan melayani taman-taman kota yang tersebar di Kota Surabaya. Data tersebut dapat dilihat pada Tabel 4.1.

Tabel 4. 1 Data Kendaraan Truk Tangki Penyiraman

No	NO POLISI	TIPE KENDARAAN	KAPASITAS
1	L 9023 QP	ISUZU/NKR 71 PS	5000 Liter
2	L 9460 NP	ISUZU/NKR 71 PS	5000 Liter
3	L 9176 NP	ISUZU/NKR 71 PS	5000 Liter
4	L 9321 NP	ISUZU/NKR 71 PS	5000 Liter
5	L 9061 NF	ISUZU/NKR 71 PS	5000 Liter
6	L 9313 NP	ISUZU/NKR 71 PS	5000 Liter
7	L 9018 RP	ISUZU/NKR 71 PS	5000 Liter
8	L 9564 NP	ISUZU/NKR 71 PS	5000 Liter
9	L 9490 NP	ISUZU/NKR 71 PS	5000 Liter

4.4. Data Lokasi *Node* yang Dikunjungi Truk Tangki

DKRTH Surabaya Rayon Timur memiliki 1 lokasi depot truk tangki (diasumsikan sebagai *node* 1) sebagai tempat awal keberangkatan dan tempat parkir setelah melakukan kegiatan penyiraman. Terdapat 46 taman (diasumsikan sebagai *node* 2 s/d 47) yang tersebar di wilayah Surabaya Timur. Data tersebut dapat dilihat pada tabel 4.2.

Tabel 4. 2 Data Lokasi Taman

No	Lokasi	No	Lokasi
1	Kendalsari Wonorejo	24	Mulyosari
2	Raya Pandugo	25	Kalirungkut
3	Kedung Baruk Timur	26	Rungkut Alang-Alang
4	Mer Kedung Asem	27	Rungkut Asri
5	Mer Gunung Anyar	28	Rungkut Yakaya
6	Bratang Rmi	29	Kedung Asem
7	Ngagel Jaya	30	Rungkut Pasar Pahing
8	Ngagel Jaya Selatan	31	Dharmahusada Indah 1
9	Pucang	32	Dharmahusada Utara 8
10	Kertajaya	33	Merr Kalijudan Tepi
11	Dharmawangsa	34	Dhramahusada Indah 2
12	Karang Menjangan	35	Jemur Handayani
13	Raya Menur	36	Jemursari
14	Raya Karangmenjangan	37	Puskesmas Jemursari (Aset)
15	Putro Agung	38	Rotonde Raya Prapen

Tabel 4.2. Data Lokasi Taman (Lanjutan)

No	Lokasi	No	Lokasi
16	Raya Bronggalan	39	Kutisari
17	Dharmahusada Indah 1	40	Merr Kopertis
18	Kaliwaron	41	Ir. Soekarno
19	Tambang Boyo	42	Merr Kalijudan Tepi
20	Rotonde Muestopo	43	Arif Rahman Hakim (Arah UHT)
21	Mulyorejo Utara	44	Raya Manyar
22	Sutorejo	45	Semolowaru Utara
23	Kalisari	46	Arif Rahman Hakim (Arah ITATS)

4.5. Data *Demand* Taman

Setiap lokasi taman memiliki jumlah *demand* yang berbeda-beda. Hal ini tergantung luas dan banyaknya ruas pada masing-masing taman. Semua taman harus dilayani oleh kendaraan yang mengunjungi lokasi tersebut dengan batasan kapasitas kendaraan dan juga waktu pelayanan depot. Kendaraan truk tangki memiliki kapasitas 5000 Liter, dengan diasumsikan dalam luasan 1 meter^2 membutuhkan 7,5 Liter air sehingga semakin luas ukuran taman, semakin banyak *demand* yang dibutuhkan dan semakin banyak pula waktu penyiraman untuk taman tersebut.

Setiap kendaraan harus kembali pada pukul 15.00 WIB di Depot DKRTH Surabaya Rayon Timur untuk pertukaran *shift* berikutnya. Jumlah *demand* di beberapa taman lebih besar dari kapasitas kendaraan, sehingga untuk setiap taman dapat dikunjungi oleh lebih dari satu kali. Data ini selanjutnya digunakan untuk melihat penyebaran *demand* di Kota Surabaya yang harus dilayani setiap harinya oleh truk tangki beserta waktu penyiramannya. Data tersebut dapat dilihat pada Tabel 4.3.

Tabel 4. 3 Data *Demand* Taman

No	Lokasi	*Demand* (liter)	Waktu penyiraman (menit)
1	Kendalsari Wonorejo	11250	45
2	Raya Pandugo	7500	30
3	Kedung Baruk Timur	27000	108
4	Mer Kedung Asem	22500	90
5	Mer Gunung Anyar	10500	42
6	Bratang Rmi	1500	6
7	Ngagel Jaya	5250	21
8	Ngagel Jaya Selatan	3750	15
9	Pucang	2625	10,5
10	Kertajaya	4125	16,5
11	Dharmawangsa	3000	12
12	Depan Karang Menjangan	1687,5	6,75
13	Raya Menur	1687,5	6,75

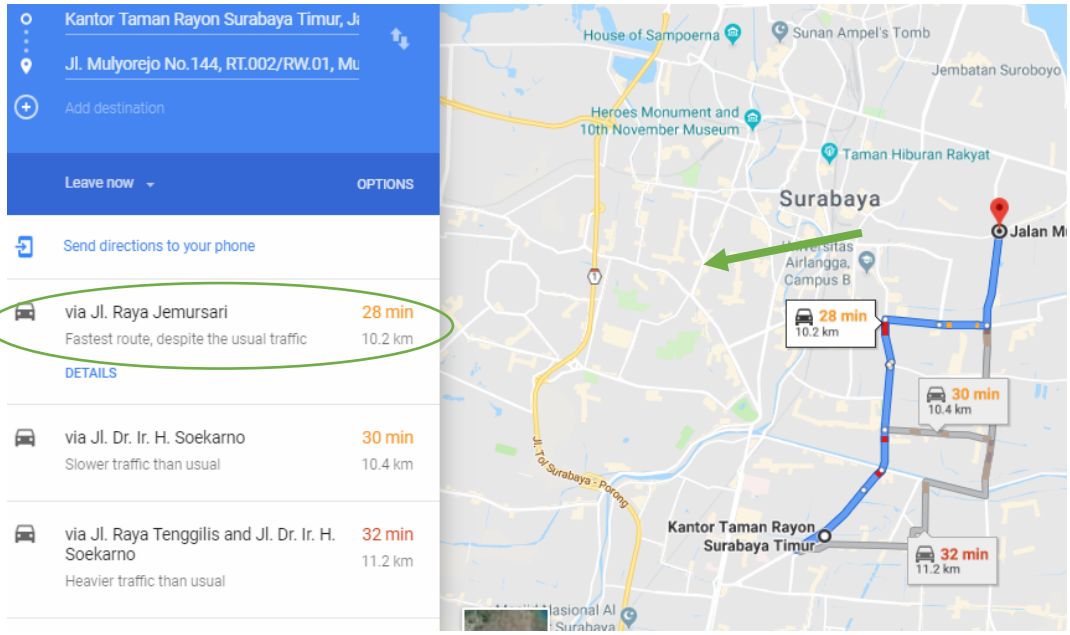
Tabel 4.3. Data *Demand* Taman (Lanjutan)

No	Lokasi	*Demand* (liter)	Waktu penyiraman (menit)
14	Raya Karangmenjangan	1875	7,5
15	Putro Agung	9750	39
16	Raya Bronggalan	12000	48
17	Dharmahusada Indah 1	6750	27
18	Kaliwaron	1875	7,5
19	Tambang Boyo	1687,5	6,75
20	Rotonde Muestopo	3600	14,4
21	Mulyorejo Utara	1312,5	5,25
22	Sutorejo	6750	27
23	Kalisari	1875	7,5
24	Mulyosari	13500	54
25	Kalirungkut	6750	27
26	Rungkut Alang-Alang	10500	42
27	Rungkut Asri	1275	5,1
28	Rungkut Yakaya	4125	16,5
29	Kedung Asem	562,5	2,25
30	Rungkut Pasar Pahing	1875	7,5
31	Dharmahusada Permai	13500	54
32	Dharmahusada Utara 8	9750	39
33	Merr Kalijudan Tepi	6000	24
34	Dharmahusada Indah 2	13500	54
35	Jemur Handayani	15000	60
36	Jemursari	15000	60
37	Puskesmas Jemursari (Aset)	1650	6,6
38	Rotonde Raya Prapen	4050	16,2
39	Kutisari	4125	16,5
40	Merr Kopertis	22500	90
41	Ir. Soekarno	27000	108
42	Merr Kalijudan Tepi	40500	162
43	Arif Rahman Hakim (Arah UHT)	4875	19,5
44	Raya Manyar	12000	48
45	Semolowaru Utara	3750	15
46	Arif Rahman Hakim (Arah ITATS)	8250	33

4.6. Data Waktu Tempuh Antar Lokasi

Pada penelitian ini menggunakan pengembangan model dari VRP, dimana pada model tersebut memerlukan berbagai pertimbangan salah satunya adalah waktu tempuh antar lokasi. Tujuan dari model ini adalah meminimalkan total waktu tempuh untuk semua kendaraan pada jam operasional. Waktu tempuh antar lokasi yaitu: depot ke setiap taman, antar lokasi taman, dan dari

setiap lokasi taman ke sungai didapatkan dari aplikasi Google Maps dengan waktu minimum yang akan dipilih dari beberapa opsi yang disediakan. Satuan yang digunakan adalah menit. Waktu tempuh antar lokasi tidak berbanding lurus dengan jarak tempuhnya karena beberapa faktor yang terjadi di jalan raya misalnya kemacetan, jalan tol, dsb.



Gambar 4. 1 Contoh Pengambilan Data Melalui Google Maps

Pada Gambar 4.1 adalah contoh pengambilan data untuk waktu tempuh dari *node* depot (1) menuju ke *node* Mulyorejo Utara (22). Waktu tempuh dari *node* 1 menuju *node* 22 adalah 28 menit yang didapatkan dari nilai minimum dari beberapa pilihan alternatif jalan yang ada.

4.7. Data Rute dan Penugasan Truk Tangki *Existing*

Data dan penugasan kendaraan truk tangki *existing* didapatkan dari DKRTH Kota Surabaya Rayon Timur. Pada Tabel 4.4 diberikan contoh rute dan penugasan kendaraan truk tangki *existing*. Dari contoh tabel tersebut terdapat beberapa kendaraan yang memiliki rute masing-masing. Kendaraan (2) memulai keberangkatannya pada menit ke 351 dan melakukan penyiraman ke taman (7-8-9-10-11-12-13) dengan frekuensi pengisian air sebanyak 6 kali yang di lakukan di sungai (53-55-56-58) yang kemudian kembali ke depot dan sampai pada menit ke 593. Kendaraan (2) menempuh waktu perjalanan sekitar 24,3 menit dengan jarak 64 km. Kendaraan (3) berangkat pada menit ke 345 dan melakukan penyiraman ke taman (14-15-16-17-18-19-20-21) dengan frekuensi pengisian air sebanyak 9 kali yang di lakukan di sungai (59-61-63-64) yang kemudian kembali ke depot dan sampai pada menit ke 758. Kendaraan (3) menempuh waktu perjalanan sekitar 45,3 menit dengan jarak 121 km. Kedua kendaraan tersebut sampai di depot

sebelum batas *time windows*. Namun pada data tersebut ada 2 kendaraan yang melebihi time windows, yaitu kendaraan (1) dan kendaraan (8).

Kendaraan (1) memulai keberangkatannya pada menit ke 348 dan melakukan penyiraman ke taman (2-3-4-5-6) dengan frekuensi pengisian air sebanyak 17 kali yang di lakukan di sungai (48-49-50) yang kemudian kembali ke depot. Kendaraan 1 menempuh waktu perjalanan sekitar 291 menit dengan jarak 98 km dengan *demand* sebanyak 78750 Liter. Dengan banyaknya *demand* tersebut, mengakibatkan kendaraan (1) beroperasi melebihi *time windows* yang ditentukan. Untuk kendaraan (1) memiliki waktu kepulangan pada menit 1209, dengan kata lain memiliki kelebihan waktu 309 menit dari batas waktu jam operasionalnya. Sehingga mengakibatkan keterlambatan untuk *shift* selanjutnya. Sedangkan kendaraan lain banyak yang memiliki waktu luang yang lebih banyak.

Tabel 4. 4 Contoh Data Rute dan Penugasan Kendaraan Truk Tangki *Existing*

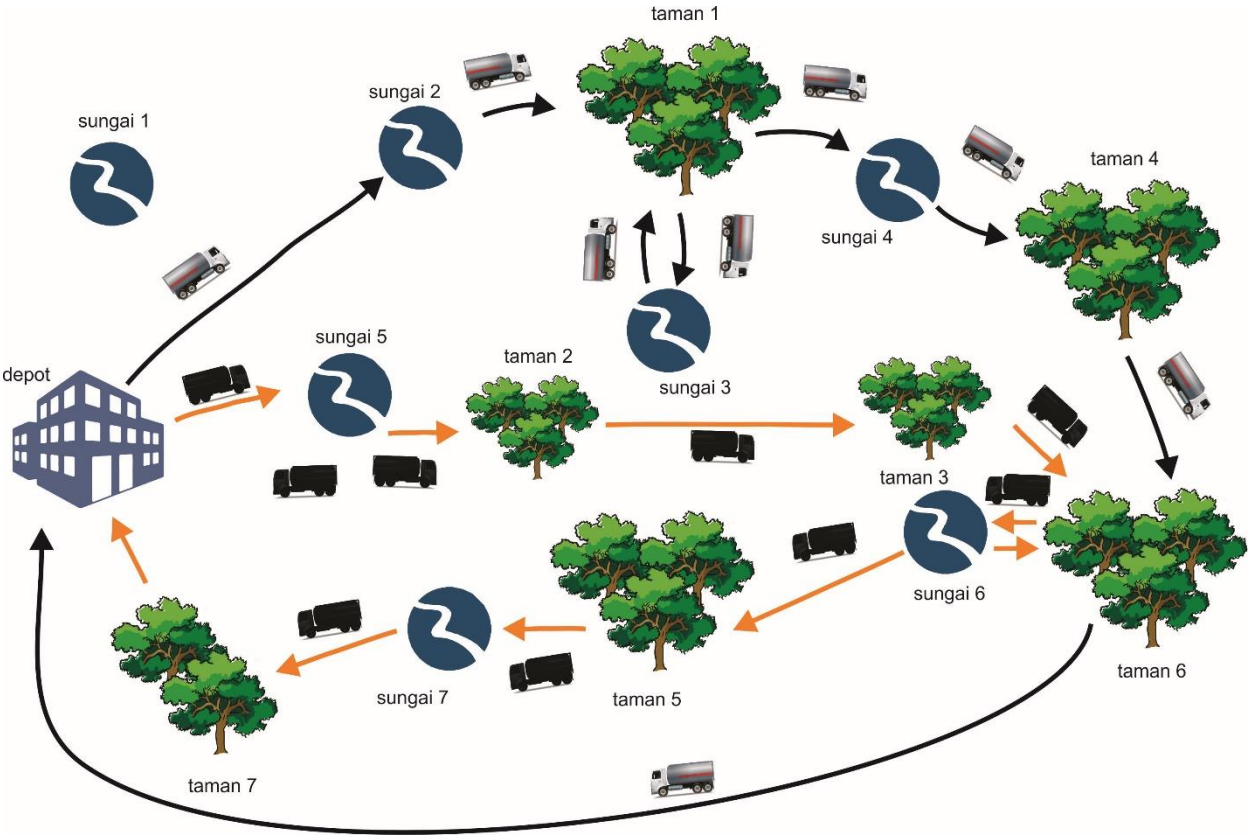
Kendaraan	Jarak dan Waktu	Rute																																										Jumlah Jarak	Jumlah Pengi	*Service Time*	Jadwal Keber	Jadwal kepul	Waktu *Idle*
L 9023 QP	ruto	1	48	2	48	2	48	2	3	49	3	50	4	50	4	50	4	50	4	50	4	50	4	5	50	5	50	5	50	5	50	5	6	49	6	49	6	1							17	570	348	1209	-309
	waktu		12	13	12	13	12	13	8	12	1	16	2	16	2	16	2	16	2	16	2	16	2	6	2	7	2	7	2	7	2	7	2	4	7	4	7	21						291					
	jarak		6	3,9	6	3,9	6	3,9	3	3,6	0,1	4,9	0,75	4,9	0,75	4,9	0,75	4,9	0,75	4,9	0,75	4,9	0,75	1,6	0,75	2	0,75	2	0,75	2	0,75	2	1,4	1,3	2,2	1,3	2,2	6,7						98					
	demand		0	5000	0	5000	0	1250	3750	0	3750	0	5000	0	5000	0	5000	0	5000	0	5000	0	2000	3000	0	5000	0	5000	0	5000	0	4500	500	0	5000	0	5000	0						78750					
L 9460 NP	ruto	1	53	7	8	55	8	9	55	9	10	11	56	11	12	58	12	13	1																										6	178	351	593	307
	jarak		9	5	1	1	1	5	1	3	3	1	1	1	4	1	1	4	22																									64					
	waktu		5	1,6	0,35	0,45	0,45	1,4	0,45	0,8	1,1	0,3	0,3	0,3	1,4	0,65	0,65	1,1	8																									24,3					
	demand		0	1500	3500	0	1750	3250	0	500	2625	1875	0	2250	2750	0	250	1688	0																									21937,5					
L 9176 NP	ruto	1	59	14	15	21	59	21	20	17	61	17	61	17	61	17	16	63	16	63	16	19	18	64	18	1																			9	292	345	758	142
	jarak		15	9	8	4	5	5	5	4	1	1	1	1	1	1	3	2	2	2	2	12	6	2	2	27																		121					
	waktu		8,2	2,8	2,4	1,4	1,4	1,4	1,9	1,6	0,15	0,15	0,15	0,15	0,15	0,15	1,4	0,75	0,75	0,75	0,75	5,2	2,4	0,75	0,75	9,8																		45,3					
	demand		0	1688	1875	1437	0	2163	1688	1149	0	5000	0	5000	0	851	4149	0	5000	0	601	1875	2524	0	4226	0																		39226					

BAB 5
PENGEMBANGAN MODEL

Pada bab ini akan dijelaskan mengenai proses pengembangan model optimasi untuk penentuan rute penyiraman taman di Kota Surabaya dengan pendekatan algoritma *Ant Colony Optimization*.

5.1. Pengembangan Model untuk Penyiraman Taman

Model pada penelitian ini diilustrasikan melalui gambar untuk lebih memahami bagaimana alur proses pada sistem yang sudah ada. Ilustrasi dari model dapat dilihat di Gambar 5.1.



Gambar 5. 1 Ilustrasi Model Penyiraman Taman

Dalam kasus perutean penyiraman taman, setiap truk akan berangkat dan kembali ke depot sesuai dengan jam operasional. Keberangkatan truk dari depot dimulai dari menit ke-360 dan harus kembali sebelum menit ke-900, yang artinya waktu jam operasional satu *shift* dalam sehari adalah 480 menit (setelah dikurangi waktu istirahat sebanyak 60 menit).

Pada Gambar 5.1 truk berangkat dari depot dalam kondisi tangki kosong dan melakukan pengisian air (*loading*) terlebih dahulu di sungai terdekat (*pickup node*). Kemudian kendaraan menuju taman terdekat untuk melakukan penyiraman. Untuk taman yang memiliki *demand* kurang

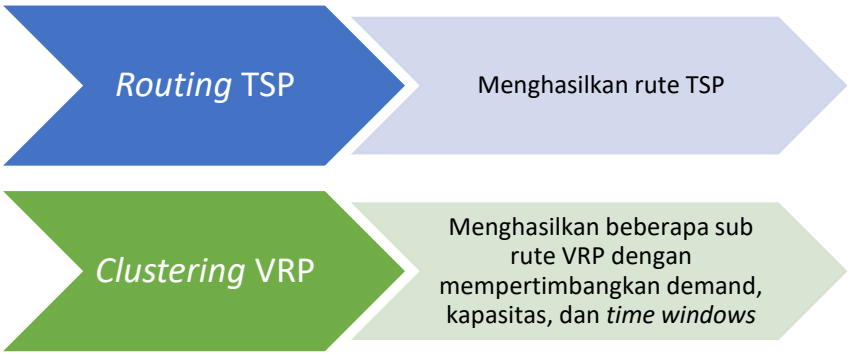
dari atau sama dengan kapasitas, truk hanya perlu melakukan penyiraman (*unloading*) dalam satu kali ritase, seperti yang terlihat pada taman 4.

Apabila *demand* suatu taman lebih dari kapasitas truk, maka taman bisa dikunjungi beberapa kali hingga semua *demand* terpenuhi, baik dengan truk yang sama maupun berbeda. Misalnya, seperti yang terlihat di Gambar 5.1 pada taman 1, kendaraan 1 awalnya melakukan pengisian air di sungai 2 dan melakukan penyiraman di taman 1. Karena *demand* taman 1 melebihi kapasitas truk 1, maka truk 1 kembali lagi melakukan pengisian air di sungai terdekat yaitu di sungai 3. Setelah semua *demand* taman 1 terpenuhi, maka truk 1 melanjutkan perjalanan menuju lokasi penyiraman berikutnya. Berarti dalam kasus ini, taman 1 dilayani oleh truk 1 sebanyak 2 kali.

Lain halnya dengan kasus yang ada di taman 6. Awalnya taman 6 dilayani oleh truk 1. Namun *demand* untuk taman 6 melebihi kapasitas truk 1, sehingga ada sisa *demand* yang belum terpenuhi. Setelah kapasitas truk 1 habis, truk 1 langsung kembali ke depot dan tidak melakukan pengisian ulang di sungai terdekat dan memenuhi sisa *demand* taman 6. Hal ini mungkin terjadi karena waktu yang dibutuhkan untuk memenuhi sisa *demand* tersebut akan melebihi batasan jam operasional, sehingga dibutuhkan kendaraan lain untuk memenuhi *demand* yang tersisa yaitu truk 2 yang berwarna hitam. Hal ini menyatakan bahwa taman 6 dan taman 4 dapat dilayani secara *split service*, yang artinya dapat dilayani lebih dari satu kali dengan kendaraan yang sama maupun kendaraan yang berbeda.

5.2. Pengembangan Algoritma ACO untuk Penyiraman Taman

Pada pengembangan model VRPSDTW dengan pendekatan Ant Colony Optimization untuk permasalahan rute penyiraman taman kota di Surabaya terdapat 2 tahapan utama yang dilakukan, yaitu pembangkitan rute TSP dan pembagian menjadi beberapa sub rute.



Gambar 5. 2 Tahapan Algoritma *Ant Colony Optimization* untuk Permasalahan VRPSDTW

Berikut ini akan dibahas mengenai 2 tahapan utama dalam pengembangan model *Ant Colony Optimization* untuk penentuan rute penyiraman taman kota.

1. Pembangkitan rute TSP dengan pendekatan *Ant Colony Optimization*

   Proses pembangkitan rute TSP dengan algoritma Ant Colony Optimization dilakukan dalam beberapa tahapan. Selanjutnya algoritma tersebut dimodifikasi sedemikian rupa disesuaikan dengan bahasa pemrograman *Python*, koding lengkap terdapat pada Lampiran.

   a) Inisialisasi

      Pada tahap ini dilakukan inisialisasi input data dan parameter. Input data merupakan nilai yang berisi kondisi dan permasalahan suatu kasus. Biasanya data input sudah ditentukan nilainya berdasarkan permasalahan yang ada. Sedangkan untuk input parameter merupakan nilai yang tidak pasti dan dalam penentuan nilainya perlu dilakukan beberapa percobaan dengan rekomendasi tertentu. Dalam permasalahan *routing* TSP menggunakan algoritma ACO, input yang digunakan antara lain:

      - Data:
        - Waktu tempuh depot ke taman
        - Waktu tempuh depot ke sungai
        - Waktu tempuh taman ke taman
        - Waktu tempuh taman ke sungai
      - Parameter:
        - Feromon awal (τ_{ij}^{1})

          Nilai feromon akan selalu diperbaharui pada setiap iterasi algoritma, mulai dari iterasi pertama sampai iterasi maksimum yang ditentukan atau telah mencapai hasil yang optimal.
        - Indeks *Visibility* (*h*)

          Nilai *visibility* antar *node* adalah invers dari jarak ($\frac{1}{jarak}$) dari masing-masing *node*.
        - Rasio evaporasi (ρ)

          Rasio evaporasi adalah tingkat penguapan feromon, nilai rasio ini adalah $0 < \rho < 1$.
        - Jumlah grup semut (*N*)

          Algoritma ACO merupakan *population based metaheuristic*, sehingga pencarian solusi langsung berdasarkan grup-grup tertentu. Jumlah grup semut yang ditetapkan akan dijalankan dalam satu kali iterasi. Tidak ada ketentuan pasti terkait jumlah grup semut untuk

menghasilkan nilai optimal. Sehingga dibutuhkan beberapa uji coba agar mendapat hasil yang terbaik.

- Indeks tingkat feromon (α) dan indeks *visibility* (β)

  Nilai α merupakan bobot *pheromone* τ dan β adalah bobot yang mengontrol *visibility* terhadap tingkat *pheromone* τ. Untuk memudahkan perhitungan, nilai kedua indeks ini disamakan yaitu 1.
- Jumlah kandidat

  Untuk memudahkan pemilihan *node* berikutnya untuk masing-masing kendaraan, maka dibuat kandidat *node* terpilih. Untuk kandidat *node* yang sudah dipilih oleh satu kendaraan, tidak akan bisa dipilih lagi oleh kendaraan yang lain.
- Iterasi maksimal

  Jumlah iterasi maksimum bersifat tetap selama algoritma berjalan. Untuk nilai optimal suatu iterasi maksimal, perlu dilakukan beberapa uji coba agar mendapa hasil yang terbaik.

b) Probabilitas dan pemilihan *node* berikutnya

Perhitungan probabilitas untuk memilih ruas menggunakan rumus $P = \frac{\tau_{ij}^{\alpha} \times h_{ij}^{\beta}}{\sum \tau_{ij} \times h_{ij}}$

Untuk masing-masing ruas ditentukan range *probabilitas* kumulatif yang berkaitan dengan pilihan ruas. Selanjutnya dipilih nilai tertentu berdasarkan bilangan random dalam *range* (0,1). Ruas tertentu akan terpilih berdasarkan bilangan random yang dibangkitkan.

Setelah setiap semut menempati masing – masing titik yang ditentukan, maka semut akan mulai melakukan perjalanan dari titik pertama masing-masing sebagai titik asal dan menuju salah satu titik - titik lainnya sebagai tujuan. Kemudian dari titik kedua masing-masing, semut akan melanjutkan perjalanan dengan memilih salah satu dari titik – titik yang belum dikunjungi sebagai titik tujuan selanjutnya. Titik yang sudah dilayani akan disimpan dalam suatu *tabu list* agar tidak dikunjungi lagi. Perjalanan semut berlangsung terus menerus sampai semua titik dikunjungi satu persatu.

c) Perhitungan waktu tempuh

Apabila suatu semut telah menyelesaikan seluruh rute dan kembali ke depot, maka dilakukan perhitungan total waktu tempuh, mulai dari depot ke sungai, sungai ke

taman, taman ke taman, dan taman ke depot. Nilai total waktu tempuh disimpan dalam memori sebagai local optimal (lk)

d) *Adjust* feromon

Setelah semua grup dijalankan, dilakukan adjust nilai feromon

$\tau_{n+1} = (1-\rho)\tau_n + \rho\tau_0$

e) Lakukan iterasi

Untuk setiap iterasi yang dilakukan, tabu *list* dikosongkan terlebih dahulu untuk diisi kembali dengan urutan titik yang baru di iterasi berikutnya. Kemudian dilakukan perulangan untuk tahapan (b) sampai dengan (d) hingga mencapai iterasi maksimum atau telah mencapai konvergensi. Dari iterasi keseluruhan tersebut akan didapat nilai global optimal total waktu tempuh minimum.

2. Pembagian Sub Rute VRP

Berikut ini merupakan algoritma yang dilakukan dalam mencari rekomendasi rute untuk penyiraman taman kota di Surabaya. Hasil dari *Ant Colony Optimization* TSP akan digunakan sebagai dasar pembuatan sub rute VRP yang sudah disesuaikan dengan *constraints* yang ada. Koding lengkap dapat dilihat pada Lampiran.

a. Inisialisasi

- Data yang digunakan sebagai input pada sub rute VRP ini adalah:
  1. Rute TSP optimum dari ACO TSP sebelumnya
  2. Waktu tempuh dari depot menuju taman
  3. Waktu tempuh dari taman ke taman
  4. Waktu tempuh dari taman ke sungai
  5. Waktu tempuh dari depot ke sungai
  6. Waktu operasional pengambilan air
  7. Waktu operasional penyiraman
  8. Jumlah truk
  9. *Demand* taman
  10. Kapasitas truk
  11. Jam akhir *shift*
- Inisialisasi truk yang akan digunakan dan taman yang dikunjungi

b. Pemilihan truk

Pada kasus ini, truk yang digunakan berjumlah 9 unit dengan ukuran kapasitas tangki 5000 Liter. Dari total 9 truk ini akan dilakukan pemilihan kandidat jumlah taman yang bisa

dikunjungi pertama kali. Untuk tiap taman yang sudah dipilih menjadi kandidat oleh satu truk, tidak bisa dipilih oleh truk lain. Proses pemilihan kandidat taman yang akan dikunjungi adalah dengan membagi jumlah taman menjadi 4 bagian. Dengan jumlah taman sebanyak 46, maka ada 12 kandidat taman. Tujuan dilakukannya pemilihan kandidat adalah agar arah kendaraan tersebar.

c. Pemilihan titik awal

Pemilihan titik awal untuk masing-masing skenario selalu dimulai dari depot. Untuk setiap truk di titik awal memiliki kapasitas = 0. Setiap truk akan kembali ke titik awal setelah selesai melakukan semua taman dikunjungi dan batas waktu jam operasional sudah habis.

d. Pemilihan taman

Pada penelitian ini terdapat 46 taman yang harus dikunjungi. Tiap taman memiliki *demand* yang berbeda-beda tergantung luasan taman. Tidak ada batasan waktu untuk kunjungan ke taman atau dengan kata lain taman bebas dikunjungi kapan saja. Pemilihan taman berkaitan dengan waktu tempuh terkecil dari posisi terakhir truk dengan mempertimbangkan kapasitas dan waktu yang tersisa.

e. Pemilihan titik penyiraman

Pada pemilihan titik penyiraman terdapat 2 skenario yang dijalankan. Untuk skenario 1, pemilihan titik penyiraman hanya mempertimbangkan waktu tempuh ke dari titik awal dengan titik penyiraman untuk menentukan titik mana yang akan dikunjungi berikutnya. Pemilihan titik ini mengikuti rute yang dihasilkan oleh TSP ACO. Kemudian dilihat waktu tempuh dan kapasitas, apakah memungkinkan untuk mengunjungi titik tersebut. Apabila kapasitas truk tidak memenuhi *demand* di titik penyiraman, maka dilakukan kunjungan ke titik pengisian air terdekat. Namun jika waktu yang tidak memenuhi maka truk akan kembali ke depot. Dalam proses ini memungkinkan untuk terjadi *split service*. Kondisi *split* terjadi apabila truk yang datang awal hanya bisa memenuhi *demand* taman sebagian saja dan sisa *demand* yang belum terpenuhi akan dilakukan oleh truk lain jika truk yang datang awal sudah mulai mendekati batasan akhir waktu kerja.

Untuk scenario 2, pemilihan titik penyiraman selain mempertimbangkan waktu tempuh juga mempertimbangkan *saving weight*. Untuk pemilihan titik kunjungan berikutnya, awalnya ditentukan terlebih dahulu titik-titik terdekat yang potensial berdasarkan waktu tempuh, kemudian dibuat list *saving value* dari masing-masing kandidat titik potensial mulai dari yang terbesar hingga terkecil. Titik yang memiliki waktu terpendek dan saving value *tertinggi* yang akan dipilih untuk kunjungan berikutnya.

f. Pemilihan titik pengambilan air

Kunjungan ke titik pengambilan air merupakan *soft constraint*, sehingga kunjungan ini dilakukan apabila kapasitas truk = 0. Tidak ada proses yang spesifik dalam pemilihan titik pengisian air, karena hanya mempertimbangkan waktu tempuh terkecil dari lokasi truk terakhir dan titik tujuan penyiraman berikutnya.

5.3. Verifikasi dan Validasi

Verifikasi dan validasi model adalah proses pengujian untuk menentukan model yang telah dibuat apakah benar-benar merepresentasikan permasalahan *real* yang dimodelkan. Proses ini dilakukan agar algoritma yang disusun bisa menyelesaikan permasalahan sehingga solusi yang dihasilkan tidak melanggar *constraint* yang telah ditentukan.

Proses verifikasi dan validasi dilakukan dengan melakukan percobaan terhadap model dengan sampel kecil sesuai dengan scenario yang telah dibuat. Pada penelitian ini terdapat 2 skenario yang diujikan

5.4.1. Verifikasi dan Validasi Model pada Kondisi 1

Pada kondisi ini diasumsikan *demand* untuk tiap taman tidak melebihi kapasitas truk, sehingga hanya memerlukan 1 ritase untuk kunjungan di masing-masing titik. Data sampel yang digunakan untuk titik penyiraman sebanyak 10 titik. Jumlah kendaraan dialokasikan sebanyak 9 unit dengan kapasitas sebanyak 5000 Liter. Data input untuk kondisi 1 dapat dilihat pada Tabel 5.1.

Tabel 5. 1 Data Input pada Kondisi 1

No	Lokasi penyiraman	*Demand*	No	Lokasi Pengambilan air
2	kendalsari wonorejo	1250	12	Sungai depan Stikom
3	raya pandugo	7500	13	Selokan Depan Perumahan YKP Pandugo I
4	kedung baruk timur	2700	14	Selokan Depan Samator
5	mer kedung asem	2250	15	Selokan Dekat Pizza Hut Merr
6	mer gunung anyar	1050	16	sungai dekat hotel narita
7	bratang rmi	1500		
8	ngagel jaya	5000		
9	ngagel jaya selatan	3750		
10	pucang	2625		
11	kertajaya	4125		

Berdasarkan hasil *running* program untuk kondisi 1, didapat rute dan jumlah kendaraan yang dibutuhkan untuk melayani semua titik pada kondisi ini. Untuk 10 titik penyiraman, hanya diperlukan 1 kendaraan untuk mengunjungi semua titik. Total waktu

perjalanan yang dibutuhkan adalah 390 menit, sehingga masih memenuhi batasan waktu jam operasional *shift*.

Tabel 5. 2 Hasil Perutean Truk untuk Kondisi 1

Truk	Rute Ke -																									Waktu per Kendaraan	Total Waktu
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25		
1	1	16	7	9	16	10	11	16	9	11	8	16	8	6	5	13	3	13	3	2	5	4	14	4	1	379	379
2	1																									0	379
3	1																									0	379
4	1																									0	379
5	1																									0	379
6	1																									0	379
7	1																									0	379
8	1																									0	379
9	1																									0	379

5.4.2. Verifikasi dan Validasi Model pada Kondisi 2

Pada kondisi ini menggunakan *demand* yang sebenarnya untuk tiap taman, dimana ada beberapa *demand* yang melebihi kapasitas truk, sehingga memerlukan lebih dari 1 ritase untuk kunjungan di masing-masing titik. Kondisi ini juga memungkinkan untuk terjadinya *split service*. Data sampel yang digunakan untuk titik penyiraman sebanyak 10 titik. Jumlah kendaraan dialokasikan sebanyak 9 unit dengan kapasitas sebanyak 5000 Liter. Data input untuk kondisi 1 dapat dilihat pada Tabel 5.3.

Tabel 5. 3 Data Input pada Kondisi 2

No	Lokasi penyiraman	*Demand*	No	Lokasi Pengambilan air
2	kendalsari wonorejo	11250	12	Sungai depan Stikom
3	raya pandugo	7500	13	Selokan Depan Perumahan YKP Pandugo I
4	kedung baruk timur	27000	14	Selokan Depan Samator
5	mer kedung asem	22500	15	Selokan Dekat Pizza Hut Merr
6	mer gunung anyar	10500	16	sungai dekat hotel narita
7	bratang rmi	1500		
8	ngagel jaya	5250		
9	ngagel jaya selatan	3750		
10	pucang	2625		
11	kertajaya	4125		

Berdasarkan hasil *running* program untuk kondisi 2, output yang dihasilkan berupa rute dan jumlah kendaraan yang dibutuhkan untuk melayani semua titik pada kondisi ini. Untuk 10 titik penyiraman, jumlah kendaraan yang dibutuhkan adalah 3 unit, meningkat 3 kali lipat dari kondisi sebelumnya. Terdapat beberapa taman yang mengalami *split service*, seperti pada taman 2, awalnya dikunjungi oleh truk 1. Setelah dilakukan penyiraman sebanyak 1 ritase oleh truk 1, terdapat sisa *demand* yang belum terpenuhi sebanyak 2500 Liter. Untuk *demand* sisa yang belum dipenuhi, selanjutnya di *cover* oleh truk 2.

Tabel 5. 4 Hasil Perutean Truk untuk Kondisi 2

Truk	Rute Ke -																									Waktu per Kendaraan	Total Waktu
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25		
1	1	16	4	14	4	14	4	14	4	14	4	14	4	5	13	5	13	5	13	3	13	5	13	5	1	474	1012
2	1	13	3	2	13	2	13	2	6	13	6	13	6	8	16	10	11	16	8	7	16	9	7	1		480	1012
3	1	6	11	1																						58	1012
4	1																									0	1012
5	1																									0	1012
6	1																									0	1012
7	1																									0	1012
8	1																									0	1012
9	1																									0	1012

5.4. Percobaan Model Algoritma

Setelah model di validasi dan di verifikasi dengan sampel data kecil, maka model siap digunakan untuk data yang lebih besar, yaitu untuk perutean penyiraman taman kota di Surabaya. Terdapat 2 skenario yang akan diuji coba dengan beberapa parameter, dimana masing-masing skenario akan di running sebanyak beberapa kali replikasi agar mendapat hasil yang terbaik. Dalam percobaan model, semakin banyak jumlah replikasi yang dilakukan akan menghasilkan nilai yang lebih bagus. Namun ada beberapa hal terkait waktu komputasi yang menjadi bahan pertimbangan, sehingga dipilih 10 kali replikasi dalam percobaan model di penelitian ini. Untuk parameter yang menghasilkan nilai terbaik, selanjutnya dilakukan *running* model sebanyak 10 replikasi untuk mendapat nilai yang mendekati optimal.

5.4.1. Parameter Model Algoritma untuk Rute Penyiraman Taman Kota di Surabaya

Pada penelitian ini pencarian rute penyiraman taman kota di Kota Surabaya menggunakan algoritma ACO. Di dalam algoritma ACO mengandung nilai random sehingga dalam penentuan nilai beberapa parameter perlu dilakukan beberapa kali perbandingan nilai parameter dalam beberapa replikasi untuk menentukan nilai parameter mana yang terbaik untuk digunakan dalam pencarian rute. Parameter yang akan diujikan adalah:

1. Evaporasi

   Nilai evaporasi yang akan diuji adalah $\rho = (0{,}3, 0.6, 0.9)$
2. Iterasi

   Nilai iterasi yang akan diujikan adalah i = 10, 50, dan 100
3. Jumlah grup semut

   Jumlah grup semut yang akan diujikan adalah 5 dan 10.
4. Jumlah Replikasi

   Untuk mengurangi variansi maka simulasi harus dilakukan sebanyak n kali replikasi. Untuk mendapatkan nilai n maka perlu dilakukan repliasi awal n0 yaitu sebanyak 10 kali replikasi.

Pengujian semua algoritma dalam penelitian ini akan dilakukan dengan *software Jupyter Notebook* dengan spesifikasi komputer: Intel® core™ i5-2450m CPU @2.50GHZ RAM 4 GB. Setiap kombinasi nilai parameter akan diujikan sebanyak 10 replikasi. Selanjutnya untuk setiap 10 replikasi akan dicari nilai total jarak minimum, rata-rata total jarak dan standart deviasinya. Rangkuman dari hasil pencarian nilai parameter terbaik dapat dilihat pada Tabel 5.5

Tabel 5. 5 Uji Coba Parameter

ρ	m	i	Min	Mean	SD	ρ	m	i	Min	Mean	SD
0.3	5	10	3890	3910,97	9,50	0.6	5	10	3727,1	3739,96	11,54
			9	9	0				8	8	0
			404	471,45	66,302				358	421,27	37,37
		50	3634,85	3647,63	12,18			50	3639,99	3647,37	6,615602
			8	8	0				8	8	0
			1552	1661,64	120,77				1843	1889,636	49,28026
		100	3639,55	3644,664	3,784			100	3595,65	3626,536	14,28308
			8	8	0				8	8	0
			4080	4255,64	110,583				3200	3362,727	146,0327
	10	10	3657,8	3688,15	16,78		10	10	3681,06	3696,656	10,84195
			8	8	0				8	8	0
			1034	1039,27	5,16				625	676,3636	66,40992
		50	3611,9	3630,43	14,45			50	3611,5	3621,325	6,948727
			8	8	0				8	8	0
			3251	3469,18	123,05				3060	3400,636	236,2852
		100	3618,05	3621,527	2,496804			100	3623,45	3624,623	1,006865
			8	8	0				8	8	0
			5353	5852,091	369,3733				6405	6466,364	45,41476
0,9	5	10	3635,5	3659,4	12,8	0,9	10	10	3651,25	3665,077	9,629492
			8	8	0				8	8	0
			1553,0	2045,4	696,6				501	582,1818	42,6694
		50	3635,54	3650,45	13,3189			50	3246,65	3489,7	174,1273
			8	8	0				8	8	0
			1469	1492,45	17,2356				2986	3071,718	67,22943
		100	3620,25	3639,98	16,0031			100	3628,05	3635,57	5,30153
			8	8	0				8	8	0
			2981	3206,36	177,094				6386	6492,36	78,7189

Berdasarkan tabel 5.5 dapat diketahui bahwa:

1. Terkait waktu komputasi, apabila semakin besar jumlah grup semut maupun jumlah iterasinya maka waktu komputasi secara nilai rata-rata secara konstan yang dibutuhkan akan semakin tinggi. Sedangkan semakin besar nilai evaporasi tidak secara konstan mempengaruhi tinggi rendahnya waktu yang dibutuhkan, sesuai dengan contoh sampel kecil perbandingan waktu.
2. Terkait dengan hasil fungsi tujuan yaitu total waktu tempuh, diketahui bahwa nilai minimum dan nilai rata-rata untuk parameter koefisien evaporasi (ρ), jumlah grup semut (m) dan iterasi (i), total waktu tempuh terkecil dimiliki oleh parameter ρ dengan

nilai 0,6 dan jumlah grup semut 5 pada iterasi sebanyak 50, dengan nilai waktu tempu minimum adalah 3611,5 detik dan rata-rata waktu tempuh minimum 3621,3 detik.

Dari uji coba dan pertimbangan tersebut maka diputuskan pada penelitian ini menggunakan parameter koefisien evaporasi $\rho = 0.6$, jumlah grup semut m = 5 dan iterasi i = 50.

Untuk perhitungan jumlah replikasi minimum yang dibutuhkan, ditunjukkan pada tabel:

Tabel 5. 6 Hasil Perhitungan Replikasi Minimum

No	Data
1	3572,55
2	3578,55
3	3569,55
4	3558,6
5	3588,56
6	3885,55
7	3930,5
8	3579,55
9	3561,55
10	3577,5
Jumlah	36402,46
Rata rata	3640,246
Stdev	141,798
$t_{n-1,\frac{\propto}{2}}$	2,685011
$(Z_{\propto/2)S}$	1,959964
hw	120,3972
n’	2,570477

Dari perhitungan diatas, nilai replikasi minimum yang dibutuhkan adalah 3 kali replikasi dan sudah menghasilkan nilai yang terbaik.

5.4.2. Hasil *Running* Model Algoritma untuk Rute Penyiraman Taman Kota di Surabaya

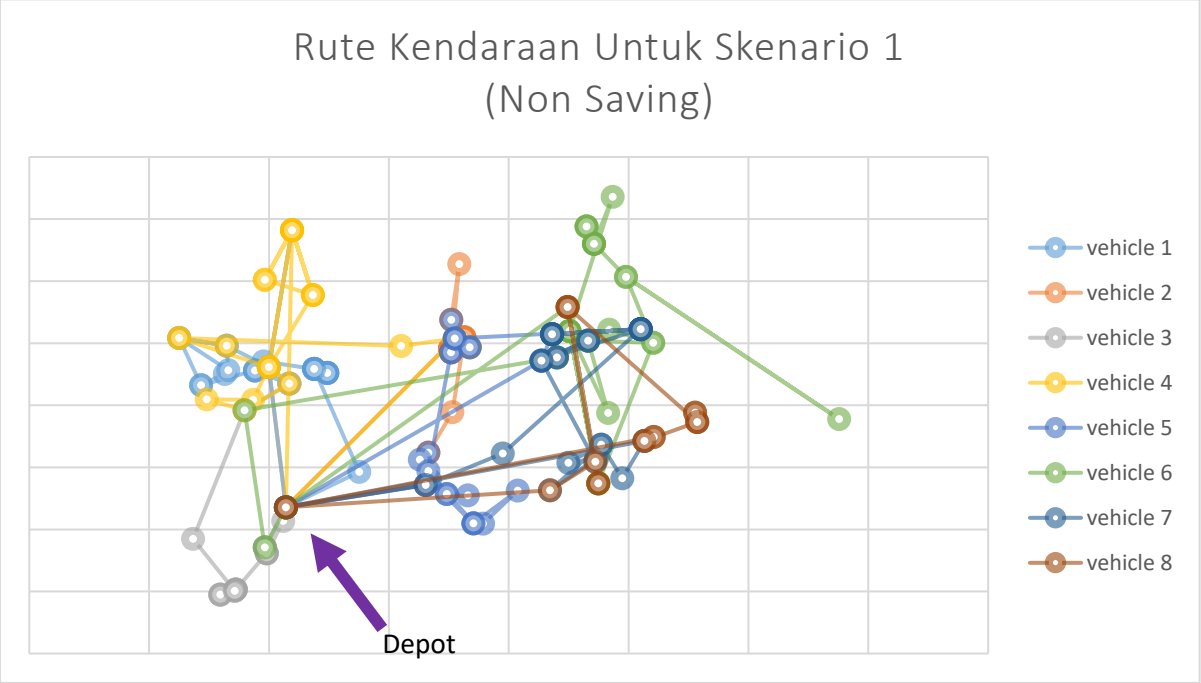
Running model algoritma untuk rute penyiraman taman di Kota Surabaya dengan algoritma ACO menggunakan parameter yang sesuai dengan keputusan sub bab sebelum ini yaitu nilai evaporasi 0.6, jumlah semut 5 dan maksimal iterasi 50. Parameter tersebut digunakan untuk *running* model skenario 1, skenario 2, skenario 3, dan skenario 4. Untuk semua skenario *running* model dilakukan sebanyak 10 replikasi dan dipilih nilai terbaik dari 10 replikasi tersebut sebagai nilai fungsi tujuan yang didapatkan oleh model algoritma ACO. Berikut hasil *running* model untuk Skenario 1.

Tabel 5. 7 Hasil *Running* 10 Replikasi Model Algoritma Skenario 1

No	Fungsi Tujuan (menit)	Jumlah Kendaraan (unit)	Waktu (detik)	No	Fungsi Tujuan (menit)	Jumlah Kendaraan (unit)	Waktu (detik)
1	3572,55	8	3503	6	3885,55	8	3851
2	3578,55	8	4065	7	3930,5	8	3937
3	3569,55	8	4607	8	3579,55	8	4792
4	3558,6	8	3840	9	3561,55	8	5100
5	3588,56	8	4637	10	3577,5	8	4689

Nilai minimum dari 10 replikasi yang dijalankan ada pada replikasi ke-4 dengan nilai fungsi tujuan sebesar 3558,6 menit. Hasil detail dari replikasi ke-4 sebagai replikasi yang memiliki nilai fungsi tujuan minimum ditunjukkan pada Tabel 5.7. Untuk plot visual rute yang dihasilkan ditunjukkan pada Gambar 5.3.

Gambar 5. 3 Plot Rute Kendaraan untuk Skenario 1



Tabel 5. 8 Rute Truk Skenario 1 Hasil *Running* Model

Truk	Rute ke																														Waktu per Kendaraan	Total Waktu
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**		
1	1	52	4	50	4	50	4	50	4	50	4	50	4	30	27	71	27	71	27	29	74	29	6	70	28	6	70	5	1				468	3558,55
2	1	53	45	53	45	85	41	86	41	86	42	86	41	86	41	86	41	47	84	44	47	84	42	82	42	1							465	3558,55
3	1	79	39	37	79	37	79	37	79	37	38	81	36	81	36	81	36	81	38	40	26	71	5	49	5	1							441	3558,55
4	1	49	3	49	3	2	49	2	49	2	5	49	5	73	31	26	71	5	6	70	6	46	42	82	42	1							476	3558,55
5	1	53	45	7	8	55	8	10	56	11	10	54	10	9	47	84	47	42	82	42	82	42	43	76	43	76	43	76	35	64	35	1	480	3558,55
6	1	79	26	18	64	19	33	83	33	83	34	64	22	34	23	65	23	25	67	25	67	25	67	25	24	33	83	33	32	1			476	3558,55
7	1	53	14	43	76	43	76	43	76	43	76	43	76	43	35	64	18	64	18	35	21	59	15	13	12	59	20	17	1				459	3558,55
8	1	79	37	79	37	79	39	37	79	37	38	81	38	36	81	36	81	36	1														293,55	3558,55

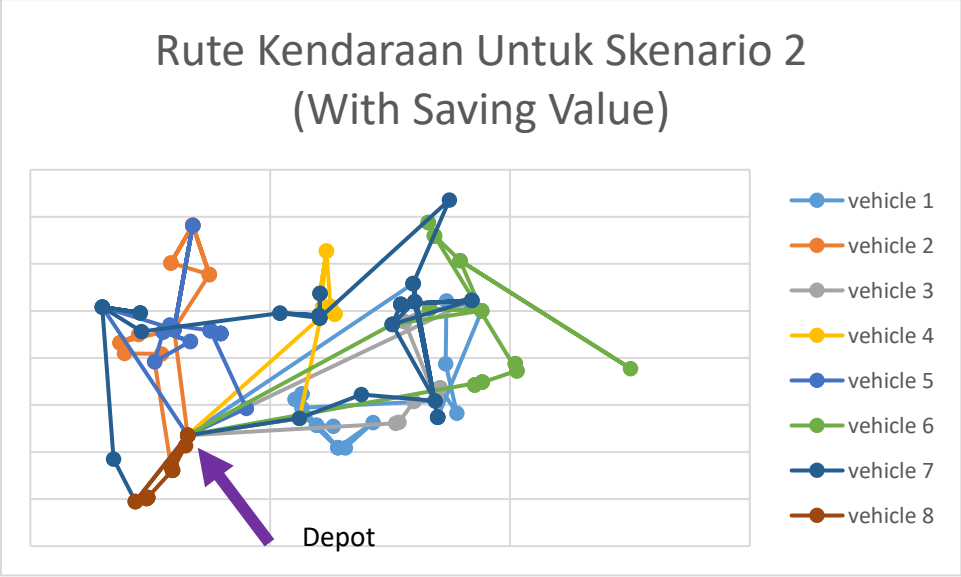
Sama halnya dengan Skenario sebelumnya, skenario ini di-*running* menggunakan parameter nilai evaporasi 0.6, jumlah semut 5 dan maksimal iterasi 50 sebanyak 10 kali replikasi. Hasil yang didapatkan dijelaskan di Tabel 5.8.

Tabel 5. 9 Hasil *Running* 10 Replikasi Model Algoritma Skenario 2

No	Fungsi Tujuan (menit)	Jumlah Kendaraan (unit)	Waktu (detik)	No	Fungsi Tujuan (menit)	Jumlah Kendaraan (unit)	Waktu (detik)
1	3582,55	8	4604	6	3506,55	8	5100
2	3571,55	8	4664	7	3529,55	8	5060
3	3575,6	8	4865	8	3508,55	8	4292
4	3530,55	8	5147	9	3533,55	8	4167
5	3519,55	8	4390	10	3544,55	8	4219

Nilai minimum dari 10 replikasi yang dijalankan ada pada replikasi ke-6 dengan nilai fungsi tujuan sebesar 3506,55 menit. Hasil detail dari replikasi ke-6 sebagai replikasi yang memiliki nilai fungsi tujuan minimum ditunjukkan pada Tabel 5.9. Untuk plot visual rute yang dihasilkan ditunjukkan pada Gambar 5.4.

Gambar 5. 4 Plot Rute Kendaraan untuk Skenario 2



Tabel 5. 10 Rute Truk Skenario 2 Hasil *Running* Model

Truk	Rute ke																																Waktu per Kendaraan	Total Waktu
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34		
1	1	53	45	53	45	53	45	7	8	55	8	10	56	11	10	54	10	9	21	59	20	19	22	34	83	33	83	32	83	32	1				454	3506,55
2	1	79	26	73	31	29	74	29	5	49	5	49	5	49	3	49	3	2	49	2	49	2	5	1											457	3506,55
3	1	57	12	15	13	59	13	18	64	35	64	18	35	76	43	76	43	76	43	76	43	76	43	76	43	76	43	1							478	3506,55
4	1	53	42	82	42	82	42	82	42	82	42	82	42	41	86	41	86	41	86	41	86	41	44	84	44	47	1								461	3506,55
5	1	52	4	50	4	50	4	50	4	50	4	50	4	30	27	71	27	71	26	27	5	49	5	49	5	6	1								445	3506,55
6	1	61	17	61	17	61	17	16	63	16	63	16	25	67	25	67	25	67	25	23	65	23	34	64	34	35	1								451	3506,55
7	1	53	14	21	35	76	43	76	43	35	33	83	33	32	83	32	24	47	84	47	84	47	46	86	46	28	6	70	6	70	6	40	36	1	470	3506,55
8	1	79	37	79	37	79	39	37	79	37	38	81	38	36	81	36	81	36	1																290,55	3506,55

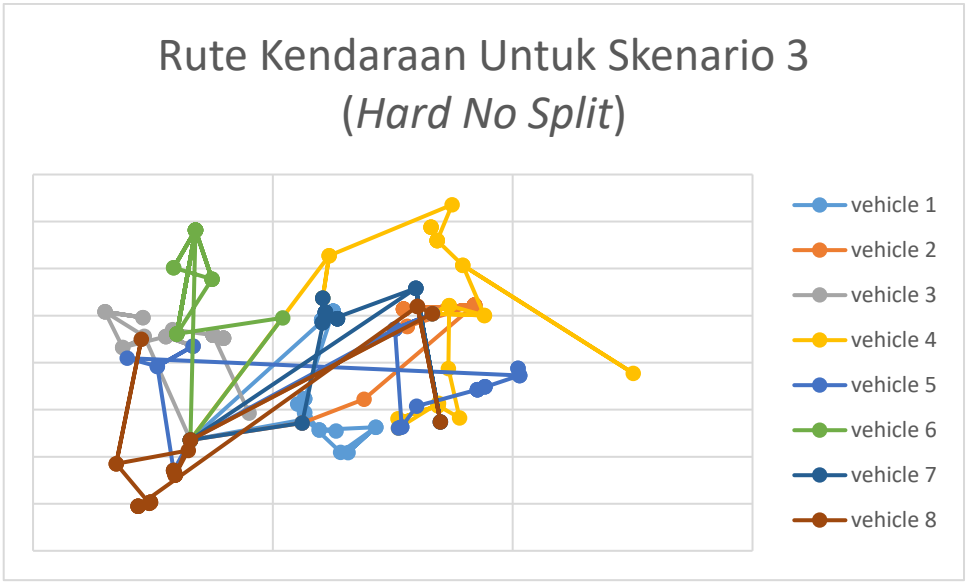
Untuk skenario 3, skenario ini di-*running* menggunakan parameter nilai evaporasi 0.6, jumlah semut 5 dan maksimal iterasi 50 sebanyak 10 kali replikasi. Hasil yang didapatkan dijelaskan di Tabel 5.10.

Tabel 5. 11 Hasil *Running* 10 Replikasi Model Algoritma Skenario 3

No	Fungsi Tujuan (menit)	Jumlah Kendaraan (unit)	Waktu (detik)	No	Fungsi Tujuan (menit)	Jumlah Kendaraan (unit)	Waktu (detik)
1	3567,55	9	4517	6	3576,55	9	4712
2	3571,55	8	4712	7	3564,55	8	4630
3	3552,55	8	4684	8	3582,55	9	4719
4	3562,55	8	4621	9	3570,55	8	4361
5	3574,55	8	4623	10	3599,55	8	4572

Nilai minimum dari 10 replikasi yang dijalankan ada pada replikasi ke-3 dengan nilai fungsi tujuan sebesar 3552,55 menit. Hasil detail dari replikasi ke-3 sebagai replikasi yang memiliki nilai fungsi tujuan minimum ditunjukkan pada Tabel 5.10. Untuk plot visual rute yang dihasilkan ditunjukkan pada Gambar 5.5.

Gambar 5. 5 Plot Rute Kendaraan Skenario 3



Tabel 5. 12 Rute Truk Skenario 3 Hasil *Running* Model

Truk	Rute ke																															Waktu per Kendaraan	Total Waktu
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**		
1	1	53	45	53	45	53	45	14	43	76	43	76	43	76	43	76	43	76	43	76	43	76	43	76	43	1							462,75	3552,55
2	1	54	10	11	56	11	12	13	59	13	20	19	64	22	34	64	34	23	65	23	25	67	25	67	25	67	25	24	44	84	44	1	460,25	3552,55
3	1	52	4	50	4	50	4	50	4	50	4	50	4	30	27	71	27	71	27	26	71	26	31	74	31	29	74	29	1				462,25	3552,55
4	1	49	3	49	3	2	49	2	49	2	5	49	5	49	5	49	5	49	5	49	5	28	1										448,1	3552,55
5	1	55	8	55	8	7	9	54	9	47	84	47	42	82	42	82	42	82	42	82	42	82	42	82	42	46	1						421	3552,55
6	1	64	33	83	33	32	83	32	83	32	83	32	16	63	16	63	16	17	61	17	61	17	52	39	1								438,2	3552,55
7	1	58	18	64	18	35	64	35	64	35	64	35	21	15	59	15	40	36	81	36	81	36	81	36	38	81	38	1					432	3552,55
8	1	53	41	86	41	86	41	86	41	86	41	6	70	6	70	6	37	79	37	79	37	79	37	1									428	3552,55

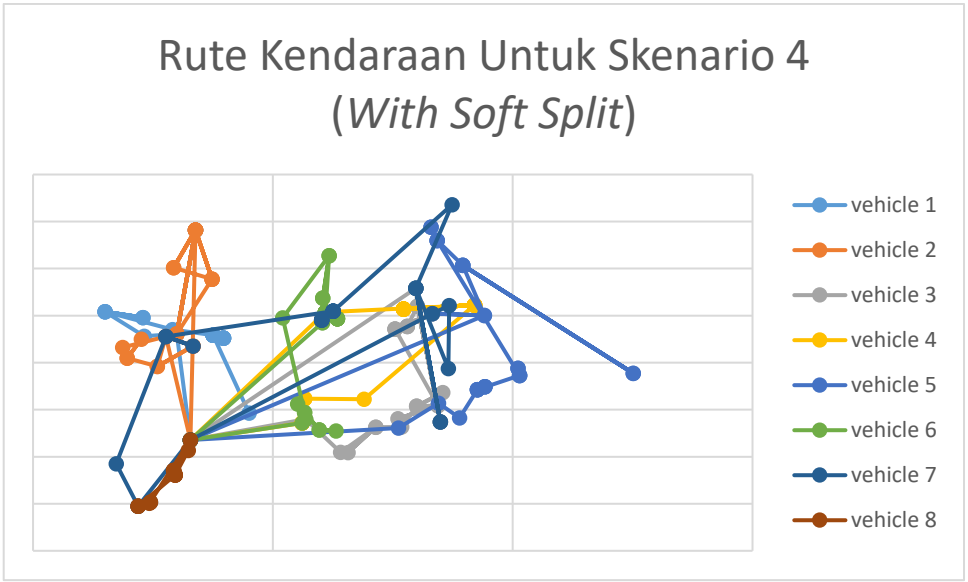
Untuk skenario 4, skenario ini di-*running* menggunakan parameter nilai evaporasi 0.6, jumlah semut 5 dan maksimal iterasi 50 sebanyak 10 kali replikasi. Hasil yang didapatkan dijelaskan di Tabel 5.12.

Tabel 5. 13 Hasil *Running* 10 Replikasi Model Algoritma Skenario 4

No	Fungsi Tujuan (menit)	Jumlah Kendaraan (unit)	Waktu (detik)	No	Fungsi Tujuan (menit)	Jumlah Kendaraan (unit)	Waktu (detik)
1	3496,55	8	4781	6	3507,55	8	4711
2	3533,55	8	4818	7	3507,55	8	4632
3	3514,55	8	4712	8	3499,55	8	4645
4	3508,55	8	4762	9	3518,55	8	4596
5	3521,55	8	4621	10	3499,55	8	4530

Nilai minimum dari 10 replikasi yang dijalankan ada pada replikasi ke-1 dengan nilai fungsi tujuan sebesar 3496,55 menit. Hasil detail dari replikasi ke-1 sebagai replikasi yang memiliki nilai fungsi tujuan minimum ditunjukkan pada Tabel 5.12. Untuk plot visual rute yang dihasilkan ditunjukkan pada Gambar 5.6.

Gambar 5. 6 Plot Rute Kendaraan Skenario 4



Tabel 5. 14 Rute Truk Skenario 4 Hasil *Running* Model

Truk	Rute ke																														Waktu per Kendaraan	Total Waktu
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30		
1	1	52	4	50	4	50	4	50	4	50	4	50	4	30	6	70	6	70	6	28	5	49	5	49	5	49	5	1			469	3496,55
2	1	49	3	49	3	2	49	2	49	2	5	49	5	49	5	29	31	74	31	26	71	26	27	71	27	1					474	3496,55
3	1	54	10	11	56	11	12	15	59	15	21	58	21	35	64	35	64	35	18	64	18	64	18	33	83	33	83	33	32	1	449	3496,55
4	1	53	45	53	45	53	45	14	43	76	43	76	43	76	43	76	43	76	43	76	43	76	43	76	43	42	1				455	3496,55
5	1	57	13	20	17	61	17	61	17	61	17	16	63	16	63	16	25	67	25	67	25	23	65	23	65	23	34	64	34	1	448	3496,55
6	1	53	7	8	55	8	9	54	9	46	44	84	44	42	82	42	82	42	82	42	82	42	82	42	82	42	47	84	47	1	469	3496,55
7	1	64	22	19	32	83	32	83	32	24	41	86	41	86	41	86	41	86	41	86	41	27	71	27	40	36	81	36	1		479	3496,55
8	1	79	39	37	79	37	79	37	79	37	36	81	36	81	36	38	1														253,55	3496,55

5.5.Penugasan Truk Tangki untuk Rute Penyiraman Taman Kota di Surabaya

Berdasarkan rute rekomendasi yang dihasilkan pada tahap sebelumnya selanjutnya dilakukan penugasan kendaraan. Apabila tidak ada kebijakan atau pertimbangan mengenai hal lain lagi yang diambil maka kendaraan yang dibutuhkan adalah sebanyak 8 unit pada skenario 1 dan skenario 2. Penugasan truk dan rute perjalanan truk hasil rekomendasi dapat dilihat pada Tabel 5.7 dan 5.9.

Dari total kendaraan yang tersedia pada DKRTH Surabaya Rayon Timur saat ini terdapat 9 unit, rekomendasi hasil penelitian ini hanya membutuhkan 8 unit kendaraan. Selisih dari keduanya adalah 1 unit. Sisa kendaraan ini nantinya akan digunakan sebagai mobil cadangan apabila terjadi peningkatan jumlah pekerja ataupun pada saat ada kendaraan yang *down* dan jika mengunakan jasa subkontraktor akan mengurangi biaya penyewaan dan pengadaan truk. Hal tersebut sangat penting bagi DKRTH Surabaya Rayon Timur karena pada saat-saat tertentu terjadi kerusakan pada truk yang membuat truk tidak bisa melakukan kegiatan penyiraman taman. Disamping itu setiap harinya taman harus selalu disiram karena jika tidak akan menganggu keberlangsungan hidup tanaman sebagai ruang hijau di Kota Surabaya.

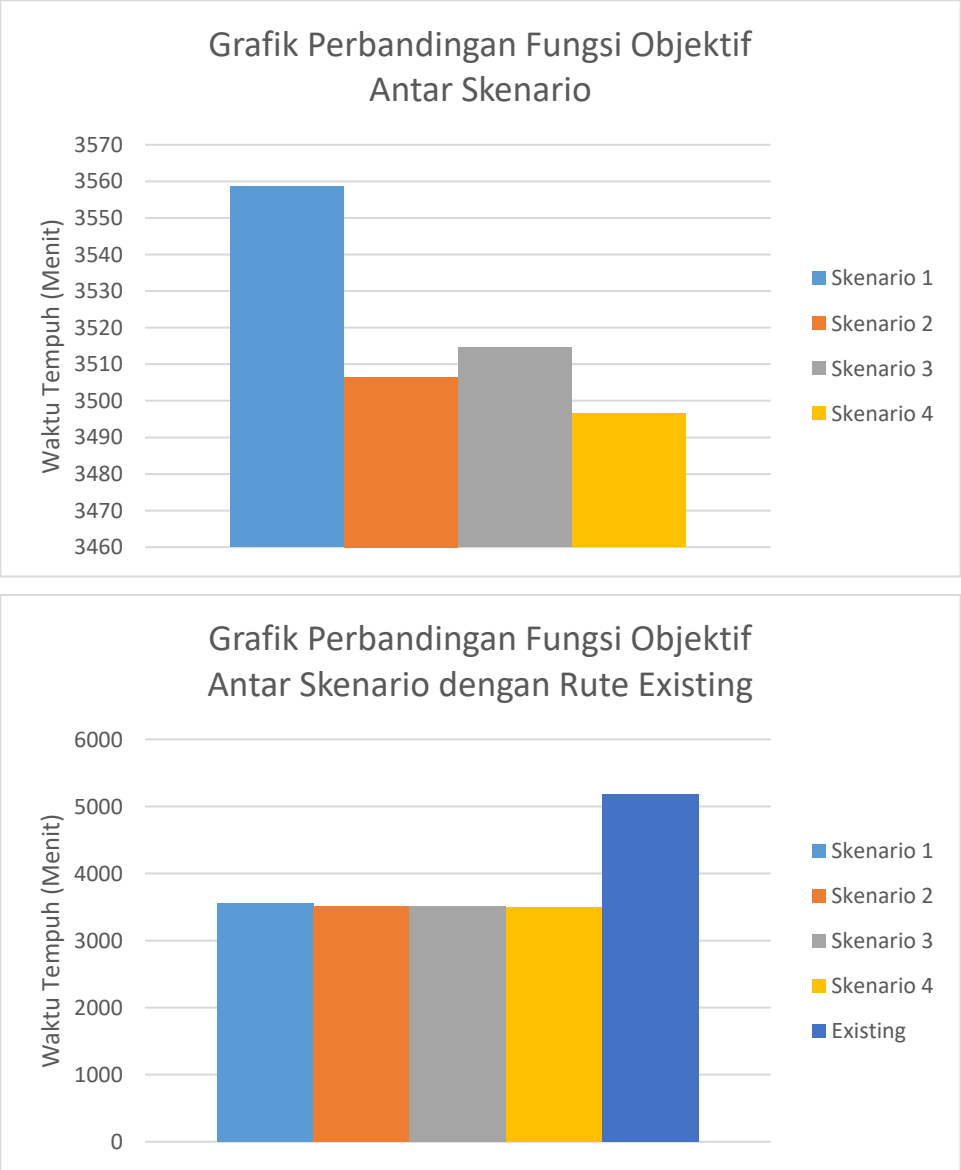
5.6.Analisis Hasil Penelitian

Parameter yang digunakan dalam uji model ACO untuk penyiraman taman di Kota Surabaya diperoleh dari uji coba beberapa parameter dengan 10 replikasi pada setiap kombinasinya. Hasil nya parameter yang terpilih merupakan parameter yang memiliki standart deviasi kecil dapat memberikan nilai fungsi tujuan yang minimum. Rekomendasi rute yang dihasilkan oleh model algoritma ACO dengan parameter nilai evaporasi 0.6, jumlah semut 10 dan iterasi maksimal 100 untuk penyiraman taman di Kota Surabaya menghasilkan nilai fungsi tujuan sebesar 3558,6 menit untuk skenario 1, 3506,55 menit untuk skenario 2, 3514,55 menit untuk skenario 3 dan 3496,55 menit untuk skenario 4. Nilai fungsi tujuan tersebut didapatkan dengan menggunakan jumlah kendaraan sebesar 8 unit. Rute tersebut dihasilkan dalam waktu komputasi sebesar 3890 detik yang merupakan replikasi ke-6 dari 10 replikasi untuk skenario 1, waktu komputasi sebesar 5100 detik yang merupakan replikasi ke-4 dari 10 replikasi untuk skenario 2, waktu komputasi sebesar 4623 detik yang merupakan replikasi ke-5 dari 10 replikasi untuk skenario 3, dan waktu komputasi sebesar 4781 detik yang merupakan replikasi ke-1 dari 10 replikasi untuk skenario 4. Jumlah kendaraan yang digunakan masih dibawah jumlah kendaraan yang dimiliki oleh DKRTH pada saat ini. Sehingga dengan *demand* setiap taman akan dapat terlayani semuanya dan juga masih memiliki truk tangki cadangan yang akan *stay* di depot.

Tabel 5. 15 Perbandingan Hasil 4 Skenario Optimasi Rute Penyiraman Taman

Skenario	Existing	Skenario 1	Skenario 2	Skenario 3	Skenario 4
Fungsi Objektif (Menit)	5081	3558,6	3506,55	3514,55	3496,55
Jumlah Kendaraan (Unit)	9	8	8	8	8
Running Time (Menit)	-	3840	5100	4623	4781
Penghematan	-	24,4%	30,98%	30,83%	31,18%

Gambar 5. 7 Perbandingan Fungsi Objektif Optimasi Rute Penyiraman Taman



Penugasan kendaraan yang direkomendasikan dalam penelitian ini adalah menggunakan 8 unit kendaraan truk tangki untuk keempat skenario sesuai dengan hasil uji menggunakan model ACO. Penugasan yang ada tidak melanggar *contraints* yang ada baik dari sisi kapasitas maupun *time window* dengan semua *demand* pada setiap taman dapat dilayani semuanya.

Untuk perbandingan hasil skenario dengan rute *existing*, apabila dilihat dari sisi efisiensi waktu untuk fungsi objektif, skenario 4 lebih unggul karena total waktu tempuh menghasilkan nilai yang paling minimum dibandingkan dengan 3 skenario yang lain. Sehingga penghematan yang didapat adalah 31,18% dibandingkan dengan rute *existing*. Namun dalam pengaplikasiannya, skenario 3 lebih memungkinkan untuk diterapkan karena tanpa *split service* sama sekali dan *split service* cenderung lebih membingungkan karena *node* yang dikunjungi adalah sebuah taman, dimana penanda taman tersebut sudah dikunjungi adalah permukaan taman dan sisi jalan sekitar taman yang basah.

Tabel 5. 16 Perbandingan Rute Existing dan Skenario

Truk	Skenario	Parameter	Rute ke-																																						Total Waktu	Jumlah Pengiriman	*Service Time*	Jadual Keberangkatan	Jadual kepulangan	Waktu *Idle*
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42						
1	Exirting	rute	1	48	2	48	2	48	2	3	49	3	50	4	50	4	50	4	50	4	50	4	50	4	50	4	5	50	5	50	5	50	5	50	5	6	49	6	49	6	1						
		waktu		12	13	12	13	12	13	8	12	1	16	2	16	2	16	2	16	2	16	2	16	2	6	2	7	2	7	2	7	2	7	2	4	7	4	7	21						291	16	555	348	1194	-294
		jarak		6	3,9	6	3,9	6	3,9	3	3,6	0,1	4,9	0,75	4,9	0,75	4,9	0,75	4,9	0,75	4,9	0,75	4,9	0,75	1,6	0,75	2	0,75	2	0,75	2	0,75	2	1,4	1,3	2,2	1,3	2,2	6,7						98					
		demand		0	5000	0	5000	0	1250	3750	0	3750	0	5000	0	5000	0	5000	0	5000	0	5000	0	5000	0	2000	3000	0	5000	0	5000	0	5000	0	4500	500	0	5000	0	5000	0						78750					
	1	rute	1	52	4	50	4	50	4	50	4	50	4	50	4	30	27	71	27	71	27	29	74	29	6	70	28	6	70	5	1																	
		waktu		21	25	17	22	17	22	17	22	17	22	17	10	6,25	11,75	17	22	17	14,25	11,75	16	9,75	14,25	17	6,1	15,9	17	25	18															468	11	468	339	807	93
		demand		0	5000	0	5000	0	5000	0	5000	0	5000	0	2000	562,5	2438	0	5000	0	3063	1938	0	2188	2813	0	1275	3725	0	5000	0															55000					
	2	rute	1	53	45	53	45	53	45	7	8	55	8	10	56	11	10	54	10	9	21	59	20	19	22	34	83	33	83	32	83	32	1														
		waktu		24	22	17	22	17	10	11	7	16	16	7	17	17,5	4,5	16	3	18	16	20	10,75	10,5	9,25	2,5	20	21	16	21	16	21	25													454	10	454	336	790	110
		demand		0	5000	0	5000	0	2000	1500	1500	0	3750	1250	0	4125	875	0	500	3750	750	0	1688	1875	1313	125	0	5000	0	5000	0	5000	0													50000					
	3	rute	1	53	45	53	45	53	45	14	43	76	43	76	43	76	43	76	43	76	43	76	43	76	43	76	43	1																					
		waktu		24	22	17	22	17	10	12,75	14,25	16	21	16	21	16	21	16	21	16	21	16	21	16	21	16	17,75	31																	462,75	11	462,75	336	798,75	101,25
		demand		0	5000	0	5000	0	2000	1688	1313	0	5000	0	5000	0	5000	0	5000	0	5000	0	5000	0	5000	0	4188	0																	54187,5					
	4	rute	1	52	4	50	4	50	4	50	4	50	4	50	4	30	6	70	6	70	6	28	5	49	5	49	5	49	5	1																					
		waktu		26	21	16	2	7	14	18	5	23	17	22	17	12	19	16	21	16	21	16	21	16	21	16	17	7	19	24															450	11	450	334	784	116
		demand		0	5000	0	250	1500	3250	0	500	4500	0	5000	0	2500	2500	0	5000	0	5000	0	5000	0	5000	0	3750	1250	0	0															50000					
2	Exirting	rute	1	53	7	8	55	8	9	55	9	10	11	56	11	12	58	12	13	1																															
		waktu		9	5	1	1	1	5	1	3	3	1	1	1	4	1	1	4	22																										64	5	163	351	578	322
		jarak		5	1,6	0,35	0,45	0,45	1,4	0,45	0,8	1,1	0,3	0,3	0,3	1,4	0,65	0,65	1,1	8																										24,3					
		demand		0	1500	3500	0	1750	3250	0	500	2625	1875	0	2250	2750	0	250	1687,5	0																										21937,5					
	1	rute	1	53	45	53	45	85	41	86	41	86	42	86	41	86	41	86	41	47	84	44	47	84	42	82	42	1																					
		waktu		24	22	17	22	20	25	16	21	16	21	16	21	16	21	16	11	12	19	23,5	4,5	19	22	16	21	23																	465	11	465	336	801	99
		demand		0	5000	0	5000	0	5000	0	5000	0	5000	0	5000	0	5000	0	2500	2500	0	4875	125	0	5000	0	5000	0																	55000					
	2	rute	1	79	26	73	31	29	74	29	5	49	5	49	5	49	3	49	3	2	49	2	49	2	5	1																					
		waktu		17	32	19	11,5	17,5	16	5	20	19	24	19	24	19	21	16	11	18	27	32	27	27	17	18																			457	9	457	343	800	100
		demand		0	5000	0	1875	3125	0	1000	4000	0	5000	0	5000	0	5000	0	2500	2500	0	5000	0	3750	1250	0																			45000					
3	Exirting	rute	1	59	14	15	21	59	21	20	17	61	17	61	17	61	17	16	63	16	63	16	19	18	64	18	1																					
		waktu		15	9	8	4	5	5	5	4	1	1	1	1	1	1	3	2	2	2	2	12	6	2	2	27																		121	8	277	345	743	157
		jarak		8,2	2,8	2,4	1,4	1,4	1,4	1,9	1,6	0,15	0,15	0,15	0,15	0,15	0,15	1,4	0,75	0,75	0,75	0,75	5,2	2,4	0,75	0,75	9,8																		45,3					
		demand		0	1688	1875	1437	0	2163	1688	1149	0	5000	0	5000	0	851	4149	0	5000	0	601	1875	2524	0	4226	0																		39226					
	1	rute	1	79	39	37	79	37	79	37	79	37	38	81	36	81	36	81	36	81	38	40	26	71	5	49	5	1																					
		waktu		17	18,2	4,8	19	24	19	24	19	20,2	8,8	16	21	16	21	16	21	16	3,8	20,5	9,7	19	26	19	24	18																	441	10	441	343	784	116
		demand		0	4050	950	0	5000	0	5000	0	4050	950	0	5000	0	5000	0	5000	0	700	4125	175	0	5000	0	5000	0																	50000					
	2	rute	1	57	12	15	13	59	13	18	64	35	64	18	35	76	43	76	43	76	43	76	43	76	43	76	43	76	43	1																					
		waktu		28	14	11,5	5,5	16	7,25	22,75	17	23	18	15,25	7,75	18	21	16	21	16	21	16	21	16	21	16	21	16	21	31															478	11	478	332	810	90
		demand		0	3000	1875	125	0	1563	3438	0	5000	0	3313	1688	0	5000	0	5000	0	5000	0	5000	0	5000	0	5000	0	5000	0															55000					
	3	rute	1	52	4	50	4	50	4	50	4	50	4	30	27	71	27	71	27	26	71	26	31	74	31	29	74	29	1																					
		waktu		21	25	17	22	17	22	17	22	17	22	17	10	6,25	11,75	17	22	17	14,25	12,75	19	23,25	6,75	18	9,75	18,25	16	4,25	17															462,25	11	462,25	339	801,25	98,75
		demand		0	5000	0	5000	0	5000	0	5000	0	5000	0	2000	562,5	2438	0	5000	0	3063	1938	0	4813	187,5	0	1688	3313	0	812,5	0															50812,5					
	4	rute	1	49	3	49	3	2	49	2	49	2	5	49	5	49	5	28	6	70	6	70	6	70	6	40	36	1																					
		waktu		29	21	16	11	18	27	32	27	27	17	19	24	19	17,75	7,1	2,15	17	22	17	22	17	2,35	26,5	5,65	5																	449	9	449	331	780	120
		demand		0	5000	0	2500	2500	0	5000	0	3750	1250	0	5000	0	3438	1275	287,5	0	5000	0	5000	0	212,5	4125	662,5	0																	45000					
4	Exirting	rute	1	64	22	23	65	23	25	69	24	69	24	69	24																																		
		waktu		18	1	4	1	1	3	4	4	4	4	4	4																														52	4	154	342	548	352
		jarak		9,7	0,4	1,1	0,35	0,35	1,4	1,3	1,1	1,1	1,1	1,1	1,1																																	
		demand		0	1313	3688	0	3063	1875	0	5000	0	5000	0	3500																														23437,5					
	1	rute	1	49	3	49	3	2	49	2	49	2	5	49	5	73	31	26	71	5	6	70	6	46	42	82	42	1																					
		waktu		29	21	16	11	18	27	32	27	27	17	19	24	21	11,5	18,5	19	11	17	17	2,85	23	7,15	16	21	23																	476	9	476	331	807	93
		demand		0	5000	0	2500	2500	0	5000	0	3750	1250	0	5000	0	1875	3125	0	1250	3750	0	212,5	3750	1038	0	5000	0																	45000					
	2	rute	1	53	42	82	42	82	42	82	42	82	42	82	42	41	86	41	86	41	86	41	86	41	44	84	44	47	1																					
		waktu		24	20	16	21	16	21	16	21	16	21	16	9	16	16	21	16	21	16	21	16	19	5	19	21,5	6,5	19																	461	11	461	336	797	103
		demand		0	5000	0	5000	0	5000	0	5000	0	5000	0	2000	3000	0	5000	0	5000	0	5000	0	4500	500	0	4375	625	0																	55000					
	3	rute	1	49	3	49	3	2	49	2	49	2	5	49	5	49	5	49	5	49	5	49	5	28	1																							
		waktu		29	21	16	11	18	27	32	27	27	17	19	24	19	24	19	24	19	24	19	9	7,1	16																			448,1	9	448,1	331	779,1	120,9
		demand		0	5000	0	2500	2500	0	5000	0	3750	1250	0	5000	0	5000	0	5000	0	5000	0	1250	1275	0																			42525					
	4	rute	1	54	10	11	56	11	12	13	59	13	19	20	61	17	61	17	61	17	16	63	16	63	16	33	83	33	83	33	34	1															
		waktu		26	11,5	10,5	16	8	16	5	16	6,75	15,5	9,75	25	21	16	21	16	9	15	17	22	17	9	20	16	21	16	7	17	29													455	10	455	334	789	111
		demand		0	2625	2375	0	1750	3000	250	0	1438	1875	1688	0	5000	0	5000	0	2000	3000	0	5000	0	1750	3250	0	5000	0	1500	3500	0													50000					

Tabel 5.15 Perbandingan Rute Existing dan Skenario (Lanjutan)

Truk	Skenario	Parameter	Rute ke-																																							Total Waktu	Jumlah Pengiriman	*Service Time*	Jadwal Keberangkatan	Jadwal kepulangan	Waktu *Idle*
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42						
5	Exirting	rute	1	71	27	71	27	71	27	30	28	31	74	26	71	26	29	74	29	1																												
		waktu		15	2	2	2	2	2	2	3	8	3	7	4	4	10	1	1	17																									85	6	190	345	620	280
		jarak		4,5	0,7	0,7	0,7	0,7	0,7	0,7	1,1	2,8	1	2	1,2	1,2	2,1	0,45	0,45	5,7																									26,7					
		demand		0	5000	0	5000	0	500	562,5	1275	1875	0	5000	0	1750	3250	0	875	0																									25087,5					
	1	rute	1	53	45	7	8	55	8	10	56	11	10	54	10	9	47	84	47	42	82	42	82	42	43	76	43	76	43	76	35	64	35	1													
		waktu		24	10	11	7	16	16	7	17	17,5	4,5	16	3	18	13	19	23,5	1,5	16	21	16	4,35	20,65	16	21	16	21	16	23	18	23	24											480	11	480	336	816	84
		demand		0	2000	1500	1500	0	3750	1250	0	4125	875	0	500	3750	750	0	4875	125	0	5000	0	837,5	4163	0	5000	0	5000	0	5000	0													55000					
	2	rute	1	52	4	50	4	50	4	50	4	50	4	50	4	30	27	71	27	71	26	27	5	49	5	49	5	6	1																	
		waktu		21	25	17	22	17	22	17	22	17	22	17	10	6,25	11,75	17	22	17	11	17,25	4,75	19	24	19	12,25	13,75	21																	445	10	445	339	784	116
		demand		0	5000	0	5000	0	5000	0	5000	0	5000	0	2000	562,5	2438	0	5000	0	1750	3063	187,5	0	5000	0	2063	2938	0																	50000					
	3	rute	1	55	8	55	8	7	9	54	9	47	84	47	42	82	42	82	42	82	42	82	42	82	42	82	42	46	1																	
		waktu		26	21	16	2	7	14	18	5	28	19	19	6	16	21	16	21	16	21	16	21	16	21	16	21	16	4	18	17														421	10	421	334	755	145
		demand		0	5000	0	250	1500	3250	0	500	4500	0	3750	1250	0	5000	0	5000	0	5000	0	5000	0	5000	0	750	3750	0																	49500					
	4	rute	1	79	26	71	26	31	42	82	42	82	42	82	42	82	42	82	42	82	42	47	18	64	18	35	64	35	1																	
		waktu		17	32	19	11	13,5	20,5	16	21	16	21	16	21	16	21	16	21	16	3,5	9	16,5	17	19,5	3,5	18	23	24																	448	10	448	343	791	109
		demand		0	5000	0	1750	1875	1375	0	5000	0	5000	0	5000	0	5000	0	5000	0	625	2000	2375	0	4375	625	0	5000	0																	50000					
6	Exirting	rute	1	78	32	77	32	77	32	33	77	33	77	33	35	76	35	76	35	76	35	34	77	34	1																							
		waktu		26	8	6	6	6	6	2	5	5	5	5	3	3	3	3	3	3	3	7	9	9	29																						155	9	306	334	795	105
		jarak		7,9	2,4	1,7	1,7	1,7	1,7	0,5	1,4	1,4	1,4	1,4	1,5	0,8	0,8	0,8	0,8	0,8	0,8	2,4	2,6	2,6	10,4																						47,5					
		demand		0	5000	0	5000	0	3500	1500	0	5000	0	3250	1750	0	5000	0	5000	0	1750	3250	0	2750	0																						42750					
	1	rute	1	79	26	18	64	19	33	83	33	83	34	64	22	34	23	65	23	25	67	25	67	25	67	25	24	33	83	33	32	1															
		waktu		17	25,8	28,2	17	13,5	19,5	16	21	16	25	19	6,25	6	14,75	16	17,25	6,75	16	21	16	21	16	11,25	14,5	11,25	16	5,25	17,75	25													476	10	476	343	819	81
		demand		0	3450	1550	0	1875	3125	0	5000	0	5000	0	1313	1000	2688	0	4062,5	937,5	0	5000	0	5000	0	2563	1875	562,5	0	1063	3938	0													50000					
6	2	rute	1	61	17	61	17	61	17	16	63	16	63	16	25	67	25	67	25	67	25	23	65	23	24	64	24	35	1																	
		waktu		35	21	16	21	16	9	15	17	22	17	9	24	16	21	16	21	16	2	22	16	9	16	19	15,5	15,5	24																	451	10	451	325	776	124
		demand		0	5000	0	5000	0	2000	3000	0	5000	0	1750	3250	0	5000	0	5000	0	250	4750	0	2000	3000	0	2875	2125	0																	50000					
	3	rute	1	64	33	83	33	32	83	32	83	32	83	32	16	63	16	63	16	17	61	17	61	17	52	39	1																			
		waktu		33	23	16	20	3	16	21	16	21	16	14	18	17	22	17	14	11	16	21	16	21	39	23,2	4																		438,2	10	438,2	327	765,2	134,8
		demand		0	5000	0	4750	250	0	5000	0	5000	0	3250	1750	0	5000	0	3000	2000	0	5000	0	5000	0	4050	0																		49050					
	4	rute	1	53	14	43	76	43	76	43	76	43	76	43	76	43	76	43	76	43	76	43	35	64	35	64	35	32	1																	
		waktu		24	12,75	22,25	16	21	16	21	16	21	16	21	16	21	16	21	16	21	16	9,75	12,25	18	23	18	3,25	24,75	25																	469	10	469	336	805	95
		demand		0	5000	0	1750	1875	1375	0	5000	0	5000	0	5000	0	5000	0	5000	0	625	2000	2375	0	4375	625	0	5000	0																	50000					
7	Exirting	waktu		2	4	4	4	4	4	4	2	6	1	1	1	1	1	1	1	1	16	13																									71	9	294	358	723	177
		jarak		0,6	1,3	1,3	1,3	1,3	1,3	1,3	1,3	1,7	0,3	0,3	0,3	0,3	0,3	0,3	0,3	0,3	4,9	4,8																									23,5					
		demand		0	5000	0	5000	0	5000	0	4050	0	5000	0	5000	0	5000	0	1650	0	4125	0																									39825					
	1	rute	1	53	14	43	76	43	76	43	76	43	76	43	35	64	18	64	18	35	21	59	15	13	12	59	20	17	1																	
		waktu		24	13,75	22,25	16	21	16	21	16	21	16	21	16	13,1	8,9	18	22	17	2,8	7,1	18,1	20	10,5	11,75	9,75	17	10,75	17,25	31															459	10	459	336	795	105
		demand		0	1688	3313	0	5000	0	5000	0	5000	0	5000	0	3025	1975	0	5000	0	200	1525	3275	0	1875	1688	1438	0	1688	3313	0															50000					
7	2	rute	1	53	14	21	35	76	43	76	43	35	33	83	33	32	83	22	24	47	84	47	84	47	46	86	46	28	6	70	6	70	6	40	36	1										
		waktu		24	13,75	20,4	6,85	18	21	16	3	17,9	4,1	16	18,9	4,1	16	12,9	14,5	10,6	19	24	19	13,9	12,1	17	6,9	13,1	11	17	22	17	2,25	26,5	6,25	5									470	10	470	336	806	94
		demand		0	1688	2850	462,5	0	5000	0	500	4225	275	0	4475	525	0	2975	1875	150	0	5000	0	2475	2525	0	1225	1275	2500	0	5000	0	62,5	4125	813	0									50000					
	3	rute	1	58	18	64	18	35	64	35	64	35	64	35	21	15	59	15	40	36	81	36	81	36	81	36	38	81	38	1																	
		waktu		31	24	17	9	14	18	23	18	23	18	4	19,4	8,6	18	5,9	38,5	3,6	16	21	16	21	16	20,4	1,6	16	7	4															432	10	432	329	761	139
		demand		0	5000	0	1750	3250	0	5000	0	5000	0	250	3600	1150	0	725	4125	150	0	5000	0	5000	0	4850	150	0	1500	0															46500					
	4	rute	1	64	34	22	23	65	23	65	23	25	67	25	24	69	24	44	84	44	32	83	32	21	58	21	15	38	81	38	36	1															
		waktu		33	14	7,25	8,75	16	21	16	3,25	20,75	16	21	16	17,25	10,75	19	7,75	23,25	19	7,25	26,75	16	18,5	10,5	20	16,9	11,5	18,6	16	7	15	5													479	10	479	327	806	94
		demand		0	2500	1313	1188	0	5000	0	562,5	4438	0	5000	0	4063	937,5	0	937,5	4063	0	812,5	4188	0	4375	625	0	2975	1875	150	0	1500	3500	0													50000					
8	Exirting	rute	1	82	42	82	42	82	42	82	42	82	42	82	42	41	82	41	82	41	82	41	82	41	43	83	43	83	43	83	43	83	43	83	43	84	44	1								
		waktu		24	1	1	1	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4	4	4	8	3	3	3	3	3	3	3	3	3	3	3	9	4	25								
		jarak		7,3	0,36	0,36	0,36	0,36	0,36	0,36	0,36	0,36	0,36	0,36	0,36	1,1	1,2	1,2	1,2	1,2	1,2	1,2	1,2	1,2	2,7	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	1	0,9	3	1,3	9	165	19	665	336	1166	-266
		demand		0	5000	0	5000	0	5000	0	5000	0	5000	0	2000	3000	0	5000	0	5000	0	5000	0	4500	500	0	5000	0	5000	0	5000	0	5000	0	5000	0	5000	0	4875	0					94875					
	1	rute	1	61	17	61	17	16	63	16	63	16	32	83	32	83	32	21	12	1																												
		waktu		35	21	16	15,75	8,25	17	22	17	15,75	17,25	16	21	16	13	9,3	11,25	22																									293,55	6	293,55	325	618,55	281,45
		demand		0	5000	0	3688	1313	0	5000	0	3438	1563	0	5000	0	3000	325	1562,5	0																									29887,5					
	2	rute	1	79	37	79	37	79	39	37	79	37	38	81	38	36	81	36	81	36	1																												
		waktu		17	24	19	24	19	18,2	4,8	19	20,2	8,8	16	3,8	18,2	16	21	16	20,55	5																									290,55	7	290,55	343	633,55	266,45
		demand		0	5000	0	5000	0	4050	950	0	4050	950	0	700	4300	0	5000	0	4888	0																									34887,5					
	3	rute	1	53	41	86	41	86	41	86	41	86	41	6	70	6	70	6	37	79	37	79	37	79	37	1																						
		waktu		24	31	16	21	16	21	16	21	16	11	20	17	22	17	14	21	19	24	19	24	19	16	3																			428	10	428	336	764	136
		demand		0	5000	0	5000	0	5000	0	5000	0	2500	2500	0	5000	0	3000	2000	0	5000	0	5000	0	3000	0																			48000					
	4	rute	1	81	36	81	36	81	36	37	79	37	79	37	79	37	39	1																														
		waktu		20	21	16	21	16	4,35	21,65	19	24	19	24	19	7,35	17,2	4																											253,55	6	253,55	340	593,55	306,45
		demand		0	5000	0	5000	0	837,5	4163	0	5000	0	5000	0	837,5	4050	0																													29887,5					
9	Exirting	rute	1	87	45	87	45	87	45	47	86	47	86	47	46	1																																
		waktu		19	4	4	4	4	4	9	5	5	5	5	2	17																													87	5	171	341	599	301
		jarak		5,7	1,1	1,1	1,1	1,1	1,1	2,8	1,6	1,6	1,6	1,6	0,7	6,1																													27,2					
		demand		0	5000	0	5000	0	2000	3000	0	5000	0	250	3750	0																													24000					
	1	rute																																													
		waktu																																										0	0	0	-	-	-
		demand																																										0					
	2	rute																																													
		waktu																																										0	0	0	-	-	-
		demand																																										0					
	3	rute																																													
		waktu																																										0	0	0	-	-	-
		demand																																										0					
	4	rute																																													
		waktu																																										0	0		-	-	-
		demand																																										0					

(Halaman ini sengaja dikosongkan)

BAB 6
KESIMPULAN

Pada bab ini akan dijelaskan mengenai kesimpulan yang didapat pada penelitian berdasarkan tujuan yang ingin dicapai, hasil pengolahan data, dan analisa data, serta saran yang diajukan untuk penelitian selanjutnya.

6.1. Kesimpulan

Kesimpulan yang dapat ditarik dari penelitian ini adalah sebagai berikut:

1. Telah dilakukan pengembangan model *Ant Colony Optimization* (ACO) dengan 4 skenario untuk pengambilan keputusan dan memberikan solusi alternatif bagi DKRTH Kota Surabaya dalam menentukan rute dan jumlah kendaraan yang dibutuhkan dalam kasus penyiraman taman Kota Surabaya.
2. Berdasarkan hasil uji model ACO pada penelitian ini didapatkan rute yang direkomendasikan pada 4 skenario. Total jarak yang ditempuh pada rute rekomendasi oleh skenario 1 adalah 3558,6 menit, rute rekomendasi skenario 2 adalah 3519,6 menit, rute rekomendasi skenario 3 adalah 3514,55 menit dan rute rekomendasi skenario 4 adalah 3496,55 menit.
3. Dari 9 truk yang disediakan oleh DKRTH Surabaya Rayon Timut, pada skenario 1 dan 2 hanya 8 truk yang dibutuhkan pada rute baru yang direkomendasikan. 1 truk yang tidak melakukan perjalanan berfungsi sebagai cadangan ketika ada kenaikan *demand* pada taman ataupun pengganti pada saat ada kendaraan yang *down*.
4. Dari 4 skenario yang dibuat, skenario 4 lebih unggul apabila dilihat dari sisi efisiensi waktu, karena total waktu tempuh menghasilkan nilai yang paling minimum dibandingkan dengan 3 skenario yang lain dan menghasilkan penghematan sebesar 31,18% dari rute *existing*. Namun dalam pengaplikasiannya, skenario 3 lebih memungkinkan untuk diterapkan karena *split service* cenderung lebih membingungkan karena *node* yang dikunjungi adalah sebuah taman, dimana penanda taman tersebut sudah dikunjungi adalah permukaan taman dan sisi jalan sekitar taman yang basah.

6.2. Saran

Adapun saran pada penelitian ini adalah sebagai berikut:

1. Model dapat dikembangkan dengan tambahan *constraints* efisiensi bahan bakar. Hal ini berkaitan dengan kondisi real di lapangan mengenai batasan bahan bakar kendaraan dalam satu hari.

2. Mengintegrasikan jarak dan waktu tempuh secara *real time*. Hal ini memungkinkan untuk pencarian rute baru jika ada penambahan jumlah taman secara mendadak atau meng-*cover* tugas kendaraan lain apabila ada kendaraan yang rusak.
3. Membuat *user interface* algoritma ACO sehingga dapat dengan mudah digunakan oleh orang awam. Sehingga memudahkan untuk proses perutean.

DAFTAR PUSTAKA

A. Rais, F. A. (2013). New mixed integer-programming model for the pickup-and-delivery. *European Journal of Operational Research*.

A., R. P. (2007). Aplikasi Kombinatorial pada Vehicle Routing Problem. 1-7.

Balas, E. &. (2001). Linear time dynamic-programming algorithms for new classes of restricted TSPs: A computational study. *INFORMS journal on Computing*, 56-75.

Bang-Jesen, J. d. (2007). *Digraph Theory, Algorithms and Application.* Berlin Heidelberg New York: Springer.

Barnhart C. & Laporte G. (2007). Transportation. In L. G. Cordeau JF, *Handbooks in Operations Research and Management Science, Vol. 14* (pp. 367–428). Amsterdam: Elsevier.

Bazaraa, M. J. (1990). *Linear Programming and Network Flow (Second Edition).* New York: Wiley.

Bianchessi, N. and Irnich, S. (2016, 02). Branch-and-Cut for the Split Delivery Vehicle Routing Problem. *Transportation Science*, pp. 1-50.

Blum, C. &. (2003). Metaheuristic in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys 35*, 268-308.

Budi Santosa, T. J. (2018). *Pengantar Metaheuristik Implementasi Dengan Matlab.* Surabaya: ITS Tekno Sains.

Clarke, G. & Wright, J. W. . (1964). Scheduling of Vehicles from a Central Depot to a Number of Delivery Point. *Institute for Operations Research and the Management Sciences (INFORMS)*, 568-581.

Corberán Á. & Laporte, G. (2015). *Arc Routing: Problems. Methods, and Applications, Society for Industrial and Applied Mathematics.* Philadelphia: PA.

Diestel, R. (2000). *Graph Teory.* New York: Springer-Verlag.

Dorigo, L. M. (2000). An ant colony system hybridized with a new local search for the sequential ordering problem. *INFORMS Journal on Computing*, 237-255.

Dorigo, M. a. (1999). Ant Colony Optimization: A New Metaheuristic. *Proceedings of the Congress on Evolutionary Computation*, 1470-1477.

Doyuran, T. &. (2011). A robust enhancement to the Clarke–Wright savings algorithm. *Journal of the Operational Research Society*, 223-231.

F. S. Hillier and G. J. Lieberman . (2010). *Introduction to Operations Research (9th utg.).* McGraw-Hill International Edition.

Falkenauer, E. (1996). A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, 2:5-30.

Febrianti, N. & Sofan, P. (2014). RUANG TERBUKA HIJAU DI DKI JAKARTA BERDASARKAN ANALISIS SPASIAL DAN SPEKTRAL DATA LANDSAT 8. *Seminar Nasional Penginderaan Jauh*, 498-504.

Fisher., M. (1995). Vehicle routing (Chapter 1.8). In *Handbooks of Operations Research and Management Science* (pp. 1-31).

Gregory Gutin, M. J. (2018). Parameterized Complexity of the k-Arc Chinese. *Department of Computer Science Royal Holloway, University of London*.

Guy Desaulniers, J. D. (2002). VRP with Pickup and Delivery. 227.

Holmberg, K. (2015). Heuristics for the Weighted k-Chinese/rural. *Department of Mathematics Linköping Institute of Technology Sweden*.

Imansari, N. & Khadiyanta P. (2015). Penyediaan Hutan Kota dan Taman Kota sebagai Ruang Terbuka Hijau (RTH) Publik Menurut Preferensi Masyarakat di Kawasan Pusat Kota Tangerang. *RUANG (VOL.1) NO 3*, 101-110.

J. K. Lenstra and A. H. G. Rinnooy Kan. (1981). Complexity of vehicle routing and scheduling problems. In *Networks* (pp. 221-227).

Jiri Cejka, R. K. (2017). Winter Maintenance Optimization by Graph Theory. *Periodica Polytechnica Transportation Engineering*.

Ke Tang, J. W. (2017). A Scalable Approach to Capacitated Arc Routing Problems Based on Hierarchical Decomposition. *IEEE TRANSACTIONS ON CYBERNETICS, VOL. 47, NO. 11*, 3928-3940.

Marco Colombi, Á. C. (2015). The Hierarchical Mixed Rural Postman Problem. *Institute for Operations Research and the Management Sciences (INFORMS) Transportation Science*, 755-770.

Mason, B. V. (2005). *Formulation and Analysis of Linear Programs.* Stanford.

Maya Rahmawati, A. R. (2016). *PENENTUAN RUTE TRUK TANGKI PENYIRAMAN TAMAN DENGAN KONSEP VEHICLE ROUTING PROBLEM DAN RURAL POSTMAN PROBLEM.* Malang.

Monirehalsadat Mahmoudi, X. Z. (2012). Finding optimal solutions for vehicle routing problem with pickup and delivery services with time windows: A dynamic programming approach based on state–space–time network representations. *School of Sustainable Engineering and the Built Environment, Arizona State University*.

Munir, R. (2006). *Matematika Diskrit.* Bandung: Teknik Informatika ITB.

Nagy, N. A. (2014). Vehicle Routing Problem with Deliveries and Pickups: Modelling Issues and Meta-heuristics Solution Approaches. *International Journal of Transportation*, 95-110.

Najoan, A. S. (2001). Prototipe Aplikasi Sistem Informasi Akademik Pada Perangkat Android. *E-journal Teknik Elektro dan Komputer*, 3.

Purnomo, H. D. (2014). *Cara Mudah Belajar Metode Metaheuristik Menggunakan Matlab.* Yogyakarta: Gava Media.

Rajappa, G. P., Wilck, J. H., & Bell, J. E. . (2016). An ant colony optimization and hybrid metaheuristics algorithm to solve the split delivery vehicle routing problem. *International Journal of Applied Industrial Engineering (IJAIE)*, 55-73.

Said, I. M. (2005). APLIKASI UNTUK PERANGKAT BERGERAK MENGGUNAKAN JAVA 2 MICRO EDITION (J2ME). *Seminar Nasional Aplikasi Teknologi Informasi* , H-117-H121.

Santosa, B. & Ai, T. J. (2017). *Pengantar Metaheuristik Implementasi dengan Matlab.* Surabaya: ITS Tekno Sains.

Talbi, E. G. (2009). *Metaheuristic: from design to implementation.* Hoboken: New Jersey: John Wiley & Son, Inc.

Toth, P. & Vigo, D. (2014). VRP with Split Deliveries. In P. a. Toth, *Vehicle Routing Problems, Aplication and Method* (pp. 255-259). USA: Mathematical Optimization Society Philadelphia (MOS) - Society for Industrial Applied Mathematics and Mathematical (SIAM).

Toth, P., & Vigo, D. (2002). Models, relaxations and exact approaches for the capacitated vehicle routing problem. In *Discrete Applied Mathematics* (pp. 487-512).

(2007). *UNDANG-UNDANG REPUBLIK INDONESIA NOMOR 26.*

Viga Apriliana S., E. R. (2017). PENYELESAIAN MASALAH RUTE PENYIRAMAN TANAMAN MENGGUNAKAN ALGORITMA ARTIFICIAL IMMUNE SYSTEM (AIS) DI KOTA YOGYAKARTA. *SEMINAR MATEMATIKA DAN PENDIDIKAN MATEMATIKA UNY* , PT281-PT288.

Wassan, N. and Nagy, G. (2014). Vehicle Routing Problem with Deliveries and Pickups: Modelling Issues and Meta-heuristics Solution Approaches. *International Journal of Transportation*, 95-110.

Yoshiaki Shimizu, T. S.-K. (2016). A hybrid method for solving multi-depot VRP. *Journal of Advanced Mechanical Design, Systems, and Manufacturing, Vol 10, No 1*, 1-13.

(Halaman ini sengaja dikosongkan)

Lampiran 1

Berikut merupakan data *demand* tiap taman:

NO Kendaraan	Lokasi	Panjang Taman (Meter)	Lebar Taman (Meter)	Jumlah Ruas	Luas Taman (Meter^2)	*Demand* (Liter)
1	Jl.Kendal Sari Wonorejo	750	1	2	750	11250
	Jl. Raya Pandugo	500	1	2	500	7500
	Jl.Kedung Baruk Timur	900	2	2	1800	27000
	Jl. Mer. Kedung Asem	750	2	2	1500	22500
	Jl. Mer Gunung Anyar	350	2	2	700	10500
2	Jl. Bratang Rmi	400	0,5	1	200	1500
	Jl. Ngagel Jaya	700	1	1	700	5250
	Jl. Ngagel Jaya Selatan	500	1	1	500	3750
	Jl. Pucang	700	0,5	1	350	2625
	Jl. Kertajaya	1100	0,5	1	550	4125
	Jl. Dharmawangsa	800	0,5	1	400	3000
	Depan Karang Menjangan	450	0,5	1	225	1687,5
3	Jl. Raya Menur	450	0,5	1	225	1687,5
	Jl. Raya Karang Menjangan	500	0,5	1	250	1875
	Jl. Putro Agung	650	1	2	650	9750
	Jl. Raya Bronggalan	1600	0,5	2	800	12000
	Jl. Dharmahusada Indah 1	900	1	1	900	6750
	Jl. Kaliwaron	500	0,5	1	250	1875
	Jl. Tambang Boyo	450	0,5	1	225	1687,5
	Rotonde Muestopo	480	1	1	480	3600
4	Jl. Mulyorejo Utara	350	0,5	1	175	1312,5
	Jl. Sutorejo	900	1	1	900	6750
	Jl. Kalisari	250	1	1	250	1875
	Jl. Mulyosari	1800	1	1	1800	13500
5	Jl. Kali Rungkut	450	2	1	900	6750
	Jl. Rungkut Alang2	700	2	1	1400	10500
	Jl. Rungkut Asri	170	1	1	170	1275
	Jl. Rungkut Yakaya	550	1	1	550	4125
	Jl. Kedung Asem	150	0,5	1	75	562,5
	Jl. Rungkut Pasar Paing	500	0,5	1	250	1875

NO Kendaraan	Lokasi	Panjang Taman (Meter)	Lebar Taman (Meter)	Jumlah Ruas	Luas Taman (Meter^2)	*Demand* (Liter)
6	Jl.Dharmahusada Permai	900	2	1	1800	13500
	Jl.Dharmahusada Utara 8	650	2	1	1300	9750
	Jl.Merr Kalijudan Tepi	800	1	1	800	6000
	Jl.Dharmahusada Indah 2	900	2	1	1800	13500
7	Jl.Jemur Handayani	1000	1	2	1000	15000
	Jl.Jemursari	2000	1	1	2000	15000
	Puskesmas Jemursari (Aset)	220	1	1	220	1650
	Rotonde Raya Prapen	540	1	1	540	4050
	Jl.Kutisari	550	1	1	550	4125
8	Jl.Merrr Kopertis	1000	1	3	1000	22500
	Jl.Ir Soekarno	1200	1	3	1200	27000
	Jl.Merr Kalijudan	1800	1	3	1800	40500
	Jl. Arif Rahman Hakim Uht	1300	0,5	1	650	4875
9	jl. raya manyar	800	1	2	800	12000
	jl. semolo waru utara	500	1	1	500	3750
	jl. arif rahman hakim/itats	550	1	2	550	8250

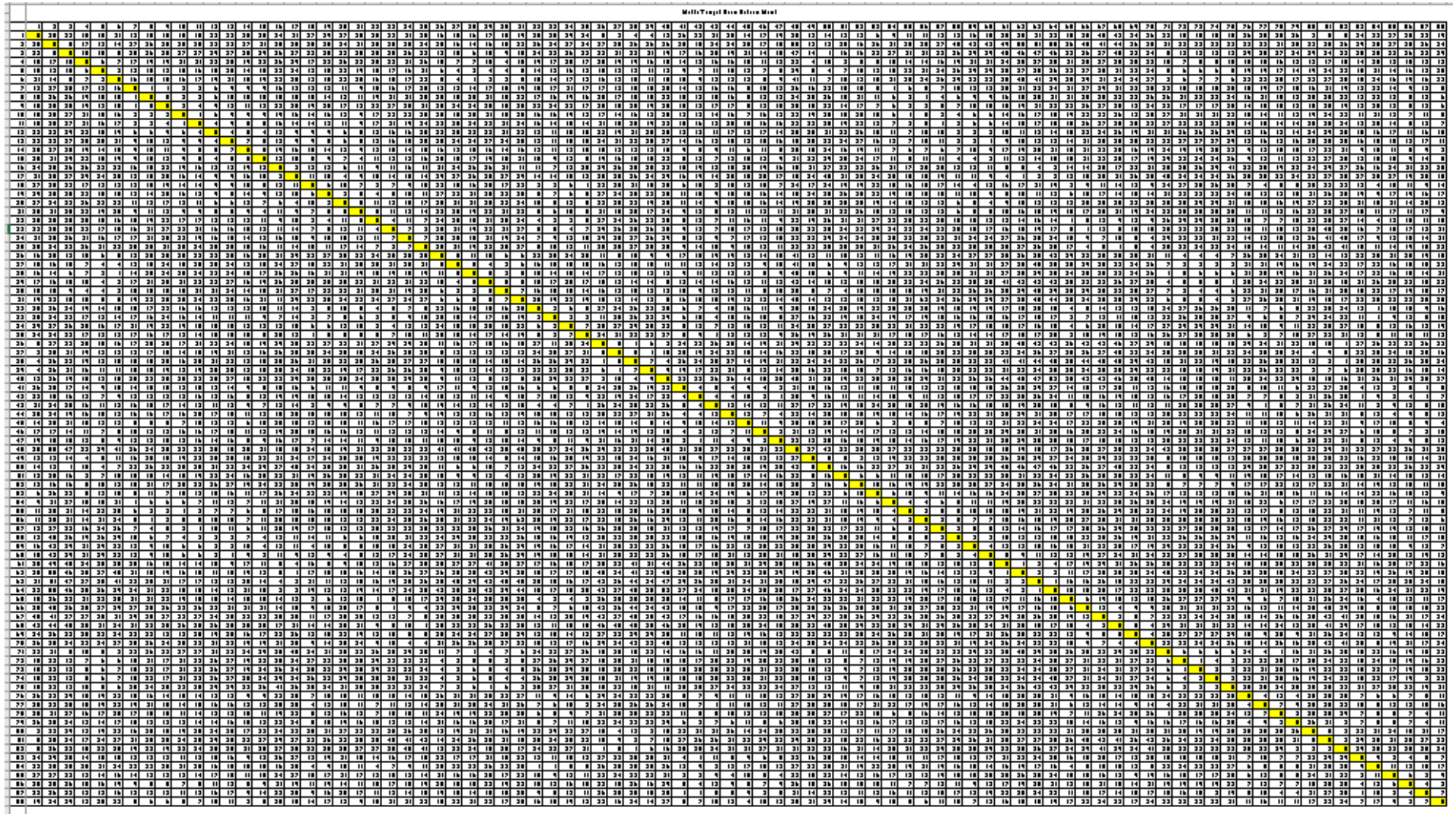
Lampiran 2

Berikut merupakan data lokasi pengambilan air:

No	Lokasi	No	Lokasi
48	Sungai depan Stikom	69	Selokan Depan SPBU Mulyosari
49	Selokan Depan Perumahan YKP Pandugo I	70	Sungai Dekat Polsek Rungkut Asri
50	Selokan Depan Samator	71	Selokan Seberang Masjid Al Hidayah Kalirungkut
51	Selokan Dekat Pizza Hut Merr	72	Selokan Sebrang Fresh Laundry Kalirungkut
52	sungai dekat hotel narita	73	Selokan Samping SDN Kalirungkut I
53	sungai dekat bakmie PGM	74	Sungai Samping SMAN 17
54	sungai dekat apotik Kimia Farma Pucang	75	Sumur Tandon Depan MNC
55	Sungai Sebelah Puskesmas Pucang	76	Selokan dekat Resto Niki Sae
56	Sungai Sebelah Masakan Padang Pucang	77	Selokan Depan RM Padang Sederhana
57	Sungai Sebelah Toko Kertas Sari Agung	78	Selokan Depan Sampoerna
58	Sungai Sebelah Vokasi Unair	79	Sumur Bor Depan Pizza Hut Jemursari
59	Tandon Sumur Depan Lapangan Hoky	80	Sungai Depan Jemur Andayani 1
60	Sungai Seberang Soto Madura Cabang Kalasan	81	Selokan Pinggir SDN Jemur Wonosari 3
61	Sungai Seberang Alfamidi Gresikan	82	Sungai Depan PT. Garam
62	Sungai Seberang Ampera Restu Mulya	83	Sungai Depan RSGM Unair
63	Sungai Seberang Indomaret Putro Agung	84	Selokan Depan Regency 21
64	Selokan Dharmahusada Indah Utara XIV	85	Sungai Seberang Apartemen Gunawangsa Menur
65	Selokan Depan Pool Blue Bird Mulyorejo	86	Sungai Depan ITATS
66	Selokan Depan Vilia Mulyosari	87	Sungai Depan SPBU Manyar (Dekat RS Bedah)

Lampiran 3

Berikut merupakan data waktu tempuh antar taman:



Lampiran 4

Berikut ini merupakan data penugasan *existing:*

Kendaraan	Jarak dan Waktu	Rute																																								Jumlah Jarak	Jumlah Pengi	*Service Time*	Jadwal Keber	Jadwal kepul	Waktu *Idle*
L 9023 QP	ruto	1	48	2	48	2	48	2	3	49	3	50	4	50	4	50	4	50	4	50	4	50	4	5	50	5	50	5	50	5	50	5	6	49	6	49	6	1							17	570	348	1209	-309
	waktu		12	13	12	13	12	13	8	12	1	16	2	16	2	16	2	16	2	16	2	16	2	6	2	7	2	7	2	7	2	7	2	4	7	4	7	21						291					
	jarak		6	3,9	6	3,9	6	3,9	3	3,6	0,1	4,9	0,75	4,9	0,75	4,9	0,75	4,9	0,75	4,9	0,75	4,9	0,75	1,6	0,75	2	0,75	2	0,75	2	0,75	2	1,4	1,3	2,2	1,3	2,2	6,7						98					
	demand		0	5000	0	5000	0	1250	3750	0	3750	0	5000	0	5000	0	5000	0	5000	0	5000	0	2000	3000	0	5000	0	5000	0	5000	0	4500	500	0	5000	0	5000	0						78750					
L 9460 NP	ruto	1	53	7	8	55	8	9	55	9	10	11	56	11	12	58	12	13	1																										6	178	351	593	307
	jarak		9	5	1	1	1	5	1	3	3	1	1	1	4	1	1	4	22																									64					
	waktu		5	1,6	0,35	0,45	0,45	1,4	0,45	0,8	1,1	0,3	0,3	0,3	1,4	0,65	0,65	1,1	8																									24,3					
	demand		0	1500	3500	0	1750	3250	0	500	2625	1875	0	2250	2750	0	250	1688	0																									21937,5					
L 9176 NP	ruto	1	59	14	15	21	59	21	20	17	61	17	61	17	61	17	16	63	16	63	16	19	18	64	18	1																			9	292	345	758	142
	jarak		15	9	8	4	5	5	5	4	1	1	1	1	1	1	3	2	2	2	2	12	6	2	2	27																		121					
	waktu		8,2	2,8	2,4	1,4	1,4	1,4	1,9	1,6	0,15	0,15	0,15	0,15	0,15	0,15	1,4	0,75	0,75	0,75	0,75	5,2	2,4	0,75	0,75	9,8																		45,3					
	demand		0	1688	1875	1437	0	2163	1688	1149	0	5000	0	5000	0	851	4149	0	5000	0	601	1875	2524	0	4226	0																		39226					
L 9321 NP	ruto	1	64	22	23	65	23	25	69	24	69	24	69	24																															5	169	342	563	337
	jarak		18	1	4	1	1	3	4	4	4	4	4	4																														52					
	waktu		9,7	0,4	1,1	0,35	0,35	1,4	1,3	1,1	1,1	1,1	1,1	1,1																														20,1					
	demand		0	1312,5	3687,5	0	3062,5	1875	0	5000	0	5000	0	3500																														23437,5					
L 9061 NF	ruto	1	71	27	71	27	71	27	30	28	31	74	26	71	26	29	74	29	1																										7	205	345	635	265
	jarak		15	2	2	2	2	2	2	3	8	3	7	4	4	10	1	1	17																									85					
	waktu		4,5	0,7	0,7	0,7	0,7	0,7	0,7	1,1	2,8	1	2	1,2	1,2	2,1	0,45	0,45	5,7																									26,7					
	demand		0	5000	0	5000	0	500	562,5	1275	1875	0	5000	0	1750	3250	0	875	0																									25087,5					
L 9313 NP	ruto	1	78	32	77	32	77	32	33	77	33	77	33	35	76	35	76	35	76	35	34	77	34	1																					10	321	334	810	90
	jarak		26	8	6	6	6	6	2	5	5	5	5	3	3	3	3	3	3	3	7	9	9	29																				155					
	waktu		7,9	2,4	1,7	1,7	1,7	1,7	0,5	1,4	1,4	1,4	1,4	1,5	0,8	0,8	0,8	0,8	0,8	0,8	2,4	2,6	2,6	10,4																				47,5					
	demand		0	5000	0	5000	0	3500	1500	0	5000	0	3250	1750	0	5000	0	5000	0	1750	3250	0	2750	0																				42750					
L 9018 RP	ruto	1	79	37	79	37	79	37	79	39	81	36	81	36	81	36	81	38	81	40	1																								10	309	358	738	162
	jarak		2	4	4	4	4	4	4	2	6	1	1	1	1	1	1	1	1	16	13																							71					
	waktu		0,6	1,3	1,3	1,3	1,3	1,3	1,3	1,3	1,7	0,3	0,3	0,3	0,3	0,3	0,3	0,3	0,3	4,9	4,8																							23,5					
	demand		0	5000	0	5000	0	5000	0	4050	0	5000	0	5000	0	5000	0	1650	0	4125	0																							39825					
L 9564 NP	ruto	1	82	42	82	42	82	42	82	42	82	42	82	42	41	82	41	82	41	82	41	82	41	43	83	43	83	43	83	43	83	43	83	43	83	43	83	43	83	43	84	44	1		20	680	336	1181	-281
	jarak		24	1	1	1	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4	4	4	8	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	9	4	25	165					
	waktu		7,3	0,36	0,36	0,36	0,36	0,36	0,36	0,36	0,36	0,36	0,36	0,36	1,1	1,2	1,2	1,2	1,2	1,2	1,2	1,2	1,2	2,7	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	2,8	1,3	8,8	51,96					
	demand		0	5000	0	5000	0	5000	0	5000	0	5000	0	2000	3000	0	5000	0	5000	0	5000	0	4500	500	0	5000	0	5000	0	5000	0	5000	0	5000	0	5000	0	5000	0	5000	0	4875	0	94875					
L 9490 NP	ruto	1	87	45	87	45	87	45	47	86	47	86	47	46	1																														6	186	341	614	286
	jarak		19	4	4	4	4	4	9	5	5	5	5	2	17																													87					
	waktu		5,7	1,1	1,1	1,1	1,1	1,1	2,8	1,6	1,6	1,6	1,6	0,7	6,1																													27,2					
	demand		0	5000	0	5000	0	2000	3000	0	5000	0	250	3750	0																													24000					

Lampiran 5

Berikut koding lengkap model VRPSDTW dengan Algoritma Ant Colony Optimization

```
# Standard lib
import pandas as pd
import numpy as np
import math
import random
from tqdm import tqdm
import re

# Import data set
df_park_info = pd.read_excel('data_input naik 20%.xlsx', sheet_name='park')
df_river_info = pd.read_excel('data_input naik 20%.xlsx', sheet_name='river')
df_vehicle_info      =      pd.read_excel('data_input      naik      20%.xlsx',
sheet_name='vehicle')
df_park_park       =       pd.read_excel('data_input       naik       20%.xlsx',
sheet_name='park_to_park').set_index('park_park')
df_park_river      =      pd.read_excel('data_input      naik      20%.xlsx',
sheet_name='park_to_river').set_index('park_river')
df_park_depot      =      pd.read_excel('data_input      naik      20%.xlsx',
sheet_name='park_to_depot').set_index('park_depot')
df_depot_river      =      pd.read_excel('data_input      naik      20%.xlsx',
sheet_name='depot_to_river').set_index('depot_river')

# View all columns
pd.set_option('display.max_columns', None)

def calculate_prob(df, alpha, beta):
    df['feromon_weight'] = df['feromon']**alpha
    df['trail_weight'] = df['trail']**beta
    df['numerator'] = df['feromon_weight']*df['trail_weight']
    df['denominator'] = df['numerator'].sum()
    df['probability'] = df['numerator']/df['denominator']
    df['cum_probability'] = df['probability'].cumsum()
    return df

def calculate_prob_saving(df, alpha, beta, gamma):
    df['feromon_weight'] = df['feromon']**alpha
    df['trail_weight'] = df['trail']**beta
    df['saving_weight'] = df['saving']**gamma
    df['numerator']                                                                =
df['feromon_weight']*df['trail_weight']*df['saving_weight']
    df['denominator'] = df['numerator'].sum()
    df['probability'] = df['numerator']/df['denominator']
    df['cum_probability'] = df['probability'].cumsum()
    return df

def find_potential_node(df, df_river, tau, rute, candidate, completed, pickup,
is_saving, split):
    # Last node
    last_node = rute[-1]

    # Delete completed node
    df = df[~df.index.isin(completed)]

    ### Get prob
    # Pick time travel data
    if pickup:
        df_prob      =      df.copy()[[last_node]].rename(columns={last_node:
'travel_time'})
```

```
        try:
            # Get to river
            river = pd.melt(df_river.reset_index(), id_vars=['depot_river'],
var_name='river', value_name='time_to_river')

            # Get river to park
            river_park = pd.melt(df_park_river.reset_index(),
id_vars=['park_river'], var_name='river',
value_name='time_to_park').set_index('park_river')

            # Merge to river to park
            river_park = river_park.reset_index().merge(river[['river',
'time_to_river']], how='left', on='river').rename(columns={'park_river':
'park_depot'}).set_index('park_depot')

            # Merge to df prob
            df_prob = df_prob.merge(river_park, how='left', left_index=True,
right_index=True)

            # Travel time
            df_prob['travel_time'] = df_prob['time_to_river'] +
df_prob['time_to_park']

            # Top closest
            df_prob = df_prob.reset_index().sort_values(by=['park_depot',
'travel_time'])
            df_prob =
df_prob.groupby('park_depot').head(1).set_index('park_depot')

        except:
            # Get to river
            river = pd.melt(df_river.reset_index(), id_vars=['park_river'],
var_name='river', value_name='time_to_river')
            river = river.query('park_river == @last_node')

            # Get river to park
            river_park = pd.melt(df_park_river.reset_index(),
id_vars=['park_river'], var_name='river',
value_name='time_to_park').set_index('park_river')

            # Merge to river to park
            river_park = river_park.reset_index().merge(river[['river',
'time_to_river']], how='left', on='river').rename(columns={'park_river':
'park_park'}).set_index('park_park')

            # Merge to df prob
            df_prob = df_prob.merge(river_park, how='left', left_index=True,
right_index=True)

            # Travel time
            df_prob['travel_time'] = df_prob['time_to_river'] +
df_prob['time_to_park']

            # Top closest
            df_prob = df_prob.reset_index().sort_values(by=['park_park',
'travel_time'])
            df_prob =
df_prob.groupby('park_park').head(1).set_index('park_park')

    else:
        df_prob = df.copy()[[last_node]].rename(columns={last_node:
'travel_time'})
```

```
        df_prob['river'] = ''

    # Filter top candidate
    df_prob_all = df_prob.copy()
    df_prob = df_prob.sort_values(by='travel_time').head(candidate)

    # Trail tau and trail
    df_prob  =  df_prob.merge(tau[last_node],  how='left',  left_index=True,
right_index=True).rename(columns={last_node: 'feromon'})
    df_prob['trail'] = 1/df_prob['travel_time']

    if is_saving:
        # saving
        try:
            df_prob['doi'] = df.loc[last_node, 'depot_1']
        except:
            df_prob['doi'] = 0.01
        df_prob  =  df_prob.merge(df_park_depot,  how='left',  left_index=True,
right_index=True).rename(columns={'depot_1': 'doj'})
        df_prob['saving']    =    df_prob['doi']    +    df_prob['doj']    -
df_prob['travel_time']

        # Replace neg saving
        df_prob['saving']    =    np.where(df_prob['saving']<0.01,    0.01,
df_prob['saving'])

        # Calculate prob
        df_prob = calculate_prob_saving(df_prob, alpha, beta, gamma)
    else:
        # Calculate prob
        df_prob = calculate_prob(df_prob, alpha, beta)

    ### Select
    # Random number
    rand = random.uniform(0,1)

    # Roulette wheel
    df_prob['select'] = 0
    for i in range(df_prob.shape[0]):
        if rand < df_prob.iloc[i]['cum_probability']:
            df_prob.loc[df_prob.index[i], 'select'] = 1
            break

    # Last node demand
    try:
        last_node_demand                                                    =
df_demand.query('location==@last_node').reset_index(drop=True).iloc[0]['deman
d_left']
    except:
        last_node_demand = 0

    # Check if single split happen
    if split != 'multiple' and last_node_demand > 0:
        # Potential next node selected
        potential_next = last_node

        # River to pickup
        potential_river       =       df_prob_all.query('park_park       ==
@last_node').reset_index(drop=True).iloc[0]['river']

        # Time
```

```
        time_to_node         =         df_prob_all.query('park_park         ==
@last_node').reset_index(drop=True).iloc[0]['travel_time']

        # Time to park and to river
        try:
            time_to_park       =       df_prob_all.query('park_park       ==
@last_node').reset_index(drop=True).iloc[0]['time_to_park']
        except:
            time_to_park       =       df_prob_all.query('park_park       ==
@last_node').reset_index(drop=True).iloc[0]['travel_time']
        try:
            time_to_river       =       df_prob_all.query('park_park       ==
@last_node').reset_index(drop=True).iloc[0]['time_to_river']
        except:
            time_to_river = 0

    else:

        # Potential next node selected
        potential_next = df_prob.query('select==1').index[0]

        # River to pickup
        potential_river                                                      =
df_prob.query('select==1').reset_index(drop=True).iloc[0]['river']

        # Time
        time_to_node                                                         =
df_prob.query('select==1').reset_index(drop=True).iloc[0]['travel_time']

        # Time to park and to river
        try:
            time_to_park                                                     =
df_prob.query('select==1').reset_index(drop=True).iloc[0]['time_to_park']
        except:
            time_to_park                                                     =
df_prob.query('select==1').reset_index(drop=True).iloc[0]['travel_time']
        try:
            time_to_river                                                    =
df_prob.query('select==1').reset_index(drop=True).iloc[0]['time_to_river']
        except:
            time_to_river = 0

    # Time from self to closest river
    time_to_self            =            pd.melt(df_park_river.reset_index(),
id_vars=['park_river'], var_name='river', value_name='time_to_park')
    time_to_self                                                             =
time_to_self.query('park_river==@potential_next')['time_to_park'].min()*2

    return  potential_next,  potential_river,  time_to_node,  time_to_park,
time_to_river, time_to_self

def find_next_node(df, df_river, tau, rute, candidate, capacity, capacity_max,
current_time, max_time, completed, is_saving, split):

    # Initiate pickup
    if capacity != 0:
        pickup = False
    else:
        pickup = True
        capacity = capacity_max

    # Find potential node
```

```
    potential_output = find_potential_node(df, df_river, tau, rute, candidate,
completed, pickup, is_saving, split)
    potential_next = potential_output[0]
    potential_river = potential_output[1]

    # Demand at potential next node
    demand                                                                    =
df_demand[df_demand['location']==potential_next].iloc[0]['demand_left']

    # Calculate potential fulfil
    fulfil = min(capacity, demand)

    # Get time
    time_to_node = potential_output[2]
    time_to_park = potential_output[3]
    time_to_river = potential_output[4]
    time_to_self = potential_output[5]
    time_to_depot                                                             =
df_park_depot[df_park_depot.index==potential_next].iloc[0][rute[0]]
    time_watering = fulfil/250
    if pickup:
        time_filling = 15
    else:
        time_filling = 0

    # Time and capacity check
    if split == 'single_hard':
        # Check if vehicle can complete the node in single split
        add_trip = max(0, math.ceil((demand - capacity)/capacity_max))
        time_to_finish = current_time + time_to_node + add_trip*time_to_self +
demand/250 + time_filling + add_trip*15 + time_to_depot

        if time_to_finish <= max_time:
            next_node = potential_next
            demand_left = demand - fulfil
            capacity = capacity - fulfil
            current_time  =  current_time  +  time_to_node  +  time_watering  +
time_filling
            time_exceed = False
        else:
            next_node = rute[0]
            potential_river = ''
            demand_left = demand
            try:
                time_to_depot  =    df_park_depot[df_park_depot.index==rute[-
1]].iloc[0][rute[0]]
            except:
                time_to_depot = 0
            current_time = current_time + time_to_depot
            time_exceed = True
    else:
        if  (current_time  +  time_to_node  +  time_to_depot  +  time_watering  +
time_filling) <= max_time:
            next_node = potential_next
            demand_left = demand - fulfil
            capacity = capacity - fulfil
            current_time  =  current_time  +  time_to_node  +  time_watering  +
time_filling
            time_exceed = False
        else:
            next_node = rute[0]
            potential_river = ''
```

```
            demand_left = demand
            try:
                time_to_depot  =    df_park_depot[df_park_depot.index==rute[-
1]].iloc[0][rute[0]]
            except:
                time_to_depot = 0
            current_time = current_time + time_to_depot
            time_exceed = True

    # Adjust time to park and river with fulfil
    time_to_park = time_to_park + time_watering
    time_to_river = time_to_river + time_filling

    return  next_node,  potential_river,  capacity,  demand_left,  fulfil,
current_time, time_to_park, time_to_river, time_to_depot, time_exceed

def   calculate_distance(rute,   distance_park_depot,   distance_park_park,
distance_park_river, distance_depot_river, fulfil):
    distance = 0
    for i in range(len(rute)-1):
        # Park depot
        try:
            try:
                distance += distance_park_depot.loc[rute[i], rute[i+1]]
            except:
                distance += distance_park_depot.loc[rute[i+1], rute[i]]
        except:
            pass

        # Park park
        try:
            distance += distance_park_park.loc[rute[i], rute[i+1]]
        except:
            pass

        # Park river
        try:
            try:
                distance += distance_park_river.loc[rute[i], rute[i+1]]
            except:
                distance += distance_park_river.loc[rute[i+1], rute[i]]
        except:
            pass

        # Park park
        try:
            distance += distance_depot_river.loc[rute[i], rute[i+1]]
        except:
            pass

    # Watering time and filtering time
    distance += fulfil/250
    distance += len(list(filter(lambda x: re.search(r'river', x), rute)))*15

    return distance

def adjust_feromon(df_rute, feromon_depot, feromon_node, rho):
    dtau_node             =             pd.DataFrame(index=feromon_node.index,
columns=feromon_node.columns).fillna(0)
    dtau_depot            =            pd.DataFrame(index=feromon_depot.index,
columns=feromon_depot.columns).fillna(0)
```

```
    # Filter success
    df_rute_use = df_rute.query('performance == "success"')

    # Calcalate dtau
    for group_ant in range(m):
        for vehicle in range(vehicle_number):
            try:
                # Get ant rute
                ant_rute  =  df_rute_use.loc[[('group_ant_'+str(group_ant+1),
'vehicle_'+str(vehicle+1))]].select_dtypes(include=['O']).squeeze().tolist()[
:-1]
                ant_rute = list(filter(lambda a: a == a, ant_rute))
                ant_rute = list(filter(lambda x: re.search(r'(depot)|(park)',
x), ant_rute))

                # Local Optimal
                lk     =     df_rute_use.loc[[('group_ant_'+str(group_ant+1),
'vehicle_'+str(vehicle+1))]].iloc[0]['total_time']

                # Adjust dtau
                for i in range(len(ant_rute)-1):
                    try:
                        try:
                            dtau_depot.loc[ant_rute[i],   ant_rute[i+1]]   +=
(1/lk)
                        except:
                            dtau_depot.loc[ant_rute[i+1],   ant_rute[i]]   +=
(1/lk)
                    except:
                        dtau_node.loc[ant_rute[i], ant_rute[i+1]] += (1/lk)
                        dtau_node.loc[ant_rute[i+1], ant_rute[i]] += (1/lk)
            except:
                pass

    # New Feromon
    feromon_depot     =     pd.DataFrame(np.array(feromon_depot)*(1-rho)     +
np.array(dtau_depot),                                  index=feromon_depot.index,
columns=feromon_depot.columns)
    feromon_node      =     pd.DataFrame(np.array(feromon_node)*(1-rho)      +
np.array(dtau_node), index=feromon_node.index, columns=feromon_node.columns)

    return feromon_depot, feromon_node

# Parameter
it_max = 50 #Iteration
m = 10 #Ants number
alpha = 1 #Feromon Index
beta = 1 #Trail Index
gamma = 1 #Saving Index
rho_init = 0.6 #Evaporation Ratio
tau_park_park       =       pd.DataFrame(np.ones((df_park_park.shape))*0.01,
index=df_park_park.index, columns=df_park_park.columns) #Feromon
tau_park_depot      =      pd.DataFrame(np.ones((df_park_depot.shape))*0.01,
index=df_park_depot.index, columns=df_park_depot.columns) #Feromon
candidate = math.ceil(df_park_park.shape[0]/4) # Filter number of candidate
vehicle_number = df_vehicle_info.shape[0] # Number of vehicle
time_window = 480 # Max time window
is_saving = True # Use saving matrix for probability
split = 'single_soft' # Split with single vehicle (single_hard, single_soft,
multiple)

# Df to store all local optimal
```

```
df_rute_list = pd.DataFrame()
df_fulfil_list = pd.DataFrame()
df_time_list = pd.DataFrame()

# Loop for iteration
for it in tqdm(range(it_max)):
    # Adjust evaporation rate
    rho = rho_init - it*(1/it_max)*rho_init

    # Df to store all rute
    index_temp = pd.MultiIndex.from_product([['group_ant_'+str(i+1) for i in
range(m)],
                                             ['vehicle_'+str(j+1) for j in
range(vehicle_number)]],
                                            names=['group_ants', 'vehicles'])
    df_rute = pd.DataFrame(index=index_temp)
    df_fulfil = pd.DataFrame(index=index_temp)
    df_time = pd.DataFrame(index=index_temp)

    # Loop for all ants
    for group_ant in range(m):

        # Node already completed
        node_complete = []

        # Df demand
        df_demand = df_park_info[['location', 'demand']].drop_duplicates()
        df_demand['demand_left'] = df_demand['demand'].copy()

        for vehicle in range(vehicle_number):
            # Rute start from depot
            rute = []
            rute.append(df_park_depot.columns[0])

            # Fullfil depot is 0
            fulfil = [0]

            # Current time depot is 0
            time_rute = [0]

            # Vehicle capacity initial
            capacity_max = df_vehicle_info.iloc[vehicle]['capacity']
            capacity = 0

            # Start time
            current_time = 0

            # Find next node
            while len(node_complete) < len(df_park_park.columns):
                # Output from alg wih capacity and time check
                if re.search(r'depot', rute[-1]):
                    output = find_next_node(df_park_depot, df_depot_river,
tau_park_depot, rute, candidate, capacity, capacity_max, current_time,
time_window, node_complete, is_saving, split)
                else:
                    output = find_next_node(df_park_park, df_park_river,
tau_park_park, rute, candidate, capacity, capacity_max, current_time,
time_window, node_complete, is_saving, split)

                # Update rute
                if output[1] != '':
                    rute.append(output[1])
```

```
                    rute.append(output[0])
                else:
                    rute.append(output[0])

                # Update fulfil
                if output[1] != '':
                    fulfil.append(0)
                    fulfil.append(output[4])
                else:
                    fulfil.append(output[4])

                # Update time rute
                if not output[9]:
                    if output[1] != '':
                        time_rute.append(output[7])
                        time_rute.append(output[6])
                    else:
                        time_rute.append(output[6])
                else:
                    time_rute.append(output[8])

                # Update capacity and current time
                capacity = output[2]
                current_time = output[5]

                # Update demand left
                df_demand.loc[df_demand['location']==output[0],
'demand_left'] = output[3]

                # Update complete
                node_complete      =      df_demand.query('demand_left      ==
0')['location'].unique().tolist()

                # Time exceed
                if output[9]:
                    break

            # Back to depot
            if not re.search(r'depot', rute[-1]):
                current_time            =            current_time            +
df_park_depot[df_park_depot.index==rute[-1]].iloc[0][rute[0]]
                fulfil.append(0)
                time_rute.append(df_park_depot[df_park_depot.index==rute[-
1]].iloc[0][rute[0]])
                rute.append(df_park_depot.columns[0])

            # Store rute to df_rute
            for n in range(len(rute)):
                df_rute.loc[('group_ant_'+str(group_ant+1),
'vehicle_'+str(vehicle+1)), 'node_'+str(n+1)] = rute[n]
            for n in range(len(fulfil)):
                df_fulfil.loc[('group_ant_'+str(group_ant+1),
'vehicle_'+str(vehicle+1)), 'node_'+str(n+1)] = fulfil[n]
            for n in range(len(time_rute)):
                df_time.loc[('group_ant_'+str(group_ant+1),
'vehicle_'+str(vehicle+1)), 'node_'+str(n+1)] = time_rute[n]

            # Calculate total time
            df_rute.loc[('group_ant_'+str(group_ant+1),
'vehicle_'+str(vehicle+1)), 'vehicle_time'] = current_time
```

```
#                                 df_rute.loc[('group_ant_'+str(group_ant+1),
'vehicle_'+str(vehicle+1)),    'vehicle_time']    =    calculate_distance(rute,
df_park_depot, df_park_park, df_park_river, df_depot_river, fulfil)

        # Label fail or success
        if len(node_complete) < len(df_park_park.columns):
            df_rute.loc['group_ant_'+str(group_ant+1), 'performance'] = 'fail'
        else:
            df_rute.loc['group_ant_'+str(group_ant+1),    'performance']    =
'success'

    # Calculate total time by group ants
    df_rute = df_rute.reset_index()
    df_rute['total_time']                                                    =
df_rute.groupby('group_ants')['vehicle_time'].transform('sum')
    df_rute = df_rute.set_index(['group_ants', 'vehicles'])

    # Rearrange columns
    df_rute                                                                  =
pd.merge(df_rute[df_rute.columns[~df_rute.columns.isin(['vehicle_time',
'performance'])]],   df_rute[['vehicle_time',   'performance']],   how='left',
left_index=True, right_index=True)

    # Get local optimal from all ants
    local_optimal          =          df_rute.query('performance          ==
"success"').reset_index().sort_values(by=['total_time',         'group_ants',
'vehicles']).head(vehicle_number)

    # Adjust global optimal
    try:
        if               local_optimal.iloc[0]['total_time']               <
global_optimal.iloc[0]['total_time']:
            global_optimal = local_optimal.copy()
    except:
        global_optimal = local_optimal.copy()

    # Store local optimal to local optimal list
    local_to_store = df_rute.copy().reset_index()
    local_to_store['iteration'] = it+1
    df_rute_list = pd.concat([df_rute_list, local_to_store], sort=False)

    # Store local optimal fulfil and time
    # select_group = local_optimal["group_ants"].unique()[0]
    df_fulfil = df_fulfil.reset_index()#.query('group_ants==@select_group')
    df_fulfil['iteration'] = it+1
    df_time = df_time.reset_index()#.query('group_ants==@select_group')
    df_time['iteration'] = it+1
    df_fulfil_list = pd.concat([df_fulfil_list, df_fulfil], sort=False)
    df_time_list = pd.concat([df_time_list, df_time], sort=False)

    # Adjust feromon
    tau_park_depot,  tau_park_park  =  adjust_feromon(df_rute,  tau_park_depot,
tau_park_park, rho)

# Sorting local optimal list
df_rute_list                                                                 =
pd.concat([df_rute_list[df_rute_list.columns[~df_rute_list.columns.isin(['per
formance',        'vehicle_time',        'total_time',        'iteration'])]],
df_rute_list[['performance',  'vehicle_time',  'total_time',  'iteration']]],
axis=1)
```

```
df_fulfil_list                                                                =
pd.concat([df_fulfil_list[df_fulfil_list.columns[~df_fulfil_list.columns.isin
(['iteration'])]], df_fulfil_list[['iteration']]], axis=1)
df_time_list                                                                  =
pd.concat([df_time_list[df_time_list.columns[~df_time_list.columns.isin(['ite
ration'])]], df_time_list[['iteration']]], axis=1)

global_optimal

df_rute_list

df_fulfil_list

df_time_list

file_name1 = 'E:/tesis/output/glob_taman_soft10.xlsx'
file_name2 = 'E:/tesis/output/rute_taman_soft10.xlsx'
file_name3 = 'E:/tesis/output/fulfil_taman_soft10.xlsx'
file_name4 = 'E:/tesis/output/timelist_taman_soft10.xlsx'
global_optimal.to_excel('{}'.format(file_name1), index=False)
df_rute_list.to_excel('{}'.format(file_name2), index=False)
df_fulfil_list.to_excel('{}'.format(file_name3), index=False)
df_time_list.to_excel('{}'.format(file_name4), index=False)
```

Lampiran 6

Berikut hasil uji coba parameter :

ρ	m	i	Ket	Replikasi										Min	Mean	SD
				1	2	3	4	5	6	7	8	9	10			
0,3	5	10	Waktu (menit)	3925,05	3890	3914	3919	3905,4	3918,55	3913,87	3914,40	3914,94	3915,47	3890	3910,971	9,500409
			Jumlah Kendaraan (Unit)	9	9	9	9	9	9	9	9	9	9	9	9	0
			Running Time (Menit)	645	495	480	494	491	456	442	423	404	452	404	471,4545	66,302
		50	Waktu (menit)	3668,5	3634,85	3656,8	3653,4	3637,05	3664,7	3651,8	3647,2	3639,1	3635,7	3634,85	3647,632	12,18316
			Jumlah Kendaraan (Unit)	8	8	8	8	8	8	8	8	8	8	8	8	0
			Running Time (Menit)	1781	1938	1587	1643	1593	1673	1576	1762	1621	1552	1552	1661,636	120,7653
		100	Waktu (menit)	3650,8	3649,55	3648,3	3647,05	3645,8	3644,55	3643,3	3642,05	3640,8	3639,55	3639,55	3644,664	3,784563
			Jumlah Kendaraan (Unit)	8	8	8	8	8	8	8	8	8	8	8	8	0
			Running Time (Menit)	4320	4260	4200	4140	4080	4220	4360	4390	4384	4378	4080	4255,636	110,5831
	10	10	Waktu (menit)	3679,9	3711,55	3657,8	3671,95	3709,7	3692,18	3694,18	3696,18	3698,18	3700,18	3657,8	3688,145	16,78022
			Jumlah Kendaraan (Unit)	8	8	8	8	8	8	8	8	8	8	8	8	0

ρ	m	i	Ket	Replikasi										Min	Mean	SD
				1	2	3	4	5	6	7	8	9	10			
			Running Time (Menit)	1045	1039	1051	1043	1037	1039	1038	1037	1035	1034	1034	1039,273	5,159673
		50	Waktu (menit)	3611,9	3641,2	3656,25	3638	3647,95	3639,73	3626,62	3623,51	3620,4	3617,29	3611,9	3630,432	14,44539
			Jumlah Kendaraan (Unit)	8	8	8	8	8	8	8	8	8	8	8	8	0
			Running Time (Menit)	3712	3452	3445	3251	3621	3442	3514	3445	3474	3554	3251	3469,182	123,0546
		100	Waktu (menit)	3624,25	3622,7	3622,15	3619,6	3618,05	3626,5	3619,95	3623,4	3621,85	3620,3	3618,05	3621,527	2,496804
			Jumlah Kendaraan (Unit)	8	8	8	8	8	8	8	8	8	8	8	8	0
			Running Time (Menit)	6451	6329	6207	6085	5963	5841	5719	5597	5475	5353	5353	5852,091	369,3733
0,6	5	10	Waktu (menit)	3727,1	3763,85	3739,85	3752,1	3748,4	3746,9	3738,6	3735,1	3730,8	3729,8	3727,1	3739,964	11,54152
			Jumlah Kendaraan (Unit)	8	8	8	8	8	8	8	8	8	8	8	8	0
			Running Time (Menit)	358	420	468	425	480	435	390	445	400	455	358	421,2727	37,3726
		50	Waktu (menit)	3659,5	3646,35	3655,5	3656,2	3645,55	3647,21	3645,4	3643,6	3641,79	3639,985	3639,985	3647,37	6,615602
			Jumlah Kendaraan (Unit)	8	8	8	8	8	8	8	8	8	8	8	8	0

ρ	m	i	Ket	Replikasi										Min	Mean	SD
				1	2	3	4	5	6	7	8	9	10			
			Running Time (Menit)	1980	1860	1843	1988	1883	1891	1884,4	1877,8	1871,2	1864,6	1843	1889,636	49,28026
		100	Waktu (menit)	3642,9	3636,15	3639,4	3642,65	3635,9	3629,15	3632,4	3595,65	3619,9	3622,15	3595,65	3626,536	14,28308
			Jumlah Kendaraan (Unit)	8	8	8	8	8	8	8	8	8	8	8	8	0
			Running Time (Menit)	3200	3382	3564	3246	3228	3210	3492	3474	3556	3438	3200	3362,727	146,0327
	10	10	Waktu (menit)	3705,9	3704,15	3698,75	3705,25	3718,45	3695,16	3696,19	3691,15	3686,1	3681,06	3681,06	3696,656	10,84195
			Jumlah Kendaraan (Unit)	8	8	8	8	8	8	8	8	8	8	8	8	0
			Running Time (Menit)	809	787	718	636	625	644	643	647	651	655	625	676,3636	66,40992
		50	Waktu (menit)	3623,2	3615,1	3629,75	3632,46	3629,3	3622,3	3611,5	3619,7	3615,46	3624,3	3611,5	3621,325	6,948727
			Jumlah Kendaraan (Unit)	8	8	8	8	8	8	8	8	8	8	8	8	0
			Running Time (Menit)	3645	3300	3060	3709	3635	3546	3320	3090	3409	3633	3060	3400,636	236,2852
		100	Waktu (menit)	3625,25	3625,15	3625,05	3626,50	3625,85	3623,75	3624,50	3623,55	3623,45	3624,35	3623,45	3624,623	1,006865
			Jumlah Kendaraan (Unit)	8	8	8	8	8	8	8	8	8	8	8	8	0

ρ	m	i	Ket	Replikasi										Min	Mean	SD
				1	2	3	4	5	6	7	8	9	10			
			Running Time (Menit)	6405	6420	6435	6450	6465	6480	6495	6510	6525	6540	6405	6466,364	45,41476
0,9	5	10	Waktu (menit)	3635,5	3673,1	3645,65	3670,8	3672,6	3657,15	3668,2	3672,85	3660,8	3661,65	3635,5	3659,4	12,8
			Jumlah Kendaraan (Unit)	8	8	8	8	8	8	8	8	8	8	8,0	8,0	0,0
			Running Time (Menit)	3032	3643	1622	1646	1553	1561	1904	1902	1974	2109	1553,0	2045,4	696,6
		50	Waktu (menit)	3679,1	3647,7	3660,35	3666,85	3651,3	3650,12	3646,48	3642,84	3639,19	3635,54	3635,54	3650,455	13,31893
			Jumlah Kendaraan (Unit)	8	8	8	8	8	8	8	8	8	8	8	8	0
			Running Time (Menit)	1522	1512	1508	1506	1497	1492	1486	1481	1475	1469	1469	1492,455	17,23562
		100	Waktu (menit)	3657,45	3648,25	3639,05	3639,85	3620,65	3641,45	3620,25	3639,05	3638,85	3674,65	3620,25	3639,977	16,00312
			Jumlah Kendaraan (Unit)	8	8	8	8	8	8	8	8	8	8	8	8	0
			Running Time (Menit)	3357	3060	3363	3066	3169	3272	3575	3278	2981	3168	2981	3206,364	177,0941
	10	10	Waktu (menit)	3677,05	3674,15	3678,3	3651,25	3662,6	3674,5	3664,1	3668,3	3653,25	3661,1	3651,25	3665,077	9,629492
			Jumlah Kendaraan (Unit)	8	8	8	8	8	8	8	8	8	8	8	8	0

ρ	m	i	Ket	Replikasi										Min	Mean	SD
				1	2	3	4	5	6	7	8	9	10			
			Running Time (Menit)	625	501	604	601	611	615	521	614	601	610	501	582,1818	42,6694
		50	Waktu (menit)	3645,2	3634,45	3633,85	3651	3642,25	3644,5	3645,61	3646,65	3647,74	3648,8	3633,85	3643,082	5,721494
			Jumlah Kendaraan (Unit)	8	8	8	8	8	8	8	8	8	8	8	8	0
			Running Time (Menit)	3215	3102	3129	3111	3106	3069,9	3049	3028	3007	2986	2986	3071,718	67,22943
		100	Waktu (menit)	3630,8	3636,25	3631,7	3637,15	3642,6	3628,05	3633,5	3638,95	3644,4	3639,85	3628,05	3635,573	5,301533
			Jumlah Kendaraan (Unit)	8	8	8	8	8	8	8	8	8	8	8	8	0
			Running Time (Menit)	6620	6594	6568	6542	6516	6490	6464	6438	6412	6386	6386	6492,364	78,71891