TUGAS AKHIR – TI184833

LAPORAN TUGAS AKHIR

MULTI-OBJECTIVE VEHICLE ROUTING PROBLEM WITH TIME WINDOW AND DRONES* (MO-VRPTW-D) MENGGUNAKAN ALGORITMA *SIMULATED ANNEALING* DAN *ANT COLONY OPTIMIZATION* UNTUK *LAST-MILE DELIVERY

PENULIS:
Meidani Nuzul Tri Pamungkas
NRP. 02411840000094

DOSEN PEMBIMBING:
Prof. Ir. Budi Santosa, M.Sc., Ph.D.
NIP. 19690512 199402 1 001

DEPARTEMEN TEKNIK SISTEM DAN INDUSTRI
FAKULTAS TEKNOLOGI INDUSTRI DAN REKAYASA SISTEM
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2022

(Halaman ini sengaja dikosongkan)

FINAL PROJECT – TI184833

FINAL PROJECT REPORT

MULTI-OBJECTIVE VEHICLE ROUTING PROBLEM WITH TIME WINDOW AND DRONES (MO-VRPTW-D) USING SIMULATED ANNEALING AND ANT COLONY OPTIMIZATION ALGORITHM FOR LAST-MILE DELIVERY

AUTHOR:
Meidani Nuzul Tri Pamungkas
NRP. 02411840000094

SUPERVISOR:
Prof. Ir. Budi Santosa, M.Sc., Ph.D.
NIP. 19690512 199402 1 001

DEPARTMENT OF INDUSTRIAL AND SYSTEM ENGINEERING
FACULTY OF INDUSTRIAL TECHNOLOGY AND SYSTEMS ENGINEERING
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2022

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

MULTI-OBJECTIVE VEHICLE ROUTING PROBLEM WITH TIME WINDOW AND DRONES* (MO-VRPTW-D) MENGGUNAKAN ALGORITMA *SIMULATED ANNEALING* DAN *ANT COLONY OPTIMIZATION* UNTUK *LAST-MILE DELIVERY

TUGAS AKHIR

Diajukan untuk Memenuhi Salah Satu Syarat Memperoleh Gelar Sarjana Teknik
Program Studi S-1 Departemen Teknik dan Sistem Industri
Fakultas Teknologi Industri dan Rekayasa Sistem
Institut Teknologi Sepuluh Nopember
Surabaya

Oleh:
MEIDANI NUZUL TRI PAMUNGKAS
NRP. 02411840000094

Disetujui oleh Dosen Pembimbing Tugas Akhir

Prof. Ir. Budi Santosa, M.Sc., Ph.D.
NIP. 19690512 199402 1 001

SURABAYA, 9 FEBRUARI 2022

(Halaman ini sengaja dikosongkan)

MULTI-OBJECTIVE VEHICLE ROUTING PROBLEM WITH TIME WINDOW AND DRONES* (MO-VRPTW-D) MENGGUNAKAN ALGORITMA *SIMULATED ANNEALING* DAN *ANT COLONY OPTIMIZATION* UNTUK *LAST-MILE DELIVERY

Nama : Meidani Nuzul Tri Pamungkas

NRP : 02411840000094

Pembimbing : Prof. Ir. Budi Santosa, M.Sc., Ph.D.

ABSTRAK

Drone adalah kendaraan udara tanpa awak yang saat ini sedang marak digunakan dalam bidang fotografi dan bidang lainnya. Dalam kondisi khusus, *drone* digunakan untuk kendaraan logistik di mana barang-barang harus diangkut oleh truk yang dilengkapi dengan *drone* karena bentuk lahannya atau bahkan karena lokasinya vertikal (seperti apartemen, hotel, dll) yang tentunya tidak dapat dijangkau oleh truk. *Multi-Objective Vehicle Routing Problem with Time Window and Drones* (MO-VRPTW-D), di mana tujuannya adalah meminimasi biaya dengan cara mencari rute yang optimal agar konsumsi energi truk, konsumsi energi *drone*, dan jumlah truk yang dibutuhkan minimal. Problem ini menjadi kompleks apabila jumlah destinasi yang dikunjungi banyak. Pendekatan metaheuristik diperlukan untuk menyelesaikan problem ini karena kompleksitas yang tinggi walaupun solusi yang dihasilkan belum tentu optimal global. Algoritma *Simulated Annealing* dan *Ant Colony Optimization* dipilih karena terbukti cukup efektif untuk menyelesaikan problem kombinatorial semacam penentuan rute. Inovasi ini dapat diadopsi oleh perusahaan jasa pengiriman pada masa mendatang karena pengiriman akan menjadi lebih efisien, cepat, dan mengurangi biaya secara signifikan. Eksperimen dilakukan dalam 24 skenario dengan jumlah pelanggan 25 – 200. Algoritma ACO mampu menghasilkan solusi lebih baik daripada Algoritma SA sebanyak 15 dari total 24 skenario.

Kata Kunci: *Ant Colony Optimization*, Logistik, Metaheuristik, *Multi-Objective Vehicle Routing Problem with Time Window and Drones*, *Simulated Annealing*

(Halaman ini sengaja dikosongkan)

MULTI-OBJECTIVE VEHICLE ROUTING PROBLEM WITH TIME WINDOW AND DRONES (MO-VRPTW-D) USING SIMULATED ANNEALING AND ANT COLONY OPTIMIZATION ALGORITHM FOR LAST-MILE DELIVERY

Name : Meidani Nuzul Tri Pamungkas

NRP : 02411840000094

Supervisor : Prof. Ir. Budi Santosa, M.Sc., Ph.D.

ABSTRACT

Drone is remote controlled aerial vehicle that is currently being widely used in photography and other fields. In special conditions, drones are used for logistics vehicles where goods must be transported by trucks equipped with drones because of the shape of the land or even because of its vertical location (such as apartments, hotels, etc.) Multi-Objective Vehicle Routing Problem with Time Window and Drones (MO-VRPTW-D), where the goal is to minimize costs by finding the optimal route so that truck energy consumption, drone energy consumption, and the number of trucks needed are minimal. This problem becomes complex if the number of destinations visited is large. A metaheuristic approach is needed to solve this problem because of its high complexity, although the resulting solution is not necessarily global optimal. The Simulated Annealing and Ant Colony Optimization Algorithm was chosen because it proved to be quite effective in solving combinatorial problems such as route determination. This innovation can be applied by shipping service companies in the future because shipping will be more efficient, faster, and reduce costs significantly. Experiments were carried out in 24 scenarios with the number of customers 25 – 200. The ACO Algorithm was able to produce a better solution than the SA Algorithm by 15 out of a total of 24 scenarios.

Keywords: Ant Colony Optimization, Logistics, Metaheuristic, Multi-Objective Vehicle Routing Problem with Time Window and Drones, Simulated Annealing

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

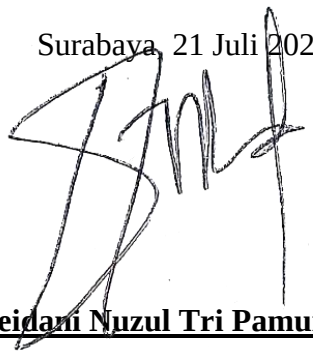
Puji syukur Penulis panjatkan atas kehadirat Tuhan Yang Maha Esa yang telah memberi rahmat dan karunia-Nya sehingga Penulis dapat menyelesaikan Laporan Tugas Akhir yang berjudul "*Multi-Objective Vehicle Routing Problem with Time Window and Drones* (MO-VRPTW-D) Menggunakan Algoritma *Simulated Annealing* dan *Ant Colony Optimization* untuk *Last-Mile Delivery*". Laporan ini disusun berdasarkan Tugas Akhir pada tanggal 21 Juli 2021 guna memenuhi kewajiban atau persyaratan kelulusan untuk program Strata 1 (S1) pada Departemen Teknik Sistem dan Industri, Fakultas Teknologi Industri dan Rekayasa Sistem, Institut Teknologi Sepuluh Nopember Surabaya. Penulis juga mengucapkan terima kasih kepada pihak-pihak yang membantu dalam pelaksanaan dan penyusunan Laporan Tugas Akhir ini baik secara materiel dan nonmateriel. Ucapan terima kasih Penulis sampaikan kepada pihak-pihak sebagai berikut.

1. Prof. Ir. Budi Santosa, M.Sc., Ph.D., selaku Dosen Pembimbing, yang telah memberikan semangat, motivasi, arahan, kritik dan saran, serta pembelajaran kehidupan kepada Penulis dalam menyelesaikan Tugas Akhir
2. Bapak Yudha Andrian Saputra S.T., M.B.A. dan Bapak Dody Hartanto S.T., M.T., selaku Dosen Penguji Seminar Proposal dan Sidang Akhir, yang telah memberikan kritik serta saran untuk kemajuan Tugas Akhir
3. Bapak Nurhadi Siswanto, S.T., MSIE., Ph.D selaku Kepala Departemen Teknik Sistem dan Industri ITS atas bimbingan dan arahan selama proses penyusunan Tugas Akhir
4. Ibu Nani Kurniati, S.T., M.T., Ph.D., selaku Sekretaris Departemen Teknik Sistem dan Industri ITS sekaligus Koordinator Tugas Akhir atas bimbingan dan arahan selama proses penyusunan Tugas Akhir
5. Bapak dan Ibu Dosen Departemen Teknik Sistem dan Industri ITS yang telah mendidik dan mengajarkan banyak ilmu serta pelajaran berharga kepada Penulis selama masa perkuliahan

6. Orang tua Penulis, yaitu Bapak Lilik Guntomo, S.Pd. dan Ibu Ninik Suntari selalu memberikan doa, motivasi, bantuan, serta dukungan kepada Penulis.
7. Laboratorium QMIPA (*Quantitative Modelling and Industrial Policy Analysis*) selaku laboratorium tempat Penulis belajar, berkembang, sekaligus mengabdi yang selalu memberikan inovasi kepada Penulis
8. Teman-teman Teknik Sistem dan Industri ITS angkatan 2018 "Nagadharma" yang memberikan doa, semangat, dan pengetahuan dalam pengerjaan Laporan Tugas Akhir
9. Pihak-pihak yang telah membantu pelaksanaan Tugas Akhir dan pengerjaan laporan ini yang tidak dapat disebutkan satu per satu

Penulis menyadari bahwa penyusunan Laporan Tugas Akhir ini masih banyak kekurangan. Oleh karena itu Penulis mengharapkan kritik dan saran dari semua pihak yang ingin memberikan kritik dan saran sebagai sebuah pengingat dan media pembelajaran bagi Penulis. Demikian Laporan Tugas Akhir ini Penulis susun, semoga bermanfaat bagi Penulis dan semua pihak. Akhir kata Penulis ucapkan terima kasih.

Surabaya, 21 Juli 2021



Meidani Nuzul Tri Pamungkas

NRP. 02411840000094

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

Pada bab ini akan dibahas mengenai Latar Belakang terkait pengerjaan Tugas Akhir, Rumusan Masalah dalam pengerjaan Tugas Akhir, Tujuan dari pengerjaan Tugas Akhir, Manfaat pengerjaan Laporan Tugas Akhir, Batasan dan Asumsi yang akan digunakan dalam pengerjaan Laporan Tugas Akhir, serta Sistematika Penyusunan Laporan Tugas Akhir.

1.1 Latar Belakang

Distribusi adalah kegiatan untuk mengirimkan produk ke pelanggan dengan rangkaian aktivitas yang dilakukan secara berulang yang berhubungan dengan pemasaran produk (Hall, 2001). Distribusi selain memberikan pengaruh harga barang, juga mempengaruhi *service level*. Tanpa mengurangi kualitas, pihak konsumen selalu menginginkan pengiriman barang yang cepat, sedangkan pihak perusahaan juga menginginkan biaya distribusi yang murah. Hal tersebut menjadi *trade off* mengingat distribusi yang cepat membutuhkan biaya yang mahal. Dengan adanya *trade off* seperti itu, tentu hal ini menjadi masalah apabila sebuah perusahaan kehilangan konsumen karena konsumen tidak puas dengan pelayanan distribusi yang diberikan oleh perusahaan yang membuat harga barang menjadi mahal. Dari pihak konsumen, tentu akan memilih alternatif lain (membeli produk lain) jika produk tersebut mampu memberikan harga yang lebih murah dengan kualitas yang tidak jauh berbeda.

Sudah menjadi mimpi semua pihak bahwa pengiriman cepat dengan biaya murah itu terjadi. Hal tersebut memberikan keuntungan bagi semua pihak, baik itu pihak perusahaan ataupun pihak konsumen. Untuk mewujudkan hal tersebut, perlu ditelusuri lagi terkait sistem yang terjadi di dalam proses distribusi. Salah satu cara untuk mewujudkan biasa distribusi yang cepat dan murah adalah dengan mengaplikasikan teknologi terbaru. Selain menerapkan ilmu optimasi ke dalam sistem penjadwalan pendistribusian barang, diperlukan juga untuk memperhatikan teknis pengiriman yaitu kendaraan pengangkut. Walaupun murah, teknologi yang digunakan saat ini masih kurang cepat dalam mengirim barang ke beberapa *retail partner*. Salah satu *improvement* yang dapat dilakukan adalah menggunakan *drone*. *Drone* sendiri adalah kendaraan terbang tak berawak yang

kemunculan awalnya adalah sebagai keperluan militer. Namun seiring berjalannya waktu, pemanfaatan *drone* semakin meluas dari bidang fotografi, pengiriman logistik, atau hanya sekadar hobi. Saat ini teknologi *drone* berkembang dengan pesat dan digadang-gadang mampu menjadi moda transportasi masa depan. Hal tersebut disebabkan *drone* mampu menuju tempat manapun dengan cepat dan mudah tanpa mempertimbangkan medan. Selain itu, biaya operasional *drone* lebih murah dibandingkan kendaraan konvensional saat ini. Namun, *drone* juga memiliki kelemahan dimana kapasitas angkutnya terbatas. Hal tersebut dapat diatasi dengan mengkombinasikan antara truk dengan *drone* untuk mengangkut logistik ke *customer*. Bukan tidak mungkin perihal pengiriman distribusi yang modern, cepat, dan murah menjadi kenyataan apabila perhitungan dilakukan dengan cermat dan sistem dapat diintegrasikan dengan baik.

Gambar 1 Kombinasi Truk dan *Drone* untuk Pengiriman Barang
Sumber: https://www.fromthegrapevine.com/

Pemanfaatan *drone* sebagai kendaraan logistik sebenarnya sudah banyak diterapkan di negara lain. Misalnya saja perusahaan asal Tiongkok yaitu JD.com yang telah lebih dulu menerapkan pengiriman barang menggunakan *drone* untuk daerah terpencil pada saat promo tahunan. JD.com mengklaim mengirim barang menggunakan *drone* sangatlah efektif baik dari sisi biaya maupun dari sisi kecepatan pengiriman. Penggunaan *drone* untuk pengiriman barang di Indonesia juga dikenalkan pertama kali oleh perusahaan serupa cabang Indonesia yaitu JD.ID. Uji coba perdana pengiriman barang menggunakan *drone* dilakukan di kawasan Bogor, Jawa Barat. Dalam uji coba yang dilakukan, drone digunakan

untuk mengirim tas ransel dan buku-buku untuk donasi. Drone tersebut terbang melintasi rute dari Desa Jagabita, Parung Panjang, Bogor sampai ke Sekolah Dasar MIS Nurul Falah. Selain JD.ID, perusahaan lain juga sempat melakukan uji coba diantaranya adalah Bukalapak dan JNE yang melakukan uji coba ini karena ingin mencari solusi yang lebih efisien dalam hal pengiriman barang. Namun banyak kendala yang dilalui oleh pihak terkait dalam merealisasikan inovasi ini diantaranya adalah regulasi pemerintah serta perlu penyesuaian lebih lanjut untuk diterapkan di Indonesia. Belum ada regulasi spesifik yang mengatur penggunaan *drone* logistik di Indonesia. Saat ini hanya terdapat regulasi umum yang dikeluarkan oleh pemerintah yaitu Peraturan Menteri No. 90 Tahun 2015 di mana dalam peraturan tersebut menegaskan bahwa menerbangkan *drone* hanya ada dua inti peraturan, yaitu adanya zona larangan terbang dan batas ketinggian, yang dibatasi hanya sampai dengan ketinggian 150 meter saja. Selain itu, peraturan Kementerian Perhubungan Nomor 163 Tahun 2015 juga menjelaskan tentang sertifikasi dan registrasi penerbangan. Peraturan terkait juga ada dalam Peraturan Kementerian Perhubungan Nomor 180 Tahun 2015 dan Nomor 47 Tahun 2016 yang mengatur larangan menerbangkan *drone* di wilayah ruang udara di Indonesia.

Penelitian serupa pernah dipimpin oleh Jun-qing Li dari School of Information and Engineering, Shandong Normal University dengan membahas topik serupa. Jun-qing Li sendiri mengatakan bahwa MO-VRPTW-D (*Multiple-Objective Vehicle Routing Problem with Time Window and Drones*) mampu melakukan penghematan sebesar 75% dibandingkan dengan pengiriman manual menggunakan truk. MO-VRPTW-D (*Multiple-Objective Vehicle Routing Problem with Time Window and Drones*) merupakan *upgrade* dari VRPD (*Vehicle Routing Problem with Drones*) dengan menambah konstrain *time window* dan dengan multi-objektif. Tujuan dari penelitian ini adalah untuk mencari inovasi baru dalam dunia optimasi rute menggunakan metode yang di-*improve* yang mereka sebut dengan *Improved Arficial Bee Colony* (IABC) yang merupakan modifikasi dari metode *Artificial Bee Colony* (ABC). Penelitian ini berfokus untuk mencari rute yang menghasilkan konsumsi energi truk, konsumsi energi *drone*, dan jumlah truk minimal dalam setiap perjalanan berdasarkan kendala (konstrain) yang ada.

Dalam *paper* dijelaskan bahwa metode yang mereka pakai (IABC) mampu mendominasi 70% solusi terbaik dari total solusi (55 percobaan) dengan membandingkan metode IABC dengan metode yang sudah sering dipakai sebelumnya yaitu *Genetic Algorithm* (GA), *Tabu Search* (TS), dan *Variable Neighborhood Search* (VNS) di mana metode ini sudah dikenal handal dalam menyelesaikan masalah optimasi kombinatorial.

Dalam penelitian Tugas Akhir ini diharapkan mampu memberikan inovasi dalam dunia pengiriman logistik untuk efisiensi biaya dan meningkatkan *service level*. Dalam kasus rumit seperti MO-VRPTW-D (*Multiple-Objective Vehicle Routing Problem with Time Window and Drones*) dengan jumlah destinasi yang banyak, tidak bisa menggunakan optimasi metode eksak karena akan memakan waktu yang banyak serta ruang komputer yang banyak pula. Diperlukan metode metaheuristik untuk memperoleh hasil setidaknya mendekati optimal karena metode metaheuristik mampu menyeleseaikan problem semacam ini dengan perhitungan waktu komputasi yang relatif cepat. Jika dalam penelitian *paper* yang dipimpin oleh Jun-qing Li menggunakan metode *Genetic Algorithm* (GA), *Tabu Search* (TS), dan *Variable Neighborhood Search* (VNS), dalam penelitian Tugas Akhir ini yang akan dipakai adalah Algoritma *Simulated Annealing* (dikembangkan oleh M. Pincus pada tahun 1970) dan *Ant Colony Optimization* (dikembangkan oleh Marco Dorigo pada tahun 1992) di mana metode ini mengadopsi simulasi pendinginan baja untuk mencari energi internal dalam baja tersebut. Metode ini juga dikenal handal dalam menyelesaikan permasalahan optimasi kombinatorial.

Dalam penelitian Tugas Akhir ini pula Penulis ingin mengetahui apa saja yang akan menjadi batasan serta menggali lebih dalam mengenai algoritma yang valid dan mampu memberikan solusi yang mendekati optimal hingga menyaingi metode-metode lain yang sudah ada sebelumnya. Harapan Penulis adalah inovasi ini setidaknya mampu menjadi bahasan serius bagi perusahaan di kemudian hari (walaupun tidak diterapkan dalam waktu dekat) dan mampu memberikan *improvement* yang signifikan bagi perusahaan terkait. Dalam Algoritma *Simulated Annealing* dan *Ant Colony Optimization* untuk MO-VRPTW-D (*Multiple-Objective Vehicle Routing Problem with Time Window and Drones*) nantinya akan

memberikan solusi berupa pemilihan rute tiap kendaraan (truk yang membawa barang dan *drone*) di mana rute ini akan memberikan nilai konsumsi energi truk, konsumsi energi *drone*, dan jumlah truk yang minimal. Problem yang merupakan optimasi kombinatorial ini secara sistem untuk mencari solusi hanya perlu merubah urutan atau kombinasi dari setiap solusi yang tersedia. Sehingga dengan Algoritma *Simulated Annealing* dan *Ant Colony Optimization* ini seharusnya mampu memberikan iterasi solusi yang terstruktur dan berakhir dengan kurva datar (konvergen).

1.2 Rumusan Masalah

Rumusan masalah pada pengerjaan Tugas Akhir dengan topik Optimasi Metaheuristik sebagai berikut.

1. Apa saja yang menjadi konstrain dalam kasus *Multi-Objective Vehicle Routing Problem with Time Window and Drones* (MO-VRPTW-D)?
2. Bagaimana mendapatkan rute truk dan *drone* dalam kasus *Multi-Objective Vehicle Routing Problem with Time Window and Drones* (MO-VRPTW-D) yang menghasilkan total biaya minimum dengan Algoritma *Simulated Annealing* dan *Ant Colony Optimization*?

1.3 Tujuan

Tujuan yang ingin dicapai pada pengerjaan Tugas Akhir dengan topik Optimasi Metaheuristik sebagai berikut.

1. Mencari alternatif terbaik antara Metode *Weighted-Sum* dan *Lexicographic* untuk menyelesaikan kasus *multi-objective* pada *Multi-Objective Vehicle Routing Problem with Time Window and Drones* (MO-VRPTW-D)
2. Mengembangkan Algoritma *Simulated Annealing* dan *Ant Colony Optimization* untuk menyelesaikan kasus *Multi-Objective Vehicle Routing Problem with Time Window and Drones* (MO-VRPTW-D)
3. Mendapatkan Algoritma *Simulated Annealing* dan *Ant Colony Optimization* yang sudah tervalidasi sesuai dengan konstrain pada kasus *Multi-Objective Vehicle Routing Problem with Time Window and Drones* (MO-VRPTW-D)

4. Memperoleh hasil yang mendekati optimal untuk kasus *Multi-Objective Vehicle Routing Problem with Time Window and Drones* (MO-VRPTW-D)
5. Membandingkan hasil antara Algoritma *Simulated Annealing* (SA) dan *Ant Colony Optimization* (ACO) untuk kasus *Multi-Objective Vehicle Routing Problem with Time Window and Drones* (MO-VRPTW-D)

1.4 Manfaat

Manfaat pengerjaan Laporan Tugas Akhir dengan topik Optimasi Metaheuristik sebagai berikut.

1. Memberikan inovasi baru yang mungkin bisa diterapkan untuk kasus pendistribusian barang yang lebih cepat dan murah di Indonesia dengan mengadopsi *drone*
2. Menyediakan alternatif baru dengan menerapkan Algoritma *Simulated Annealing* dan *Ant Colony Optimization* untuk kasus *Multi-Objective Vehicle Routing Problem with Time Window and Drones* (MO-VRPTW-D)

1.5 Batasan dan Asumsi

Pada Penulisan Laporan Tugas Akhir, terdapat batasan dan asumsi dalam proses pengerjaannya di mana batasan dan asumsi dalam pengerjaan Laporan Tugas Akhir adalah sebagai berikut.

1.5.1 Batasan

Batasan yang diterapkan dalam pengerjaan Laporan Tugas Akhir ini adalah sebagai berikut.

1. Waktu pengerjaan Laporan Tugas Akhir pada tanggal 23 Agustus 2021 – 21 Januari 2022
2. Data yang digunakan untuk pengujian adalah data sekunder yang dapat dianggap valid pada penelitian Solomon dan Gehring & Homberger dengan penambahan data *random* oleh Penulis
3. Setiap truk hanya membawa satu *drone* dan setiap *drone* hanya membawa paket tidak melebihi kapasitas angkut
4. Terdapat hanya satu depot (titik berangkat dan titik pulang)
5. Konsumsi energi hanya dibatasi oleh konsumsi bahan bakar

1.5.2 Asumsi

Asumsi diterapkan dalam pengerjaan Laporan Tugas Akhir ini adalah sebagai berikut.

1. Matriks jarak bersifat konstan
2. Kecepatan truk dan *drone* dianggap konstan
3. Konsumsi bahan bakar hanya didapat dari jarak tempuh (tidak terpengaruh oleh berat barang yang dibawa)
4. Truk dan *drone* diasumsikan tidak habis bahan bakar di tengah jalan
5. *Drone* tidak memiliki batas maksimal waktu tempuh
6. Konsumsi bahan bakar truk tidak terhitung ketika truk berhenti untuk melayani pelanggan

1.6 Sistematika Penyusunan Laporan

Pada sub bab ini akan diuraikan fokus utama setiap bab pada Laporan Tugas Akhir yang disusun. Berikut adalah uraian fokus utama untuk setiap bab yang dimuat dalam Laporan Tugas Akhir.

1.6.1 BAB I: PENDAHULUAN

Pada bab ini akan dibahas mengenai pendahuluan berupa hal yang mendasari Penulisan Laporan Tugas Akhir. Pada bab ini juga menjawab pertanyaan terkait mengapa laporan ini dibuat dan ditulis. Beberapa bagian yang terdapat di bab ini adalah Latar Belakang terkait pengerjaan Tugas Akhir, Rumusan Masalah dalam pengerjaan Tugas Akhir, Tujuan dari pengerjaan Tugas Akhir, Manfaat pengerjaan Laporan Tugas Akhir, Batasan dan Asumsi yang akan digunakan dalam pengerjaan Laporan Tugas Akhir, serta Sistematika Penyusunan Laporan Tugas Akhir.

1.6.2 BAB II: TINJAUAN PUSTAKA

Pada bab ini akan dibahas mengenai informasi dasar terkait apa saja metode dan istilah yang akan digunakan dalam Laporan Tugas Akhir. Pada bab ini juga menjawab pertanyaan terkait apa saja yang akan digunakan dalam Laporan Tugas Akhir ini. Beberapa bagian yang terdapat di bab ini adalah Notasi yang Dipakai pada MO-VRPTW-D, macam-macam *Routing Problem*, dasar ilmu Algoritma *Push-Forward Insertion Heuristic*, *Simulated Annealing*, dan *Ant Colony Optimization*, serta analisis terhadap Penelitian Terkait.

1.6.3 BAB III: METODOLOGI PENELITIAN

Pada bab ini akan dibahas mengenai Metodologi Penelitian dalam pengerjaan laporan. Metodologi tersebut meliputi Metode Pengerjaan Tugas Akhir dalam bentuk *flowchart*, Deskripsi Permasalahan VRPTW dan MO-VRPTW-D, Pengembangan Algoritma PFIH, SA, dan ACO, Rekap Data untuk pengujian, Validasi dari algoritma terkait, Eksperimen dan Analisis algoritma, serta Kesimpulan dan Saran dari eksperimen algoritma.

1.6.4 BAB IV: EKSPERIMEN DAN ANALISIS

Pada bab ini akan dibahas mengenai eksperimen algoritma menggunakan *software* MATLAB melalui beberapa skenario yang telah disusun. Pada bab ini juga menjawab pertanyaan terkait seberapa bagus algoritma yang telah dibuat untuk menyelesaikan kasus MO-VRPTW-D. Beberapa bagian yang terdapat di bab ini adalah Pengujian Algoritma melalui beberapa skenario, Rekap Hasil dari Pengujian Algoritma, serta Analisis hasil dan performansi menggunakan perbandingan antar algoritma dari yang sudah dibuat.

1.6.5 BAB V: KESIMPULAN DAN SARAN

Pada bab ini akan dibahas mengenai penarikan kesimpulan dan pemberian saran untuk beberapa pihak. Harapan dari pihak Penulis dalam bab ini adalah agar semua pihak mampu terbuka serta bersedia menerima kritik dan saran demi kebaikan bersama yaitu bisa melakukan *improvement* terhadap sesuatu yang dilakukan sebelumnya. Beberapa bagian yang terdapat di bab ini adalah Kesimpulan dari Laporan Tugas Akhir yang menjawab Tujuan pada Bab I serta Saran untuk penelitian-penelitian selanjutnya.

BAB II
TINJAUAN PUSTAKA

Pada bab ini akan dibahas mengenai Notasi Permasalahan pada MO-VRPTW-D, macam-macam *Routing Problem*, dasar ilmu Algoritma *Push-Forward Insertion Heuristic*, *Simulated Annealing*, dan *Ant Colony Optimization*, serta analisis terhadap Penelitian Terkait serupa dengan Topik Tugas Akhir.

2.1 *Multi-Objective Optimization*

Multi-Objective Optimization adalah problem optimasi yang melibatkan lebih dari satu fungsi tujuan untuk dioptimalkan secara bersamaan (simultan). Problem optimasi seperti ini tidak hanya berbentuk linear, tetapi juga bisa jadi memiliki bentuk yang *non-linear*. *Multi-Objective Optimization* telah diterapkan di banyak bidang ilmu pengetahuan, termasuk teknik, ekonomi dan logistik di mana keputusan yang optimal perlu diambil dengan adanya *trade-off* antara dua atau lebih tujuan yang telah ditetapkan. Misalnya saja, pada kasus pembelian mobil, pembeli ingin meminimalkan biaya serta memaksimalkan kenyamanan atau pada kasus penjadwalan produksi yang memiliki tujuan untuk meminimumkan biaya produksi sekaligus memaksimalkan penjualan adalah contoh masalah optimasi *multi-objective*. Dalam masalah yang lebih kompleks, bisa terdapat lebih dari tiga fungsi objektif. Hampir seluruh kasus yang terdapat di dunia nyata adalah problem optimasi *multi-objective*. Misalkan N adalah jumlah fungsi objektif dan n adalah himpunan N, K adalah jumlah konstrain *equality* dan k adalah himpunan K, serta L adalah banyaknya konstrain *inequality* dan l adalah himpunan dari L, maka persamaan matematis secara umum dari permasalahan *multi-objective* dapat ditulis seperti berikut.

Multi-objective function:

$$min/max\ f_n(x)$$

dengan:

$$n = 1, \ldots, N \tag{2.1}$$

Subject to:

$$g_k(x) = 0 \qquad k = 1, \ldots, K \tag{2.2}$$

$$h_l(x) \leq 0 \qquad l = 1, \ldots, L \tag{2.3}$$

Optimasi *multi-objective* tidak bisa diselesaikan langsung tanpa proses transformasi. Terdapat dua metode untuk menyelesaikan kasus optimasi multi-objektif Wiley et al. (2001). Terdapat Metode *Lexicographic* dimana objektif lain akan dimasukkan ke dalam konstrain. Metode lainnya yaitu *Weighted-Sum Method* dimana masing-masing objektif akan diberi bobot sesuai dengan prioritasnya. Sebenarnya masih terdapat banyak metode lain yang merupakan pengembangan dari dua metode sebelumnya, namun tidak akan dibahas dalam penelitian ini. Berikut adalah penjelasan Metode *Weighted-Sum* dan *Lexicographic*, diberikan persamaan objektif (2.4) dengan sembarang konstrain.

Multi-objective function:

$$min(Z_1, Z_2, Z_3) \tag{2.4}$$

2.1.1 Metode Weighted-Sum

Pada Metode *Weighted-Sum*, seluruh objektif harus dinormalisasi satuannya. Kemudian, objektif akan diberi bobot sesuai dengan prioritas objektif tersebut. Semakin besar bobotnya, maka objektif tersebut semakin diprioritaskan untuk dioptimasi ($\sum_{i=1}^{N} w_i = 1$). Berikut adalah contoh perubahan fungsi *multi-objective* menggunakan Metode *Weighted-Sum.*

Multi-objective function:

$$min(w_1 Z_1 + w_2 Z_2 + w_3 Z_3) \tag{2.5}$$

2.1.2 Metode Lexicographic

Pada Metode *Lexicographic*, problem akan diubah menjadi *single objective*. Setiap objektif akan dicari nilai minimum yang mungkin, kemudian ditranformasi menjadi *inequality constraint.* Untuk menentukan objektif mana yang akan dijadikan konstrain, perlu dilakukan uji coba pada setiap objektif. Berikut adalah penyelesaian *multi-objective* dengan Metode *Lexicographic*.

Objective function:

$$min\ Z_1 \tag{2.6}$$

Subject to:

$$Z_2 \geq y_1 \tag{2.7}$$

$$Z_3 \geq y_2 \tag{2.8}$$

Note: y_i adalah nilai minimum yang memungkinkan dari masing-masing konstrain

2.2 *Routing Problem*

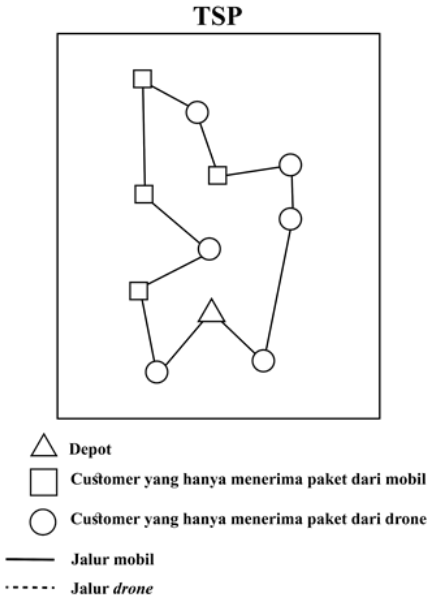
Routing Problem atau permasalahan perutean adalah proses pembentukan rute dari kendaraan yang berangkat dari depot untuk mengunjungi seluruh titik lalu kembali ke titik asal (depot) dengan tujuan mencari jarak, waktu, ataupun biaya minimum. *Routing Problem* banyak diterapkan dalam dunia distribusi untuk mencari solusi yang optimal guna memperoleh rute dengan biaya minimum. Sistem perutean dapat ditentukan oleh beberapa matriks dalam sistem logistik dan rantai pasokan dengan mencari bagaimana proyeksi matriks pelanggan dan di mana lokasi terdekat dari barang yang diproduksi (Rizal Putranto, 2014). *Routing Problem* tergolong sebagai permasalahan optimasi kombinatorial. Optimasi kombinatorial adalah problem optimasi di mana alternatif solusinya dapat dibentuk dengan membolak-balikkan urutan dari solusi sebelumnya dan menghitung kembali *fitness*/fungsi objektif. Untuk problem kecil, hal ini tidak menjadi masalah apabila hanya menggunakan ILP. Namun untuk problem besar, diperlukan metode yang tepat untuk membantu membolak-balikkan urutan rute tersebut sehingga solusi menjadi optimal dalam waktu singkat. Berikut adalah perbandingan untuk beberapa kasus *routing* dengan ciri khas masing-masing diantaranya adalah *Travelling Salesman Problem* (TSP), *Flying Sidekick Travelling Salesman Problem* (FSTSP), *Parallel Drone Scheduling Travelling Salesman Problem* (PDSTSP), *Vehicle Routing Problem* (VRP), *Capacitated Vehicle Routing Problem* (CVRP), *Vehicle Routing Problem with Time Window* (VRPTW), *Vehicle Routing Problem with Drones* (VRPD), dan *Multi-Objective Vehicle Routing Problem with Time Window and Drones* (MO-VRPTW-D).

Tabel 1 Perbandingan Karakteristik *Routing Problem*

Pembanding	**TSP**	**FSTSP**	**PDSTSP**	**VRP**	**CVRP**	**VRPTW**	**VRPD**	**MO-VRPTW-D**
Objektif	1	1	1	1	1	1	1	Multi
#Truk	1	1	1	Multi	Multi	Multi	Multi	Multi
Truck Capacity	-	√	√	-	√	-	√	√
Time Window	-	-	-	-	-	√	-	√

Pembanding	TSP	FSTSP	PDSTSP	VRP	CVRP	VRPTW	VRPD	MO-VRPTW-D
#*Drone* per Truk	-	1	1	-	-	-	1	1
Drone Capacity	-	1	1	-	-	-	1	Multi
Tandem/Paralel?	-	T	P	-	-	-	T	T

2.2.1 Travelling Salesman Problem



Gambar 2 *Travelling Salesman Problem* (TSP)

Travelling Salesman Problem (TSP) adalah masalah penentuan rute satu kendaraan dari satu depot dan kembali ke depot tersebut dengan tujuan mencari biaya minimum dari perjalanan total. Persyaratan atau kendala dari TSP adalah setiap kota wajib dikunjungi dan hanya dikunjungi satu kali. Setiap penambahan jumlah kota yang harus dikunjungi, jumlah solusi yang tersedia (*feasible*) meningkat secara eksponensial yang menyebabkan pencarian rute optimal menjadi suatu hal yang sulit ditemukan. TSP yang tergolong optimasi kombinatorial menjadi kompleks apabila permasalahan dalam skala besar sehingga TSP diklasifikasikan sebagai NP-*complete problem.* TSP bisa diselesaikan dengan metode *Integer Linear Programming* (ILP). Banyak modifikasi dari TSP, misal saja pemodelan rute tol laut yang mengharuskan kapal

harus berhenti di titik tertentu dan kembali ke depot dengan rute yang sama saat awal berangkat. Pada kondisi ini, tiap titik (termasuk depot) wajib dikunjungi selama dua kali (berangkat dan pulang) kecuali titik terakhir pemberhentian sebelum kembali ke depot. Adapun modifikasi lainnya dengan menambahkan kendaraan *drone* yang dioperasikan bersamaan dengan truk. Pada bagian ini akan dijelaskan mengenai batasan atau konstrain dalam TSP berdasarkan Miller, Tucker, dan Zemlin serta problem yang sebelumnya sudah disebutkan (disertai penambahan *drone*) yaitu *Flying Sidekick Travelling Salesman Problem* (FSTSP) dan *Parallel Drone Scheduling Travelling Salesman Problem* (PDSTSP).

Objective function:

$$min\left(\sum_{i=1}^{n}\sum_{j\neq i,j=1}^{n} c_{ij}x_{ij}\right)$$

dengan:

$$i,j = 1,\dots,n \tag{2.9}$$

Subject to:

$$\sum_{i=1,i\neq j}^{n} x_{ij} = 1 \qquad j = 1,\dots,n \tag{2.10}$$

$$\sum_{j=1,j\neq i}^{n} x_{ij} = 1 \qquad i = 1,\dots,n \tag{2.11}$$

$$u_i - u_j + nx_{ij} \leq n-1 \qquad 2 \leq i \neq j \leq n \tag{2.12}$$

$$1 \leq u_i \leq n-1 \qquad 2 \leq i \leq n \tag{2.13}$$

$$x_{ij} \in \{0,1\} \qquad i,j = 1,\dots,n \tag{2.14}$$

$$u_i \in \mathbb{Z} \qquad i = 2,\dots,n \tag{2.15}$$

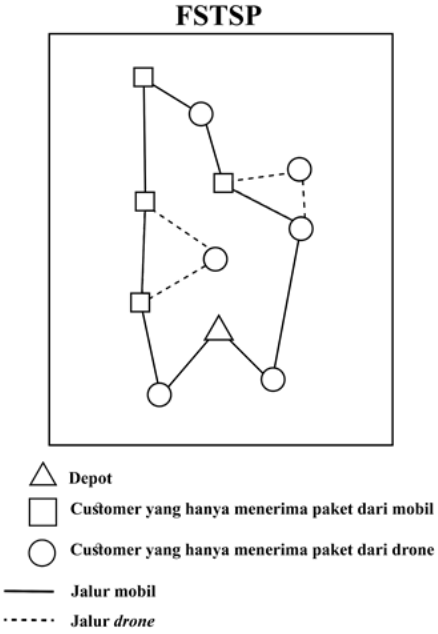
Decision variables:

$$x_{ij} = \begin{cases} 1, jika\ kendaraan\ melewati\ node-i\ ke\ node-j \\ 0, jika\ tidak \end{cases} \tag{2.16}$$

Persamaan (2.9) menunjukkan fungsi objektif dimana dalam kasus TSP yang diinginkan adalah minimasi biaya total. Persamaan (2.10) mensyaratkan bahwa setiap kota tiba dari tepat satu kota lain. Sebaliknya, persamaan (2.11) mensyaratkan bahwa dari setiap kota ada keberangkatan tepat ke satu kota lain. Untuk persamaan (2.12), menunjukkan bahwa kota j dikunjungi setelah

mengunjungi kota *i*. Persamaan (2.13) mengharuskan hanya ada satu perjalanan yang mencakup semua kota, dan bukan dua atau lebih perjalanan terputus yang hanya secara kolektif mencakup semua kota (rute tersambung mencegah adanya *subtour*). Untuk membuat seperti kendala yang sudah disebutkan, akan ditunjukkan bahwa setiap solusi yang *feasible* hanya berisi satu urutan rute yang tersambung yang mencakup semua kota (menghindari *subtour*), terdapat nilai untuk variabel *dummy* u_i yang memenuhi konstrain dimana variabel *dummy* menunjukkan urutan rute, sehingga $u_i<u_j$ mengisyaratkan bahwa kota *i* dikunjungi sebelum kota *j*. Kondisi seperti ini dapat dicapai dengan menambah u_i setiap kali kota dikunjungi.

1. *Flying Sidekick Travelling Salesman Problem* (FSTSP)



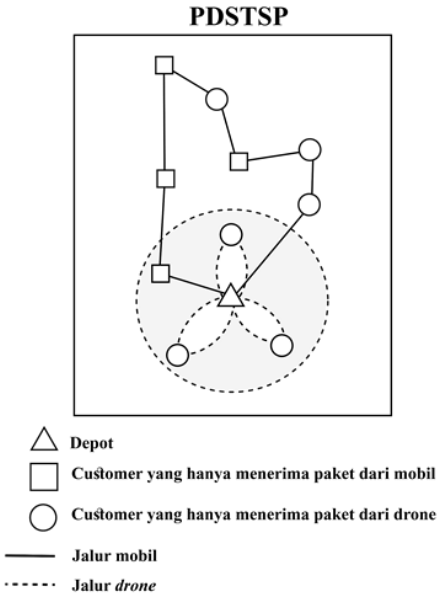
Gambar 3 *Flying Sidekick Travelling Salesman Problem* (FSTSP)

Flying Sidekick Travelling Salesman Problem (FSTSP) adalah pengembangan TSP dengan mengombinasikan kendaraan truk dengan *drone* (satu truk dan satu *drone*) untuk mengirim paket ke *customers* secara tandem. Dalam FSTSP sendiri truk dan drone bekerja secara tandem yang artinya truk dan *drone* bekerja bersama. Truk mengangkut barang sekaligus mengangkut *drone*. Hal ini bertujuan agar energi *drone* digunakan secara efisien (dimatikan ketika dimuat truk dan dinyalakan hanya ketika akan mengantar barang). Kapasitas *drone* hanya satu barang sehingga rute yang akan dipilih untuk *drone* harus benar-benar

diperhatikan. Menurut Artikel "Studi Komparasi Penggunaan *Drone* untuk Logistik *Last-Mile*" berikut adalah aturan yang terdapat di FSTSP.

- Terdapat satu buah truk dan satu buah *drone*
- Kapasitas *drone* hanya satu barang
- Selama *drone* mengantar barang, truk dapat berpindah tempat bahkan lebih dari satu kali
- *Drone* tidak dapat mendarat untuk menghemat daya baterai jika telah sampai di lokasi sebelum truk karena *drone* terbang dengan kecepatan konstan selama mengantar barang ini kecuali saat memberikan paket kepada *customer*
- Jika *drone* mendarat di truk di titik i, *drone* dapat terbang kembali dari titik i (hanya untuk mengambil barang) namun tidak dapat kembali ke titik tersebut (hanya dikunjungi satu kali)
- *Drone* harus mendarat saat truk berhenti (tidak sedang dalam perjalanan)
- Truk juga tidak boleh kembali ke titik *customer* yang sudah pernah dikunjungi untuk mengambil *drone*

2. *Parallel Drone Scheduling Travelling Salesman Problem* (PDSTSP)



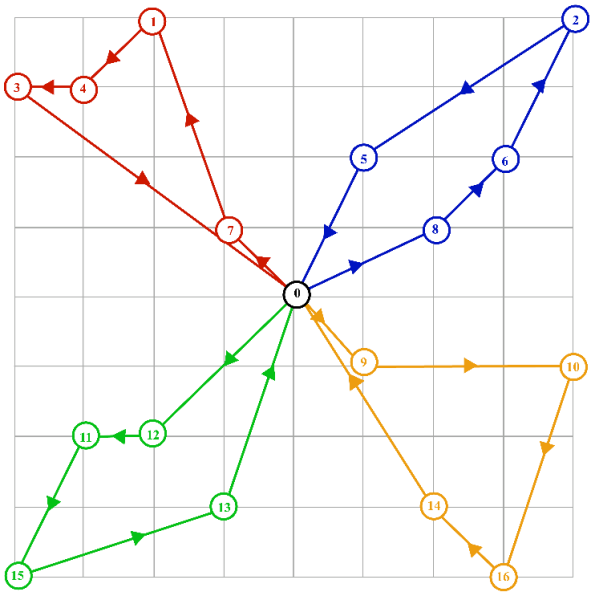
Gambar 4 *Parallel Drone Scheduling Travelling Salesman Problem* (PDSTSP)

Parallel Drone Scheduling Travelling Salesman Problem (PDSTSP) adalah pengembangan TSP dengan mengombinasikan kendaraan truk dengan *drone* (satu truk dan satu *drone*) untuk mengirim

paket ke *customers*. Perbedaan antara PDSTSP dan FSTSP adalah PDSTSP bekerja secara paralel antara truk dan *drone*. Truk dan *drone* bekerja secara paralel yang artinya truk dan *drone* akan berangkat secara terpisah (*drone* tidak dimuat dalam truk) dari depot. Kapasitas *drone* terbatas hanya satu barang, sehingga *drone* harus kembali ke depot untuk mengambil barang. Oleh karena itu rute untuk *drone* harus diperhatikan, tidak boleh terlalu jauh mengingat energi *drone* terbatas. Menurut Artikel "Studi Komparasi Penggunaan *Drone* untuk Logistik *Last-Mile*" berikut adalah aturan yang terdapat di PDSTSP.

- Terdapat satu buah truk dan satu buah *drone*
- *Drone* memiliki batas jarak maksimum (dipengaruhi oleh daya yang terbatas) di mana daya diisi di depot
- *Customer* yang tidak dapat dijangkau oleh *drone* atau jaraknya terlalu jauh dari depot akan diantarkan oleh truk
- Truk dan *drone* hanya dapat mengunjungi sebuah customer satu kali
- *Customer* yang sudah dikunjungi oleh *drone* tidak boleh dikunjungi oleh truk, begitu juga sebaliknya
- *Drone* hanya memiliki kapasitas satu barang, sehingga apabila telah mengantar satu paket *drone* akan kembali ke depot untuk mengambil barang
- Waktu pengisian daya *drone*, waktu *take off*, dan waktu pengambilan barang oleh *drone* diabaikan dalam model ini

2.2.2 *Vehicle Routing Problem*



Gambar 5 *Vehicle Routing Problem* (VRP)

Vehicle Routing Problem (VRP) adalah masalah penentuan rute kendaraan dari satu depot ke sejumlah titik tujuan. VRP merupakan masalah yang sering ditemui dalam manajemen distribusi atau logistik. VRP merupakan pengembangan dari *Travelling Salesman Problem* (TSP). Berbeda dengan TSP yang hanya membentuk satu rute untuk mengunjungi semua titik tujuan, VRP dapat membentuk beberapa rute untuk mengunjungi semua titik tujuan karena jumlah kendaraan yang dipakai lebih dari satu kendaraan. VRP bisa dikerjakan dengan menggunakan *Integer Linear Programming* (ILP). Namun apabila jumlah destinasi dan jumlah kendaraan bertambah, masalah akan menjadi semakin kompleks, sehingga metode eksak memerlukan banyak waktu untuk melakukan komputasi. Kebanyakan VRP memang seharusnya diselesaikan dengan metode Metaheuristik untuk menghindari waktu komputasi yang lama walaupun hasil yang didapat tidak optimum global. Sama dengan TSP, VRP juga tergolong optimasi kombinatorial menjadi kompleks apabila permasalahan dalam skala besar sehingga VRP diklasifikasikan sebagai NP-*complete problem.* Dalam penerapannya, kasus VRP disertai dengan jumlah permintaan di setiap titik dan kapasitas kendaraan (biasa disebut *Capacitated Vehicle Routing Problem*), dengan batas waktu buka di setiap titik tujuan (*Vehicle Routing Problem with Time Window*), atau gabungan keduanya dan masih banyak konstrain lainnya. Berikut

adalah formulasi VRP dengan konstrain kapasitas menurut model gabungan Dantzig-Fulkerson-Johnson (model dasar) dan Miller-Tucker-Zemlin (eliminasi *subtour*).

Objective function:

$$min\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{p} c_{ij}x_{ijk}\right)$$

dengan:

$$i \neq j \tag{2.17}$$

Subject to:

$$\sum_{i=1}^{n} x_{ijk} = \sum_{i=1}^{n} x_{jik} \qquad \forall j \in \{1,\ldots,n\}, \forall k \in \{1,\ldots,p\} \tag{2.18}$$

$$\sum_{k=1}^{p}\sum_{i=1}^{n} x_{ijk} = 1 \qquad \forall j \in \{2,\ldots,n\} \tag{2.19}$$

$$\sum_{j=2}^{n} x_{1jk} = 1 \qquad \forall k \in \{1,\ldots,p\} \tag{2.20}$$

$$\sum_{i=1}^{n}\sum_{j=2}^{n} q_j x_{ijk} \leq Q \qquad \forall k \in \{1,\ldots,p\} \tag{2.21}$$

$$u_j - u_i \geq q_j - Q(1 - x_{ijk}) \qquad \forall i,j \in V \setminus \{1\}\ i \neq j \tag{2.22}$$

$$q_i \leq u_i \leq Q \qquad \forall i \in V \setminus \{1\} \tag{2.23}$$

Decision variables:

$$x_{ijk} = \begin{cases} 1, jika\ kendaraan - k\ melewati\ node - i\ ke\ node - j \\ 0, jika\ tidak \end{cases} \tag{2.24}$$

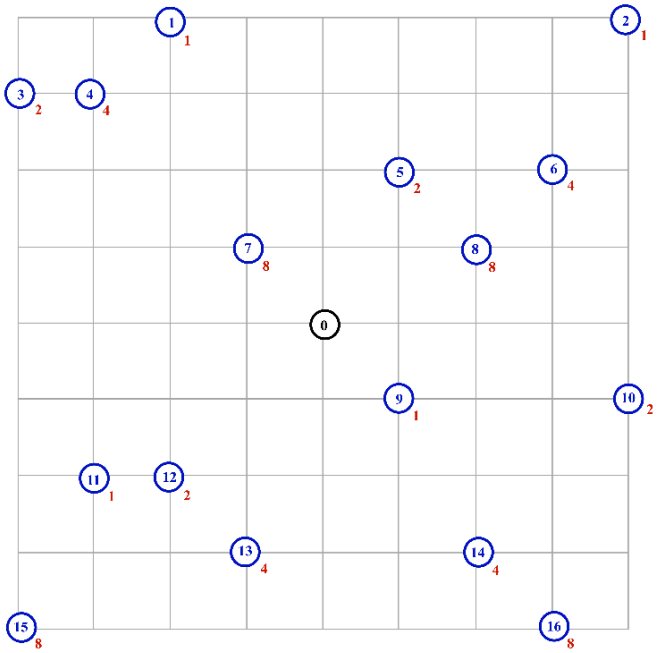
Dalam formulasi VRP, c_{ij} mewakili biaya dari node *i* ke node *j* (bisa juga diganti dengan jarak atau waktu), x_{ijk} adalah variabel biner yang memiliki nilai 1 jika kendaraan *k* dari *i* ke *j* dan bernilai 0 jika sebaliknya, serta $i \neq j$. Persamaan (2.17) menunjukkan fungsi objektif dimana dalam kasus VRP tujuannya adalah minimasi total biaya. Persamaan (2.18) menunjukkan kendaraan yang menuju kota *j* (dari kota *i*) dan pergi dari kota *i* (menuju kota *j*) adalah sama. Persamaan (2.19) menunjukkan bahwa setiap node hanya dilewati sekali. Persamaan (2.20) menunjukkan bahwa seluruh kendaraan keluar dari depot. Persamaan (2.21) menyatakan bahwa muatan tidak melebihi kapasitas kendaraan. Persamaan (2.22)

dan (2.23) adalah persamaan untuk mencegah terbentuknya *subtour*. Dengan *q* adalah *demand* dimana tidak boleh melebihi kapasitas kendaraan (*Q*). Sedangkan *V* adalah himpunan yang berisi semua node dan $n = 1$ adalah depot. Jika kendaraan *k* berangkat dari node *i* ke node *j*, maka $x_{ijk} = 1$ dan persamaan (2.22) dapat ditulis ulang menjadi $u_j \geq u_i + q_j$. Hal ini memastikan bahwa nilai u_j masih lebih besar dibanding nilai u_i dan q_j. Jadi, nilai u_j lebih besar dari nilai u_i. Jika kendaraan *k* tidak berangkat dari node *i* melewati node *j*, maka $x_{ijk} = 0$, konstrain akan tetap valid. Persamaan (2.23) dapat ditulis ulang menjadi $u_j - q_j \geq u_i - Q$. Persamaan (2.23) menyatakan bahwa q_j memiliki nilai terendah dibandingkan nilai u_i dan *Q* memiliki nilai terbesar dibandingkan dengan u_i. Jadi, $u_j - q_j$ setidaknya akan sama dengan *0* dan $u_i - Q$ paling besar nilainya adalah *0*. Jadi, nilai $u_j - q_j$ lebih besar atau sama dengan $u_i - Q$.

1. *Capacitated Vehicle Routing Problem* (CVRP)

   Capacitated Vehicle Routing Problem (CVRP) adalah VRP di mana kendaraan dibatasi oleh kapasitas angkut di mana kendaraan perlu mengambil atau mengirimkan barang di berbagai lokasi. Barang memiliki kuantitas, seperti berat atau volume, dan kendaraan memiliki kapasitas maksimum yang dapat mereka bawa. Selain itu, setiap titik juga memiliki *demand* yang harus dipenuhi pada setiap pengirimannya. Tujuan dari problem ini adalah untuk mengirimkan barang dengan biaya paling murah, dengan tidak mengabaikan kapasitas angkut kendaraan serta *demand* pada setiap titik. Ada beberapa aturan dalam CVPR diantaranya sebagai berikut.

- Kendaraan berangkat bersama-sama dari depot dan jumlah kendaraan lebih dari satu
- Setiap titik boleh dikunjungi oleh dua truk atau lebih apabila *demand* tidak bisa dipenuhi oleh satu truk
- Setiap kendaraan memiliki kapasitas angkut masing-masing
- Setiap titik memiliki *demand* masing-masing
- Total kapasitas angkut tidak boleh kurang dari total *demand*, sehingga tidak ada kendaraan yang kembali ke depot untuk mengambil barang



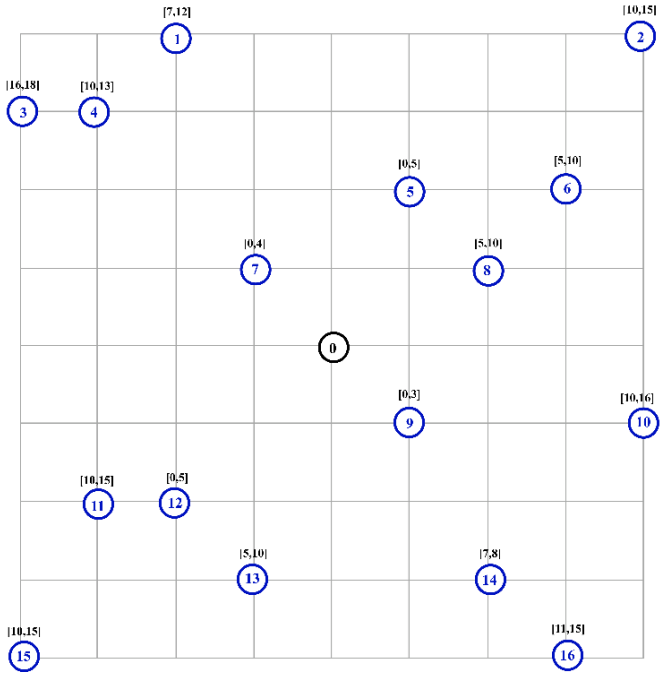
Gambar 6 *Capacitated Vehicle Routing Problem* (CVRP)

2. *Vehicle Routing Problem with Time Window* (VRPTW)

Banyak masalah perutean kendaraan melibatkan penjadwalan kunjungan ke pelanggan yang hanya tersedia selama *time window* tertentu. Problem ini disebut dengan *Vehicle Routing Problem with Time Window* (VRPTW). *Vehicle Routing Problem with Time Window* (VRPTW) adalah pengiriman barang menggunakan multi truk ke sejumlah titik dengan memperhatikan jam buka depot serta jam tutup depot dan jam buka serta jam tutup masing-masing lokasi dimana barang harus diantar. Tidak jarang VRPTW dikombinasikan dengan konstrain kapasitas pada bahasan sebelumnya karena menyesuaikan problem *real* dimana selain dibatasi oleh jam buka dan tutup, permasalahan juga dibatasi dengan kapasitas angkut pada masing-masing kendaraan. Tujuan dari VRPTW adalah mencari biaya pengiriman yang minimum dengan memperhatikan jam buka dan jam tutup pada setiap titik pengiriman. Penerapan VRPTW sendiri sesuai untuk kondisi di Indonesia di mana pengiriman kebanyakan disesuaikan dengan jam kantor. Ada beberapa aturan dalam VRPTW diantaranya sebagai berikut.

- Kendaraan berangkat bersama-sama dari depot dan jumlah kendaraan lebih dari satu

- Setiap titik hanya boleh dikunjungi oleh satu truk
- Setiap titik memiliki jam buka dan jam tutup (*time window*) masing-masing, sehingga kendaraan harus sampai di titik tersebut pada jam yang telah ditentukan



Gambar 7 *Vehicle Routing Problem with Time Window* (VRPTW)

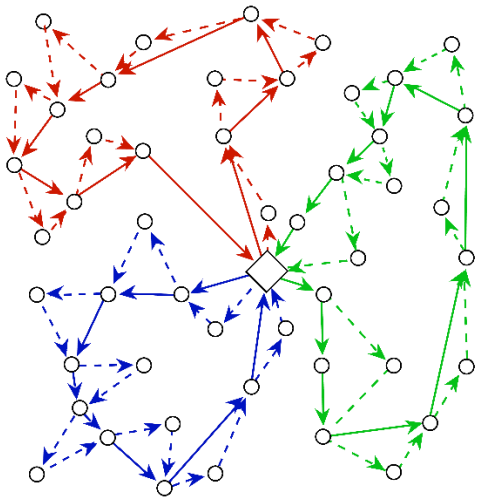
3. *Vehicle Routing Problem with Drones* (VRPD)

Vehicle Routing Problem with Drones (VRPD) adalah varian dari *Vehicle Routing Problem* dengan implementasi *drone* dalam operasi (Lee & Kitjacharoenchai, 2019). Masalah tersebut bertujuan untuk menemukan rangkaian rute yang optimal bagi armada kendaraan untuk melakukan perjalanan agar dapat dikirim ke *customer*. Dinamai VRPD karena memanfaatkan keunggulan kendaraan kecil (*drone*) dan besar (truk) dalam sistem pengiriman. Dalam taksonomi VRP, VRP paling dasar yang dipertimbangkan dalam literatur adalah yang disebut *Capacitated Vehicle Routing Problem* (CVRP). VRPD dianggap sebagai perpanjangan dari FSTSP serta mTSPD. Sementara *Flying Sidekick Travelling Salesman Problem* (FSTSD) mempertimbangkan satu *drone* dan satu truk di seluruh operasi, *Multiple Travelling Salesman Problem with Drone* (mTSPD) menggunakan beberapa *drone* dan beberapa truk untuk melakukan pengiriman. Dalam mTSPD mengasumsikan bahwa

truk dan *drone* keduanya memiliki kapasitas tak terbatas, yang mungkin tidak sesuai untuk diterapkan dalam kebanyakan kasus. Model VRPD yang diusulkan memperhitungkan kapasitas truk dan *drone* serta urutan operasi peluncuran dan pendaratan. Menurut Li et al. (2019), di luar konstrain yang terdapat di CVRP, berikut adalah aturan dalam VRPD.

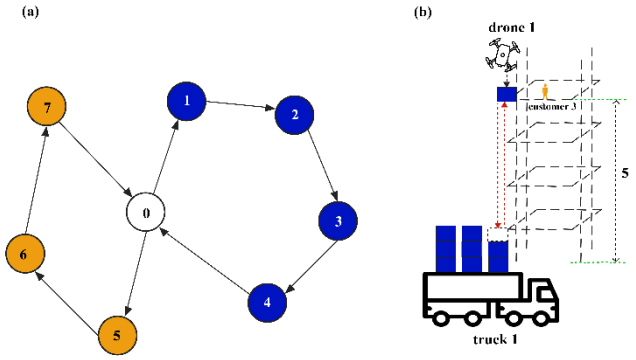
- Setiap pelanggan harus dilayani tepat satu kali baik oleh truk atau *drone* (pemenuhan *demand* tidak dapat dibagi seperti dalam CVRP)
- Waktu truk maupun *drone* di lokasi pelanggan harus disesuaikan agar sama
- Beban yang dapat dibawa oleh *drone* harus lebih kecil dari kapasitas *drone*
- *Drone* hanya dapat diluncurkan untuk perjalanan yang memiliki baterai yang cukup untuk menyelesaikan pengirimannya dan terbang kembali ke truk

VRP sendiri sudah menjadi *NP-Hard* dan dengan demikian memasukkan *drone* dalam operasi membuat masalahnya jauh lebih kompleks dan rumit karena harus memutuskan solusi kombinatorial dari rute truk serta rute *drone* sambil mempertahankan urutan operasi peluncuran dan pendaratan. Penerapan VRPD di industri berpotensi menghasilkan efisiensi biaya dan mengurangi total waktu pengiriman dari pengiriman *last-mile*.



Gambar 8 *Vehicle Routing Problem with Drones* (VRPD)

4. *Multi-Objective Vehicle Routing Problem with Time Window and Drones* (MO-VRPTW-D)



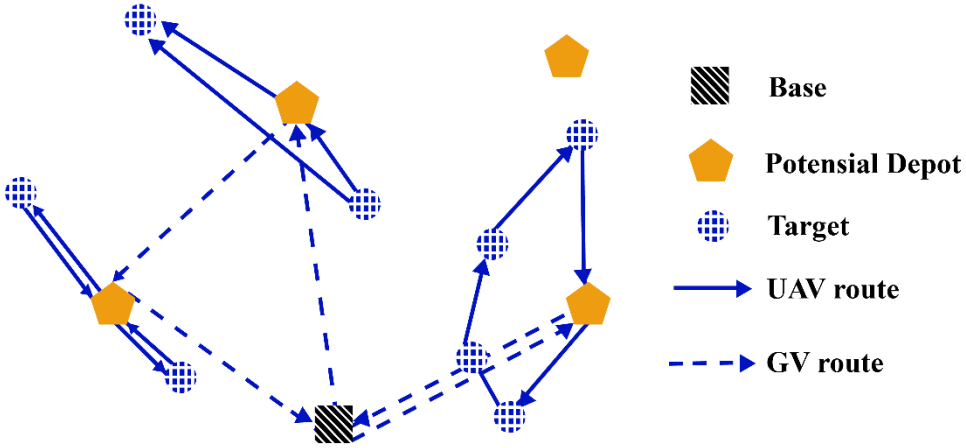
Gambar 9 *Multi-Objective Vehicle Routing Problem with Time Window and Drones* (MO-VRPTW-D)

Multi-Objective Vehicle Routing Problem with Time Window and Drones (MO-VRPTW-D) adalah pengembangan dari VRPD dimana terdapat konstrain *time window* serta multi-objektif. Fungsi objektifnya antara lain adalah minimasi konsumsi energi truk, konsumsi energi *drone*, dan jumlah truk. Model ini diusulkan oleh Yun-qi Han, Jun-qing Li, Zhengmin Liu, Chuang Liu, dan Jie Tian dalam *International Journal of Advanced Robotic Systems* dengan metode yang digunakan di dalamnya adalah *Genetic Algorithm* (GA), *Tabu Search* (TS), *Variable Neighborhood Search* (VNS), dan metode yang mereka usulkan yaitu *Improved Artificial Bee Colony* (IABC). Serupa dengan VRPD, MO-VRPTW-D juga mempertimbangkan faktor kapasitas angkut truk dan *drone* ditambah dengan faktor *time window* dari setiap *customer*. Namun model yang diusulkan pada Han, et al. (2020) sedikit berbeda dari konsep VRPD yang bisa dibilang rumit. MO-VRPTW-D sendiri lebih kepada pengembangan VRPTW dibantu dengan *drone* dalam proses mengantar barang ke *customer* (tanpa ada proses penghapusan rute truk oleh *drone* seperti pada VRPD). Berikut aturan dalam MO-VRPTW-D yang dijelaskan menurut Han, et al. (2020).

- Terdapat multi-objektif saling *trade-off* yang digabungkan dengan *Weighted-Sum Method* dimana objektif dengan bobot tertinggi adalah prioritas model

- Truk dan *drone* berangkat dan kembali ke depot yang sama dengan truk memuat barang dan *drone*
- Satu truk hanya membawa satu *drone*
- Muatan tidak melebihi kapasitas truk dan *drone*
- Tiap *customer* hanya dikunjungi oleh satu truk dan dilayani hanya sekali
- Dalam durasi pelayanan, waktu *take off* dan waktu pengisian daya diabaikan
- Waktu tempuh masing-masing truk tidak melebihi batas maksimum
- Waktu pelayanan sesuai dengan *time window*

2.3 Notasi pada MO-VRPTW-D



Gambar 10 Notasi pada MO-VRPTW-D

Pengiriman barang menggunakan *drone* yang memerlukan model matematis tentunya harus memperhatikan beberapa aturan seperti yang terlihat pada gambar. Untuk mendapatkan nilai optimal dari sebuah fungsi objektif, penting untuk memperhatikan aturan seperti *base* (depot), *parking lot*, dan target yang ketiganya tersambung dengan rute. Secara singkat, rute dibagi menjadi dua yaitu rute truk dan rute *drone* yang merupakan jalur yang dilewati oleh truk dan *drone*. Sedangkan base adalah titik berangkat dan pulang dari masing-masing truk yang membawa *drone*. Perbedaan antara *parking lot* dan target adalah *parking lot* sendiri merupakan tempat berhentinya truk untuk melepas *drone*, sedangkan target adalah para *customer*-nya. Secara lengkap, masing-masing notasi akan dijelaskan sebagai berikut.

2.3.1 Truk

Truk selain membawa muatan barang juga membawa *drone* di mana peraturannya adalah setiap truk dilengkapi dengan satu *drone*. Truk yang digunakan untuk mengangkut *drone* adalah truk van yang mungkin untuk saat ini jarang terlihat di Indonesia digunakan untuk mengangkut barang. Tidak ada spesifikasi khusus untuk truk yang harus dipergunakan, tetapi perusahaan *workhorse* memberikan referensi mengenai truk yang harus digunakan adalah truk yang bisa membuka otomatis pada kap bak belakang. Hal ini dinilai efisien untuk mengeluarkan barang secara cepat dan tidak ribet. Perusahaan ini juga menggunakan truk tenaga listrik yang dinilai lebih ramah lingkungan. Terlepas dari semua itu,

Gambar 11 Truk untuk *Drone Delivery*
Sumber: https://www.freightwaves.com/

2.3.2 Unmanned Aerial Vehicle (UAV)

Sebagian besar perusahaan atau bisnis yang memiliki beberapa urgensi akan menggunakan metode yang berbeda di bidang teknologi, terutama di bidang penerbangan. Sejak beberapa tahun lalu, industri penerbangan telah mentransformasi jenis bisnisnya dengan memperkenalkan *drone*. *Unmanned Aerial Vehicle* (UAV) atau yang sering disebut dengan *drone* adalah jenis teknologi penerbangan yang bekerja dengan sistem otonom atau dikendalikan dari jarak jauh. Peneliti masih mengembangkan *drone* jenis baru yang dapat digunakan lebih luas dari sebelumnya (Diego, 2017). *Drone* yang digunakan untuk mengantar barang biasa disebut dengan *delivery drone*. Karakteristik *delivery drone* sendiri adalah kapasitas angkut yang besar serta daya jelajah yang jauh. Hingga saat ini, regulasi khusus pengiriman menggunakan *drone* di Indonesia belum bisa dipastikan kapan waktu pastinya akan dikeluarkan. Padahal, jika itu benar-benar terjadi, diprediksi perusahaan logistik dan *e-commerce* di Indonesia

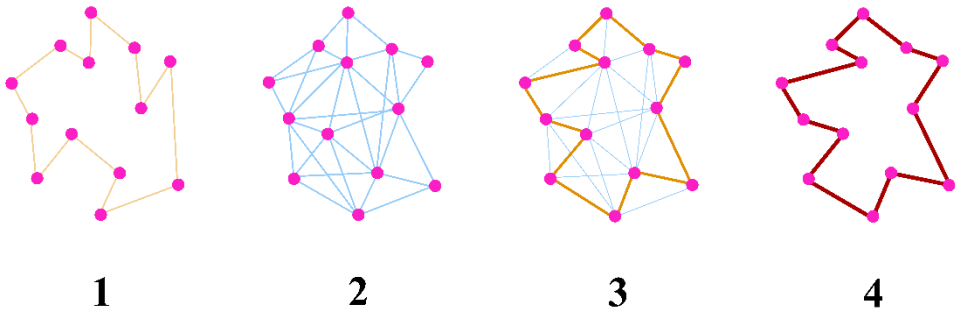
akan mendapatkan keuntungan lebih. Penggunaan *drone* sebagai pengiriman logistik yang digabungkan dengan ilmu sains dapat dipastikan mampu menekan biaya logistik.

Gambar 12 *Delivery Drone*

Sumber: https://www.rolandberger.com/

2.3.3 Rute

Rute adalah jalur yang harus ditempuh oleh kendaraan. Dalam hal ini kendaraan yang dimaksud adalah truk dan *drone*. Dalam kasus optimasi, rute menjadi *decision variabel* di mana dalam pemodelan matematis, apabila rute tersebut dilewati, maka bernilai 1, dan jika tidak akan bernilai 0 (biner). Hal ini yang menyebabkan permasalahan ini disebut dengan optimasi kombinatorial karena cara penyelesaian dari kasus ini hanyalah membolak-balik solusi (rute) untuk mendapatkan nilai yang optimal. Dalam kasus MO-VRPTW-D sendiri, rute dibagi menjadi dua yaitu rute untuk truk dari *base* atau depot menuju *parking lot* dan rute untuk *drone* dari *parking lot* menujut target atau *customer*.



Gambar 13 Rute

2.3.4 Base

Base atau depot merupakan titik awal keberangkatan dan titik pulang bagi seluruh kendaraan. Dalam kasus MO-VRPTW-D, hanya terdapat satu depot

(berbeda dengan kasus *Multi-Depot Vehicle Routing Problem*). Dalam hal ini, penelitian ini hanya berfokus pada pemberangkatan kendaraan dari satu titik saja. Tentu hal ini menjadi lebih *simple* dibandingkan dengan kasus multi-depot. Dalam matriks jarak, koordinat depot biasa ditulis dengan koordinat (x,y) yaitu (0,0) yang merupakan pusat koordinat kartesian. Dalam penerapan di perusahaan, *base* atau depot biasanya adalah sebuah *warehouse* pusat di mana kendaraan akan mengirim barang ke *warehouse* cabang, *retailer*, atau langsung ke *customer*.

2.3.5 Parking Lot

Parking lot berbeda dengan *base* atau depot di mana *parking lot* adalah titik berhentinya truk sebelum melepas *drone*. *Parking lot* bisa diibaratkan *base* atau depot bagi sebuah *drone* pengirim di mana *drone* saat berangkat dan kembali menuju truk setelah mengirim barang. Dalam perhitungan optimasi, *parking lot* tentu masuk ke dalam sebuah variabel. *Parking lot* turut diperhitungkan untuk menentukan di mana truk berhenti disebuah titik sehingga pelepasan *drone* menjadi optimal untuk seluruh titik pengiriman *customer*.

2.3.6 Target

Customer adalah titik akhir dari pengiriman barang di mana dalam penelitian ini perusahaan pengirim wajib mengantarkan barang yang berakhir di *customer* atau yang biasa disebut dengan *last-mile delivery*. Target yang dimaksud dalam hal ini adalah *customer* itu sendiri, di mana barang yang diantar ke *customer* diantar dengan *drone*, bukan dengan truk. Medan yang dituju dari truk menuju target tidak berpengaruh apabila menggunakan *drone*, karena *drone* sendiri merupakan kendaraan terbang. Berbeda jika yang mengantar barang ke customer adalah manusia atau truk yang melewati jalur darat, di mana setiap kendala pada jalan pasti berpengaruh terhadap kecepatan pengiriman.

2.4 *Push-Forward Insertion Heuristic*

Push-Forward Insertion Heuristic (PFIH), diperkenalkan oleh Solomon pada tahun 1987, adalah metode yang efisien untuk menghitung solusi *feasible* untuk kasus VRP dengan mengasumsikan jumlah kendaraan yang tak terbatas sebelum akhirnya dilanjutkan dengan metode *Simulated Annealing* dan *Ant Colony Optimization* untuk proses komputasinya. Metode PFIH cukup sering digunakan dalam routing problem untuk menentukan solusi awal dimana solusi

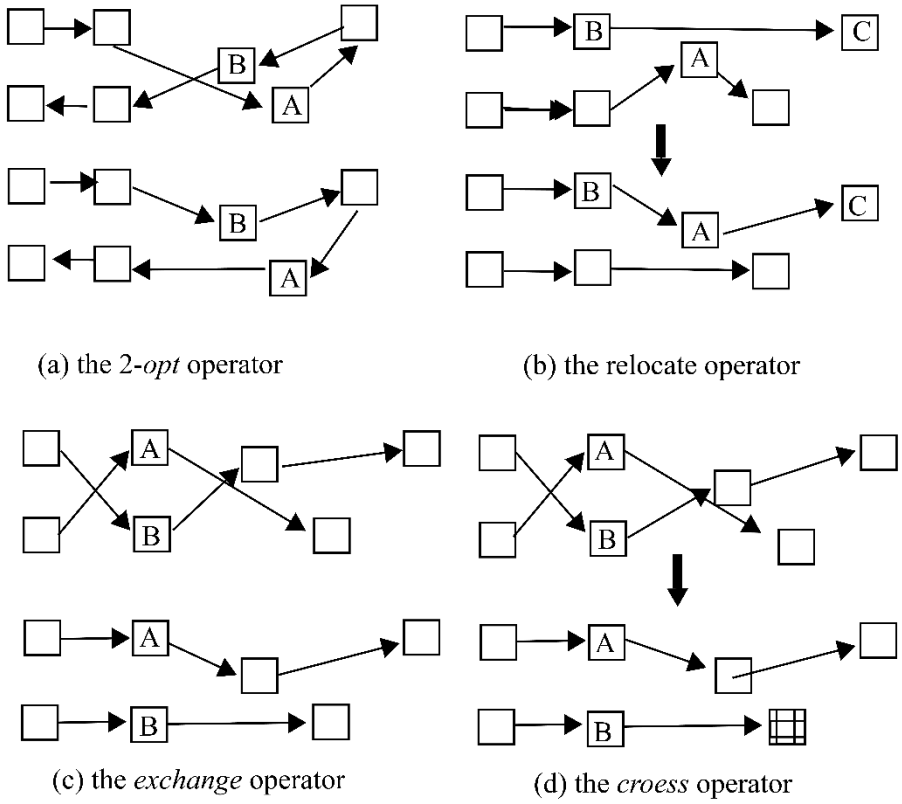
awal yang dibentuk tidak terlalu *random* karena PFIH sendiri adalah metode heuristik. Penerapan PFIH dalam penentuan solusi baru dapat mempercepat komputasi maupun memperpendek jarak menuju solusi optimal global. PFIH memulai rute baru dengan memilih pelanggan awal, biasanya terjauh dari depot pengiriman, dan kemudian secara iteratif memasukkan pelanggan yang belum di-*assign* ke dalam rute saat ini sampai kapasitas kendaraan saat ini terlampaui atau *waiting time* untuk setiap pelanggan baru yang ditambahkan akan melebihi konstrain terkait. Pada kondisi seperti ini, rute baru akan dibentuk dan proses ini berulang sampai semua pelanggan dilayani. Berikut adalah *pseudocode* untuk Algoritma PFIH.

Tabel 2 Algoritma *Push-Forward Insertion Heuristic*

	Algoritma 1: Algoritma *Push-Forward Insertion Heuristic*
	Input: *infeasible solution*
	Output: *feasible solution*
1	Bangkitkan r_0
2	$i=0$
3	$r_i = r_0$
4	**while** *kriteria penghentian belum dicapai* **do**
5	Pilih pelanggan terjauh dari depot dan masukkan ke rute saat ini r_i
6	**if** seluruh pelanggan terlayani, **then**
7	*go to* (20)
8	**else**
9	**if** kapasitas Q kendaraan k dalam rute r_i saat ini terlampaui, **then**
10	*go to* (19)
11	**else**
12	**foreach** pelanggan yang belum ter-*assign*
13	Temukan posisi terbaik untuk menyisipkan pelanggan ke dalam rute r_i (tanpa mempertimbangkan kapasitas Q kendaraan k), lalu bandingkan biaya untuk rute baru dengan posisi terbaik sebelumnya
14	**end**
15	**end**

16	**end**
17	Pilih pelanggan dengan perbedaan biaya terbesar, masukkan ke rute r_i
18	Perbarui kapasitas *Q* pada kendaraan *k* yang terlibat, *go to* (6)
19	Bangkitkan rute baru r_{i+1} dimulai dari depot; *i = i+1; go to* (5).
20	Kembali ke solusi saat ini
21	**end**

Input yang dibutuhkan adalan rute *random* (entah *feasible* ataupun *infeasible*). Apabila rute tersebut *feasible*, maka solusi tersebut yang dipakai untuk tahap selanjutnya. Namun apabila rute *infeasible*, tiap pelanggan akan di-*assign* pada posisi terbaiknya hingga mencapai *feasibility*.



Gambar 14 Visualisasi PFIH

2.5 *Simulated Annealing*

Simulated Annealing merupakan metode yang cukup sering digunakan dalam metaheuristik. Selain solusi yang dihasilkan cukup bagus baik itu untuk optimasi kontinu maupun diskret, metode SA tidak terlalu rumit digunakan dibandingkan metode metaheuristik lain (misalnya saja *Genetic Algorithm*). (Afifi, 2013). Diperkenalkan oleh Kirkpatrick, et al. (1983) yang terinspirasi dari

prosedur pendinginan (anil) sebuah logam atau baja, tujuan prosedur *annealing* mendeskripsikan penataan molekul partikel logam yang menghasilkan energi internal minimal pada logam atau baja dan mengacu pada pendinginan logam secara bertahap (iteratif) setelah mengalami panas tinggi. Secara umum, Algoritma SA mengadopsi gerakan iteratif sesuai dengan parameter suhu variabel yang meniru proses pendinginan logam atau baja.

Ide utama dari Algoritma *Simulated Annealing* (dalam kondisi tertentu) adalah untuk menerima solusi yang lebih buruk dengan harapan keluar dari optimum lokal saat ini (Afifi, 2013). Dalam pencarian solusi baru, terdapat dua kemungkinan yaitu mendapatkan solusi baru yang lebih baik dari solusi sebelumnya atau malah sebaliknya. Apabila menerima solusi yang lebih baik, secara otomatis solusi tersebut akan langsung diterima karena secara grafik, nilai *fitness* akan mengecil. Namun jika solusi yang diterima lebih buruk dari sebelumnya, berbeda dengan metode heuristik yang tidak menerima solusi yang lebih buruk (lebih rawan terjebak di optimum lokal), akan dibangkitkan bilangan random dengan distribusi tertentu dan apabila mencapai suatu nilai, maka ada kemungkinan untuk menerima solusi yang lebih buruk dengan harapan akan mendapatkan solusi yang lebih baik pada iterasi selanjutnya. Dalam SA, metode untuk pembangkitan bilangan *random* dan dibatasi oleh nilai tertentu dikenal dengan istilah kriteria Metropolis. Kriteria Metropolis adalah metode dalam *Simulated Annealing* yang digunakan untuk menentukan keputusan apakah solusi yang lebih buruk akan diterima atau tidak. Probabilitas menerima solusi yang baru dibuat dihitung sebagai $P(x) = e^{-\frac{\Delta f}{kT}}$, *d*i mana Δf adalah perbedaan nilai objektif (biasanya *fitness*) antara solusi baru dan solusi saat ini dan *T* adalah parameter yang disebut temperatur saat ini. Apabila dalam pembangkitan bilangan *random* (r) nilainya di bawah $P(x)$, maka solusi yang lebih buruk itu akan diterima, begitu juga sebaliknya. Proses iterasi akan terus berlanjut sampai selesai satu siklus dan setelah itu baru bisa dilakukan penurunan temperatur untuk siklus berikutnya. Parameter ini dikembangkan selama pencarian solusi dengan meniru proses pendinginan dalam metalurgi.

Pengembangan Algoritma *Simulated Annealing* menjadi sebuah metode metaheuristik tentu memerlukan beberapa proses iterasi yang memakan banyak

waktu terutama problem yang besar. Diperlukan *software* untuk memproses iterasi dengan cepat dan otomatis. Sebelum membuat *code* dalam sebuah *software*, diperlukan algoritma atau prosedur terlebih dahulu agar dalam proses Penulisan *code* mempunyai alur yang jelas dan valid. Berikut adalah Algoritma *Simulated Annealing* secara general menurut Santosa & Ai (2017).

Tabel 3 Algoritma *Simulated Annealing*

Algoritma 2: Algoritma *Simulated Annealing*

Input: T_0, c, it_{max}

Output: x_{opt}, f_{best}

Bangkitkan x_0
$f_{best} = f(x_0)$
$T = T_0$
$it = 1$
while *kriteria penghentian belum dicapai* **do**
  if $it < it_{max}$
    bangkitkan Δx berdistribusi Gaussian
    if $(f(x_0 + \Delta x) < f(x_0))$ **then**
      $f_{best} = f(x_0 + \Delta x)$
      $x_0 = x_0 + \Delta x$
    else
      $\Delta f = f(x_0 + \Delta x) - f(x_0)$
      bangkitkan bilangan random r
      if $r < exp(-\Delta f / kT)$ **then**
        $x_0 = x_0 + \Delta x$
      else
        $x_0 = x_0$
      end
    end
    $it = it + 1$
  end
  $T = T \cdot c$
end

Dalam input sudah jelas bahwa nilai yang dibutuhkan adalah temperatur awal (T_0), faktor pengurangan temperatur (c), dan jumlah iterasi maksimum yang diinginkan (it_{max}) dengan *output* berupa solusi (x_{opt}) dan nilai objektif atau *fitness* (f_{best}). Temperatur awal dibebaskan nilainya namun sebaiknya disesuaikan dengan jumlah iterasi maksimum. Semakin banyak jumlah iterasi yang diinginkan, seharusnya semakin tinggi pula nilai T_0. Nilai c antara 0 sampai 1, semakin besar nilai c, maka semakin perlahan pula logam atau baja menurun temperaturnya. Sehingga jika penurunan temperatur berjalan lambat, kemungkinan untuk mendapatkan solusi yang optimal akan semakin besar karena dengan penurunan temperatur yang lambat, proses iterasi tidak akan banyak melewatkan titik uji.

2.6 *Ant Colony Optimization*

Ant Colony Optimization (ACO) termasuk dalam kelompok *Swarm Intelligence*, merupakan salah satu jenis metode metaheuristik yang awalnya diciptakan untuk problem diskret dimana cara kerja dari ACO terinspirasi dari kawanan serangga (semut). ACO mengadopsi perilaku semut untuk mencari makanan dimana semut biasanya cenderung melalui rute terpendek dari rute-rute yang lebih dahulu dilalui sebelumnya. ACO biasanya digunakan untuk menyelesaikan *discrete optimization problems* dan persoalan yang kompleks dimana terdapat banyak variabel seperti *scheduling*, VRP, dll. Hasil yang diperoleh dengan menggunakan ACO, walaupun tidak optimal namun mendekati optimal.

Konsep yang ditiru ACO pada mulanya, semut berbondong-bondong pergi dari sarang menuju tempat makanan namun tidak semuanya melalui rute yang sama. Ketika semut kembali ke sarang, semut tersebut akan kembali ke tempat mencari makan untuk dibawa ke sarang lagi secara berulang. Mereka memanfaatkan feromon milik semut-semut sebelumnya dimana feromon ini akan di-*update* oleh semut baru sehingga semut-semut setelahnya akan lebih jelas mengikuti pola dari kawanannya. Feromon adalah zat kimia yang berasal dari kelenjar endokrin dan digunakan oleh makhluk hidup untuk mengenali sesama jenis, individu lain, kelompok, dan untuk membantu proses reproduksi. Berbeda dengan hormon, feromon hanya akan dikenali oleh kawanan sejenisnya saja. Feromon cenderung banyak ditemukan ketika rute tersebut banyak dilalui

oleh semut. Dan banyak semut cenderung melalui rute yang pendek untuk mencari makanan. Pada saat semut tidak melewati rute tersebut, tidak ada penambahan feromon melainkan feromon justru akan berkurang karena faktor penguapan. Rute dengan feromon tertinggi akan dipilih karena rute tersebut kemungkinan adalah rute terpendek diantara rute lainnya. Berikut adalah *pseudocode* untuk *Ant Colony Optimization* menurut Smarandache, et al. (2017).

Tabel 4 Algoritma *Ant Colony Optimization*

Algoritma 3: Algoritma *Ant Colony Optimization*

Input: N, ρ, d, it_{max}

Output: x_{opt}, f_{best}

Hitung $\eta(r,s)$ awal

Inisialisasi τ_0

$it = 1$

Probability of transition, $p_k(r,s) = \begin{cases} \frac{\tau(r,s)^{\alpha}\eta(r,s)^{\beta}}{\sum_{u \in M_k}\tau(r,u)^{\alpha}\eta(r,u)^{\beta}}, jika\ s \in M_k \\ 0, jika\ tidak \end{cases}$

while $it < it_{max}$

  foreach $n \in N$

    Buatlah *cummulative probability of transition*

    while $n \leq size(d)$

      Tentukan $r = rand$ untuk menentukan transisi

      Normalisasi matriks

      Perbarui *probability of transition*

      Perbarui *cummulative probability of transition*

      $n = n + 1$

    end

    $\tau_{ij} = (1-\rho)\tau_{ij} + \Delta\tau_{ij}$

    $it = it + 1$

  end

end

$x_{opt} = max(\tau_{ij})$

Dalam kasus *routing problem,* diperlukan data jarak sehingga fungsi objektif bisa dihitung. Dalam *pseudocode,* input yang dibutuhkan adalah N (jumlah semut), ρ (faktor penguapan feromon), d (jarak antar titik), dan it_{max}. Masing-masing node dihitung berapa peluang untuk dilewati semut. Semakin pendek jarak, semakin besar peluang dilewati oleh semut. Ketika satu siklus sudah terlewati, feromon untuk masing-masing node diperbarui (faktor penguapan dan penambahan). Data terbaru digunakan untuk melakukan perhitungan di iterasi berikutnya hingga iterasi maksimum dicapai.

2.7 Penelitian Terkait

Wen, Zhang, & Wong (2016) mempublikasikan artikel riset sebagai *improvement* di bidang transportasi khususnya bidang medis terutama dalam kondisi darurat. Dalam lingkungan medis, terutama dalam beberapa situasi darurat, UAV memainkan peran penting seperti pasokan obat-obatan dan darah dengan kecepatan dan efisiensi. Problem ini cukup kompleks karena mencakup pemeliharaan model suhu suplai darah selama transportasi, penjadwalan UAV, dan perencanaan rute jika ada banyak lokasi yang meminta darah, dan *resoure* yang terbatas. Dalam memodelkan perubahan pada temperatur darah, perlu dipelajari terlebih dahulu perubahan suhu darah karena lingkungan eksternal, agen pemanas (atau zat pendingin) dan faktor waktu selama transportasi, dan mencari solusi metode optimal untuk menghitung proporsi pencampuran darah dan pelengkap dalam keadaan dan kondisi pengiriman yang berbeda. Pada dasarnya, problem ini adalah CVRP (dengan mempertimbangkan faktor berat dan jarak) dengan kendaraan *drone* dengan *multiple-objective* (*bi-objective*). Objektif pertama dari problem ini adalah minimasi biaya pengiriman dan objektif yang kedua adalah pengoptimalan *schedule* untuk mendapatkan jumlah *drone* yang minimal. Dengan menggunakan metode *Decomposition-Based Multi-Objective Evolution Algorithm and Local Search with Random Nearest Neighbor* (MOEA/D-N) untuk menyelesaikan kasus UAV *Vehicle Routing Problem* (UVRP), metode ini terbukti lebih efisien dibanding dengan metode konvensional.

Wang & Sheu (2019) menjelaskan bahwa VRPD (*Vehicle Routing Problem with Drone*) merupakan varian dari VRP dimana dalam kasus VRPD adalah optimasi perutean dengan kombinasi kendaraan truk dan *drone* dalam

sebuah makalah. Salah satu ciri khas VRPD adalah bahwa *drone* dapat melakukan perjalanan dengan truk, lepas landas dari berhenti untuk melayani pelanggan, dan mendarat untuk kembali ke truk selama konstrain jangkauan terbang dan kapasitas. Perutean truk dan *drone* secara terintegrasi membuat masalah jauh lebih kompleks dan berbeda dari perutean kendaraan konvensional. Dalam *paper* ini, metode yang digunakan yaitu metode eksak dengan pemodelan MILP. Pemodelan dengan metode eksak tentunya mampu memperoleh optimum global namun tidak bisa menyelesaikan kasus yang sangat besar karena membutuhkan waktu komputasi yang sangat lama. Dalam *paper* juga terdapat informasi dari *paper* lain yang membahas perutean dengan truk dan drone yang telah dirangkum dalam tabel berikut.

Tabel 5 Penelitian Terkait Perutean dengan Truk dan *Drone*

Author	***Problem***	**Metode**	***Drone capacity***	***Drone:* truk**	**Objektif (min.)**	**Node**
Murray & Chu (2015)	1 truk; 1 *drone*	Heuristik	1-*customer*	1:1	Waktu kembali	~20
Wang, et al. (2017)	M truk; N *drone*	*Worst-case analysis*	1-*customer*	M:1	Waktu	-
Ponza (2016)	1 truk; 1 *drone*	SA	1-*customer*	1:1	Waktu	~200
Carlsson & Song (2017)	1 truk; 1 *drone*	Heuristik	1-*customer*	1:1	Waktu perjalanan	~100
Agatz, et al. (2018)	1 truk; 1 *drone*	Heuristik	1-*customer*	1:1	Biaya logistik	10
Ha, et al. (2018)	1 truk; 1 *drone*	Heuristik	1-*customer*	1:1	Biaya operasional	~100
Ham (2018)	M truk; N *drone*	Heuristik	M-*customer*	-	Waktu maksimum	~100
Chang & Lee (2018)	1 truk; M *drone*	Heuristik	1-*customer*	M:1	Waktu total	~100

Author	***Problem***	**Metode**	***Drone capacity***	***Drone:* truk**	**Objektif (min.)**	**Node**
Whang & Sheu (2019)	M truk; N *drone*	Eksak	M-*customer*	M:N	Biaya logistik	15

Penelitian oleh Soenandi, et al. (2019) pada Jurnal Sistem dan Manajemen Industri dengan Menggunakan *Ant Colony Optimization* dengan implementasi pada perusahaan otomotif terkait. Sebagai upaya peningkatan profit, penting bagi perusahaan manufaktur melakukan minimasi biaya logistik dan transportasi. Salah satu cara yang bisa dilakukan adalah dengan melakukan perhitungan matematis pada pengiriman barang yang bisa memberikan solusi optimal. Pengoptimalan rute dengan tujuan untuk meminimalkan total biaya transportasi merupakan permasalahan yang sering ditemukan pada *Vehicle Routing Problem* (VRP). *Capacitated Vehicle Routing Problem with Time Windows* (CVRPTW) merupakan salah satu varian dari VRP yang mempertimbangkan kapasitas kendaraan dan jangka waktu pelayanan tiap kendaraan. Seperti VRP pada umumnya, CVRPTW juga merupakan permasalahan *Non-Polynomial Hard* (NP-Hard) yang memerlukan algoritma yang efisien dan efektif dalam menyelesaikan permasalahan yang terjadi di perusahaan otomotif. Hasil *running* membuktikan bahwa metode ACO lebih baik daripada hasil eksisting. Sebagai kesimpulan, penelitian berhasil mengurangi jarak tempuh pada kendaraan sebesar 36 km per harinya.

Kitjacharoenchai & Lee (2019) mempelajari problem yang timbul dari perusahaan *e-commerce* dan ritel yang mencari cara untuk memangkas waktu dan biaya pengiriman dengan menjajaki peluang menggunakan *drone* untuk melakukan pengiriman jarak jauh. Dalam beberapa tahun terakhir, perutean dan penjadwalan *drone* telah menjadi bidang penelitian yang sangat aktif. Penelitian ini membahas konsep pengiriman kombinasi truk-*drone* bersama dengan gagasan memungkinkan *drone* terbang dari truk pengiriman, melakukan pengiriman, dan terbang ke truk pengiriman terdekat. Model yang diusulkan mempertimbangkan model perutean yang disinkronkan dengan memungkinkan beberapa *drone* terbang dari truk, melayani pelanggan, dan segera kembali ke truk yang sama untuk pertukaran baterai serta melakukan pengambilan paket. Model tersebut juga

memperhitungkan kapasitas truk dan *drone* untuk memastikan bahwa jumlah muatan yang dibawa oleh setiap *drone* tidak boleh melebihi kapasitasnya dan jumlah total muatan di setiap rute pengiriman harus kurang dari kapasitas truk. Tujuannya adalah untuk menemukan rute optimal truk dan *drone* yang meminimalkan total waktu kedatangan truk dan *drone* di depot setelah menyelesaikan pengiriman. Masalah tersebut dapat diselesaikan dengan formulasi *Mixed Integer Programming* (MIP) untuk masalah ukuran kecil. Dalam *paper* juga disajikan beberapa skenario untuk membandingkan sensitivitas dari problem ini. Kedepannya, Penulis berencana mengembangkan metode heuristik dan metaheuristik untuk memperoleh solusi yang bagus dengan waktu komputasi yang rasional.

Han, Li, Liu, Liu, & Tian (2020) memperkenalkan *improve* dari metode yang sudah ada untuk kasus MO-VRPTW-D. Dalam *paper* dijelaskan secara detail mengenai metode dan model yang disampaikan. Dalam *paper* ini mengadopsi metode *Artificial Bee Colony* yang telah di-*improve* dan dinamai oleh mereka dengan sebutan *Improved Artificial Bee Colony* (IABC) dimana metode ini adalah gabungan dari metode PFIH, *Local Search*, dan metode ABC itu sendiri. Objektif dalam *paper* bersifat *multiple* diantaranya yaitu minimasi energi total truk, energi total *drone*, dan jumlah total truk. MO-VRPTW-D merupakan varian dari VRP dimana problem ini menambahkan konstrain multi-objektif, kapasitas dan *demand*, time window, serta kendaraan *drone* untuk mengunjungi titik tertentu. Masalah ini menjadi semakin kompleks ketika mengadopsi dua jenis *demand* yang diangkut oleh dua jenis truk dengan kapasitas angkut berbeda. Di akhir bagian, Penulis juga membandingkan solusi dari metode IABC, GA, VNS, dan TS dimana IABC unggul 70% dari seluruh percobaan (55 percobaan).

Penelitian oleh Redi, et al. (2020) ini bertujuan untuk mencari satu set rute kendaraan dengan total waktu transportasi minimum untuk distribusi farmasi di PT. XYZ di Jakarta Barat. Masalah tersebut dimodelkan sebagai *Capacitated Vehicle Routing Problem* (CVRP). CVRP dikenal sebagai masalah *NP-Hard*. Oleh karena itu, diperlukan metode metaheuristik dan dalam penelitian ini digunakan *Simulated Annealing*. Metode SA dipilih karena cukup bagus untuk menyelesaikan problem optimasi diskret. Hasil penelitian menunjukkan bahwa

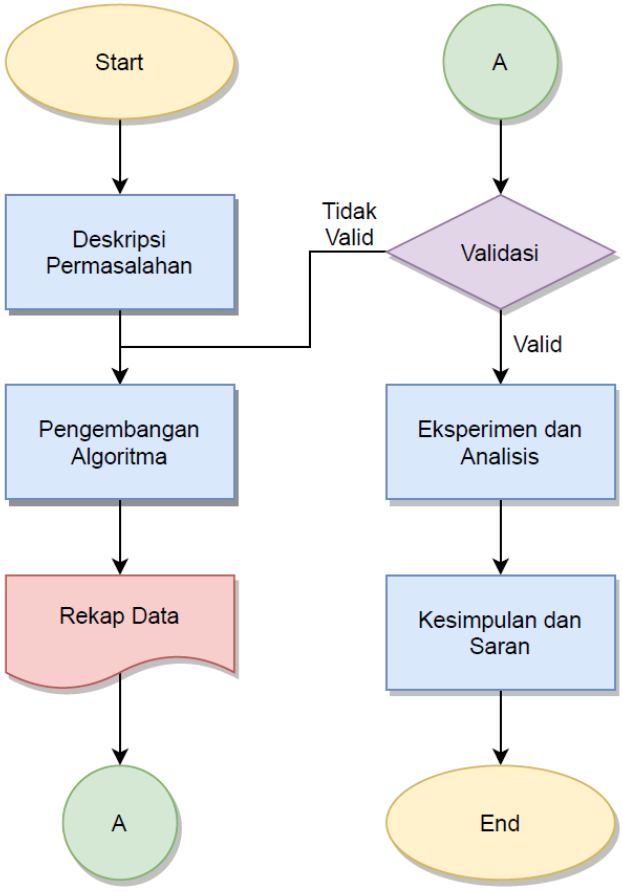
Simulated Annealing dan *Nearest Neighboor Algorithm* bekerja baik berdasarkan persentase perbedaan antara masing-masing algoritma dengan solusi optimal masing-masing 0,03% dan 5,50%. Dengan demikian, Algoritma *Simulated Annealing* memberikan hasil yang lebih baik dibandingkan dengan Algoritma *Nearest Neighboor*. Selanjutnya, Algoritma *Simulated Annealing* yang diusulkan dapat menemukan solusi yang sama terhadap metode eksak dengan cukup konsisten. Penelitian ini telah menunjukkan bahwa Algoritma *Simulated Annealing* memberikan kualitas solusi yang sangat baik untuk masalah tersebut.

Amico, Montemanni, & Novellani (2021) mengembangkan permasalahan ditulis oleh Murray & Chu (2015) dimana topik dalam penelitian ini memiliki kesamaan yaitu membahas *Flying Sidekick Traveling Salesman Problem* (FSTSP) yang dimuat dalam *International Transactions*. Penelitian ini menyajikan tiga formulasi yang telah disempurnakan untuk masalah *Flying Sidekick Traveling Salesman*, di mana truk dan *drone* bekerja sama untuk mengirimkan paket kepada pelanggan dengan meminimalkan waktu penyelesaian. *Drone* dapat berangkat dan harus kembali ke truk setelah mengunjungi satu pelanggan selama penerbangan tidak melebihi daya tahan baterainya, sementara truk dapat melayani pelanggan lain. Formulasi baru memungkinkan untuk mengurangi jumlah konstrain "*big-M*" menganut model yang sebelumnya sudah ada serta dapat mendapatkan hasil yang lebih baik dibanding sebelumnya dengan memecah beberapa contoh *benchmark* yang solusi optimalnya sebelumnya tidak diketahui. Penelitian ini juga menunjukkan bagaimana memodifikasi model baru untuk memasukkan beberapa varian masalah dari literatur. Penelitian ini bisa dibilang merupakan *improvement* dari karya Murray & Chu (2015) karena fungsi objektif dan konstrain yang telah dimodifikasi. Terbukti dari hasil *running software*, untuk 10 – 20 customer, metode yang diusulkan mampu menyelesaikan lebih baik dibanding model Murray & Chu (2015).

BAB III
METODOLOGI PENELITIAN

Pada bab ini akan dibahas mengenai Metode Pengerjaan Tugas Akhir dalam bentuk *flowchart*, Deskripsi Permasalahan VRPTW dan MO-VRPTW-D, Pengembangan Algoritma, Rekap Data untuk pengujian, Validasi dari algoritma terkait, rencana Eksperimen Algoritma, serta Pengambilan Kesimpulan dari eksperimen algoritma.

3.1 Metode Pengerjaan Tugas Akhir



Gambar 15 Metode Pengerjaan Tugas Akhir

Pada Gambar 15 disajikan metode atau prosedur yang akan dilakukan di Penelitian Tugas Akhir. Pada tahap pertama dilakukan pendeskripsian masalah dimana tahap ini menjadi dasar validasi menggunakan metode eksak. Pengembangan algoritma dimulai dengan membangun *pseudocode* dan *code* akan dilampirkan pada halaman akhir. Selanjutnya akan dilakukan rekap data sekunder dari *paper* lain. Pada tahap selanjutnya akan dilakukan proses validasi untuk menentukan apakah algoritma dapat dipakai dalam tahap eksperimen, dan apabila

tidak valid, proses akan diulang ke tahap pengembangan algoritma. Algoritma yang valid akan digunakan untuk menguji data dengan beberapa skenario dengan tujuan memperoleh hasil yang mendekati optimal global. Dari hasil eksperimen tersebut akan dianalisis yang kemudian akan diambil kesimpulan.

3.2 Deskripsi Permasalahan

VRP dalam optimasi tergolong dalam optimasi kombinatorial yang cukup kompleks (*NP-Hard*) yang membutuhkan pemodelan rumit agar kendala dalam kasus tersebut terpenuhi. Salah satu bahasa yang digunakan untuk menjelaskan kasus VRP adalah dengan menggunakan model matematis. Model matematis adalah deskripsi dari suatu sistem dengan menggunakan konsep dan bahasa matematika. Proses pengembangan model matematika disebut pemodelan matematika. Model matematis banyak digunakan dalam dunia teknik khususnya optimasi yang menggunakan konstrain atau batasan dalam pencarian solusinya. Dalam subbab ini akan dibahas mengenai notasi serta model matematis yang akan digunakan untuk menentukan rute VRPTW dan MO-VRPTWD.

3.2.1 Deskripsi, Notasi, dan Model Matematis untuk VRPTW

Jika dilihat, VRP saja sudah rumit, VRPTW akan lebih rumit dibanding VRP karena terdapat konstrain *time window*. Salah satu yang terpenting dalam VRPTW adalah terpilihnya solusi yang *feasible* dengan konstrain *time window*. Apabila solusi awal yang ditetapkan telah *feasible*, dalam iterasi berikutnya akan lebih mudah mencari rute yang lebih baik. Dalam kasus *real*, tidak hanya konstrain *time window* saja yang dibutuhkan dalam perhitungan, namun juga melihat konstrain *demand* dan kapasitas kendaraan dimana jumlah barang yang diangkut tidak boleh melebihi kapasitas angkut. Sehingga dalam pemodelan ini diperlukan penggabungan antara VRPTW dengan kapasitas konstrain. Model VRPTW ini nantinya akan menjadi dasar Model MO-VRPTW-D dengan merubah konstrain sesuai dengan algoritma MO-VRPTW-D. Berikut merupakan notasi dan model matematis untuk VRPTW dengan konstrain kapasitas menurut Kallehauge, et al. (2016) yang telah dimodifikasi oleh Penulis.

Notasi:

- $N = \{1, 2, 3, \ldots, n+1\}$, set node, untuk $n = 1$ dan $n + 1$ adalah depot
- $C = \{2, 3, \ldots, n\}$, set kota

- $V = \{1, 2, 3, \ldots, v\}$, set truk
- $|V| = v$, jumlah truk tersedia
- t_{ij} adalah waktu perjalanan dari node-i ke node-j
- D_i adalah *demand* pada node-i
- Q adalah kapasitas truk
- $a_i \; dan \; b_i$ masing-masing adalah *time window* untuk node-i
- $z_{ik} \; dan \; z_{jk}$ masing-masing adalah waktu tiba truk-k pada node-i dan node-j
- w_{ik} adalah adalah *waiting time* truk-k pada node-i
- s_i adalah *serving time* pada node-i
- $x_{ijk} = \{0,1\}$, *decision variable* apabila truk-k melewati node-i ke node-j maka bernilai 1, bernilai 0 apabila tidak

Objective function:

$$min\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{v} t_{ij}x_{ijk}\right)$$

dengan:

$i \neq j$

$i, j \in N$

$k \in V$ (3.1)

Subject to:

$x_{iik} = 0$ $\forall i \in N, \forall k \in V$ (3.2)

$x_{n+1,j,k} = 0$ $\forall j \in N, \forall k \in V$ (3.3)

$$\sum_{j=2}^{c}\sum_{k=1}^{v} x_{1jk} \leq |V|$$ (3.4)

$$\sum_{j=1}^{n}\sum_{k=1}^{v} x_{ijk} = 1$$ $\forall i \in C$ (3.5)

$$\sum_{i=i}^{n} D_i \sum_{j=1}^{n} x_{ijk} \leq Q$$ $\forall k \in V$ (3.6)

$$\sum_{j=1}^{n} x_{1jk} = 1$$ $\forall k \in V$ (3.7)

$$\sum_{i=1}^{n} x_{ilk} - \sum_{j=1}^{n} x_{ljk} = 0 \qquad \forall l \in C, \forall k \in V \qquad (3.8)$$

$$\sum_{i=1}^{n} x_{i,n+1,k} = 1 \qquad \forall k \in V \qquad (3.9)$$

$$x_{ijk}(z_{ik} + w_{ik} + s_i + t_{ij} - z_{jk}) = 0 \qquad \forall i, j \in N, \forall k \in K \qquad (3.10)$$

$$a_i \leq z_{ik} + w_{ik} \leq b_i \qquad \forall i \in N, \forall k \in K \qquad (3.11)$$

$$a_i \leq z_{ik} + w_{ik} + s_i \leq b_i \qquad \forall i \in N, \forall k \in K \qquad (3.12)$$

$$x_{ijk} \in \{0,1\} \qquad \forall i, j \in N, \forall k \in K \qquad (3.13)$$

Decision variables:

$$x_{ijk} = \begin{cases} 1, jika\ truk - k\ melewati\ node - i\ ke\ node - j \\ 0, jika\ tidak \end{cases} \qquad (3.14)$$

Fungsi objektif (3.1) menyatakan waktu tempuh total kendaraan. Persamaan (3.2) melarang kendaraan dari node-i untuk kembali ke node-i lagi. Persamaan (3.3) memastikan bahwa titik terakhir adalah *n+1* (depot). Persamaan (3.4) menunjukkan total kendaraan yang digunakan harus kurang dari total kendaraan tersedia (*V*). Persamaan (3.5) konstrain yang mengharuskan setiap pelanggan dilayani oleh tepat satu kendaraan. Persamaan (3.6) adalah batasan kapasitas kendaraan tidak lebih dari *demand*. Persamaan (3.7) – (3.9) adalah gambaran dari kendaraan yang keluar dari node-*i* menuju node-*j*, kendaraan yang melewati, dan kendaraan yang masuk node-*i* sehingga persamaan ini mengharuskan model membentuk rute. Sedangkan persamaan (3.10) – (3.12) digunakan untuk memodelkan batasan *time window*. Untuk persamaan (3.13) menunjukkan bahwa nilai x_{ijk} adalah biner. Pada kasus ini, *decision variables* (3.14) yang dibutuhkan adalah x_{ijk} jika bernilai 1 dan 0 dengan keterangan seperti persamaan di atas.

3.2.2 Deskripsi, Notasi, dan Model Matematis untuk MO-VRPTW-D

Dalam VRPTW yang dibahas di subbab sebelumnya hanya menggunakan truk saja, berbeda dengan MO-VRPTW-D yang terdapat *drone* untuk mengantarkan pesanan ke beberapa *customer* yang dipilih. Sehingga terdapat perbedaan model matematis misalnya *serving time* yang sebelumnya adalah konstanta, dalam kasus ini akan menjadi variabel, serta masih banyak perubahan lain yang akan dijelaskan pada bagian selanjutnya. Apabila dilihat

secara logika, MO-VRPTW-D akan lebih rumit dibandingkan dengan VRPTW. Hal ini terbukti pada konstrain yang tersedia akan semakin banyak mengingat kombinasi solusi yang tersedia semakin banyak. Berikut adalah rincian proses pembuatan model MO-VRPTW-D yang didapat dari model VRPTW.

- Menambahkan fungi objektif yang lebih kompleks
- Menambahkan *decision variables* y_{ik}
- Menambah variasi set kendaraan dimana set kendaraan tidak lagi identik sehingga pemilihan set kendaraan juga menjadi *decision* dalam hal ini
- Merubah *travel time* dan *serving time* tidak lagi menjadi konstanta, namun menjadi variabel yang masuk ke dalam konstrain, dimana hal ini diakibatkan oleh set kendaraan yang tidak identik
- Menambahkan batasan maksimum *total travel time* pada masing-masing jenis truk

Pada kasus MO-VRPTW-D pada Han, et al. (2020), penyelesaiannya menggunakan *Weighted-Sum Method* dimana semua objektif akan digabung dengan memberi bobot sesuai dengan prioritas masing-masing objektif untuk. Tujuan pemberian bobot adalah menitikberatkan kasus terhadap topik tujuan utama yaitu mengurangi biaya logistik secara signifikan yang timbul akibat kendaraan logistik saat ini (truk) memiliki biaya operasional mahal. Pada kasus MO-VRPTW-D dimana memiliki tujuan utama yaitu meminimasi biaya konsumsi energi truk (seperti dijelaskan sebelumnya) yang sampai saat ini bisa dibilang mahal dan mengakibatkan membengkaknya biaya logistik, juga akan berdampak pada biaya lainnya. Misalnya saja dengan hanya mempertimbangkan hanya biaya konsumsi energi truk, akan menimbulkan biaya yang mahal pula pada *setup cost* set kendaraan seperti pada penjelasan berikut.

Objective function:

$$min\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{v} x_{ijk} w_k^t t_{ijk}\right)$$

dengan:

$i \neq j$

$i, j \in N$

$k \in V$ (3.15)

Objektif yang pertama yaitu meminimasi konsumsi energi truk. Karena hal ini menjadi prioritas utama dalam MO-VRPTW-D, objektif ini perlu diberi pertimbangan utama diantara objektif lainnya. Objektif ini sendiri mengasumsikan bahwa konsumsi energi truk tidak terhitung ketika truk berhenti (hanya terhitung ketika melakukan perjalanan). Pada Persamaan (3.15) tertulis bahwa besarnya besar konsumsi energi truk adalah perkalian antara konsumsi energi truk per menit dengan lamanya total waktu perjalanan (dalam menit). Untuk konsumsi energi truk dan *travel time* sendiri bergantung pada jenis truk yang digunakan dan kedua variabel ini memiliki *trade-off* dengan *setup* set kendaraan yang akan dijelaskan pada bagian selanjutnya.

Objective function:

$$min\left(\sum_{i=2}^{n}\sum_{k=1}^{v} w_k^d s_{ik}\right)$$

dengan:

$i \in N$

$k \in V$ (3.16)

Persamaan di atas adalah objektif untuk meminimasi konsumsi energi *drone*. Konsumsi energi *drone* terhitung pada saat *drone* lepas landas dari truk hingga *drone* selesai melayani *customer*. Persamaan (3.16) menunjukkan konsumsi energi *drone* adalah hasil dari perkalian antara konsumsi energi *drone* per menit dengan total lama waktu pelayanan. *Drone* yang dipilih untuk setiap *customer* adalah *drone* yang *feasible* secara *time window*. Untuk itu, pemilihan *drone* dalam konstrain MO-VRPTW-D akan menimbulkan konsumsi energi yang berbeda. Model harus memilih minimal spesifikasi *drone* yang *feasible* melayani *customer* namun risikonya adalah konsumsi energi yang lebih besar.

Objective function:

$$min\left(\sum_{j=2}^{n}\sum_{k=1}^{v} x_{1jk}\right)$$

dengan:

$j \in N$

$k \in V$ (3.17)

Objektif yang terakhir adalah meminimasi jumlah set kendaraan digunakan. Semakin besar kapasitas dan waktu tempuh maksimal pada truk yang dipilih, maka semakin dihindari. Model akan secara otomatis memilih set kendaraan 1, dan apabila tidak *feasible*, set kendaraan selanjutnya akan digunakan. Set kendaraan 1 memiliki spesifikasi terendah dibanding lainnya (spesifikasi tertinggi adalah set kendaraan 100), namun pemilihan akan dimulai dari set kendaran terkecil. Hal ini *trade-off* terhadap kecepatan truk yang semakin lambat, sehingga *travel time* yang dihasilkan antar node akan semakin besar yang mempengaruhi ketepatan waktu truk dalam mendatangi *customer*. Misal saja terdapat dua pilihan memilih 2 truk kecil atau 1 truk besar untuk memuat barang menuju *customer*. Tentu konsumsi energi yang dihasilkan akan lebih kecil apabila menggunakan 2 truk kecil, namun objektif yang ditimbulkan akan lebih besar. Sehingga pada kasus optimasi seperti ini tentu akan lebih memilih menggunakan 1 truk besar.

Dalam perhitungan perbandingan metode, akan digunakan data R101.6 yang belum dimodifikasi. Dalam perhitungan juga disertakan 5 set kendaraan (set kendaraan 1 – 5) yang nantinya akan menghasilkan solusi optimal yang berbeda untuk setiap *case*. Selain itu, disajikan rekap hasil perhitungan dari masing-masing skenario yang nantinya akan dipilih skenario terbaik pada proses selanjutnya. Berikut adalah analisis perbandingan metode yang akan dipilih untuk menyelesaikan kasus MO-VRPTW-D.

1. Metode *Weighted-Sum*

   Pada tahap pertama, seluruh fungsi objektif harus disamakan terlebih dahulu satuannya. Pada tahap ini, Penulis menyamakan seluruh satuan dari fungsi objektif menggunakan satuan biaya (euro). Pada tahap pengujian awal, seluruh fungsi objektif akan diberi bobot sama besar yaitu 0,3333 atau bisa ditulis tanpa bobot. Untuk fungsi objektif 1 (konsumsi energi truk), nilai w_k^t akan dikalikan dengan €0,1457/liter yang setara dengan harga bahan bakar. Untuk fungsi objektif 2 (konsumsi energi *drone*), nilai w_d^t akan dikalikan dengan €0,0142/kWh yang setara dengan harga listrik. Sedangkan untuk fungsi objektif 3 (jumlah set kendaraan), perlu dikalikan dengan nilai €183,8 – €208,4 untuk set kendaraan 1 – 5.

Berikut adalah fungsi objektif hasil normalisasi untuk Metode *Weighted-Sum.*

Multi-objective function:

$$min\left[\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{v} x_{ijk}w_k^t t_{ijk}\right)+\left(\sum_{i=2}^{n}\sum_{k=1}^{v} w_k^d s_{ik}\right)+\left(\sum_{j=2}^{n}\sum_{k=1}^{v} \delta_k x_{1jk}\right)\right]$$

dengan:

$i \neq j$

$i, j \in N$

$k \in V$ (3.18)

2. Metode *Lexicographic*

Pada tahap pertama, akan didefinisikan terlebih dahulu nilai maksimal yang mungkin untuk setiap fungsi objektif. Proses perhitungan terdapat 3 skenario diantaranya skenario 1 (fungsi objektif 1), skenario 2 (fungsi objektif 2), dan skenario 3 (fungsi objektif 3). Skenario 1 menggunakan fungi objektif 1 (minimasi konsumsi energi truk) dimana fungsi objektif 2 dan 3 akan dimasukkan ke dalam konstrain dan begitu juga untuk skenario 2 dan 3. Berikut penjelasan lengkap dari masing-masing skenario.

Subject to:

$$\sum_{i=2}^{n}\sum_{k=1}^{v} w_k^d s_{ik} \geq 1{,}11 \qquad \forall i \in N, \forall k \in K \qquad (3.19)$$

Meletakkan fungsi objektif 2 ke dalam konstrain termasuk ke dalam skenario 1 dan 3. Konsumsi energi *drone* terbanyak akan diperboleh apabila menggunakan set kendaraan 5. Maka dari itu nilai RHS pada konstrain (3.19) akan di-set dimana seluruh rute menggunakan set kendaraan 5 dengan nilai konsumsi energi *drone* minimal yang mungkin adalah sebesar 1,11 kWh.

Subject to:

$$\sum_{j=2}^{n}\sum_{k=1}^{v} x_{1jk} \geq 1 \qquad \forall j \in N, \forall k \in K \qquad (3.20)$$

Fungsi objektif 3 sebagai konstrain terdapat di skenario 1 dan 2 dimana jumlah set kendaraan dibatasi. Nilai RHS sama dengan 5 yang menyatakan dalam kasus ini hanya diperbolehkan menggunakan minimal 1 set kendaraan.

Subject to:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{v} x_{ijk} w_k^t t_{ijk} \geq 85{,}43 \qquad \forall i,j \in N, \forall k \in K \qquad (3.21)$$

Fungsi objektif 1 akan diletakkan sebagai konstrain pada skenario 2 dan 3. Konsumsi energi truk terbesar akan diperboleh ketika menggunakan set kendaraan 5. Hal tersebut dikarenakan konsumsi energi per satuan waktu dari set kendaraan 5 adalah yang terbesar, serta set kendaraan 5 juga memiliki kecepatan paling lambat sehingga nilai *travel time* antar node juga akan semakin besar. Nilai RHS pada konstrain (3.21) yang setara dengan konsumsi energi truk minimal yang memungkinkan sama dengan 85,43 liter.

3. Rekap Perbandingan

Berikut adalah rekap hasil perbandingan antara *Weighted-Sum Method* dan *Lexicographic Method* pada masing-masing fungsi objektif.

Tabel 6 Rekap Perbandingan Metode *Weighted-Sum* dan *Lexicographic*

Weighted-Sum Method			
Total	$min(Z_1 + Z_2 + Z_3)$	$Z_1 = 85{,}45$ $Z_2 = 1{,}153$ $Z_3 = 1$	$*Z = 196{,}37$
Lexicographic Method			
Energi Truk	$min(Z_1)$	$Z_1 = 85{,}43$ $Z_2 = 1{,}4228$ $Z_3 = 1$	$Z = 202{,}47$
Energi *Drone*	$min(Z_2)$	$Z_1 = 109{,}84$ $Z_2 = 1{,}11$ $Z_3 = 1$	$Z = 199{,}82$
#Set Kendaraan	$min(Z_3)$	$Z_1 = 115{,}05$ $Z_2 = 1{,}64$	$Z = 212{,}86$

Lexicographic Method			
		$Z_3 = 1$ (3)	

*****: nilai terkecil

Berdasarkan penjelasan sebelumnya, dapat diambil kesimpulan bahwa metode terbaik untuk menyelesaikan kasus optimasi multi-objektif pada kasus MO-VRPTW-D adalah *Weighted-Sum Method. Weighted-Sum Method* yang digunakan Penulis akan memakai *weight coefficient* untuk masing-masing objektif yang telah dinormalisasi dengan masing-masing untuk konsumsi energi truk (α) sebesar 0,6 sebagai objektif dengan prioritas utama, konsumsi energi *drone* (β) sebesar 0,1 yang menunjukkan objektif tidak terlalu diprioritaskan, serta set kendaraan (γ) sebesar 0,3 untuk mengimbangi konsumsi energi truk yang muncul. Pada *Weighted-Sum Method,* semua objektif akan disamakan satuannya yaitu dalam satuan biaya. Berikut merupakan notasi dan model matematis untuk MO-VRPTW-D menurut Han, et al. (2020) yang telah dimodifikasi oleh Penulis.

Notasi:

- $N = \{1, 2, 3, \ldots, n+1\}$, set node, untuk $n = 1$ dan $n + 1$ adalah depot
- $C = \{2, 3, \ldots, n\}$, set kota
- $V = \{1, 2, 3, \ldots, v\}$, set kendaraan (truk dan *drone*)
- $|V| = v$, jumlah set kendaraan tersedia
- α adalah *weight coefficient* konsumsi energi truk
- β adalah *weight coefficient* konsumsi energi *drone*
- γ adalah *weight coefficient setup cost* set kendaraan
- δ_k adalah *setup cost* set kendaraan
- t_{ijk} adalah waktu perjalanan dari node-i ke node-j oleh kendaraan-k
- d_{ij} adalah jarak dari node-i ke node-j
- w_k^t adalah konsumsi bahan bakar truk-k
- w_k^d adalah konsumsi bahan bakar *drone-k*
- D_i adalah *demand* pada node-i
- q_k^t adalah kapasitas truk
- q_k^d adalah kapasitas *drone*
- h_i adalah jarak vertikal untuk tiap *customer* di node-i
- a_i dan b_i masing-masing adalah *time window* untuk node-i

- z_{ik} adalah waktu tiba kendaraan-k pada node-i
- w_{ik} adalah *waiting time* kendaraan-k pada node-i
- s_{ik} adalah *serving time* pada node-i oleh kendaraan-k
- $x_{ijk} = \{0,1\}$, *decision variable* apabila truk-k melewati node-i ke node-j maka bernilai 1, bernilai 0 apabila tidak
- $y_{ik} = \{0,1\}$, *decision variable* apabila node-i dilayani oleh kendaraan-k maka bernilai 1, bernilai 0 apabila tidak

Multi-objective function:

$$min\left[\left(\alpha\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{v}x_{ijk}w_k^t t_{ijk}\right)+\left(\beta\sum_{i=2}^{n}\sum_{k=1}^{v}w_k^d s_{ik}\right)+\left(\gamma\sum_{j=2}^{n}\sum_{k=1}^{v}\delta_k x_{1jk}\right)\right]$$

dengan:

$i \neq j$

$i, j \in N$

$k \in V$ (3.22)

Subject to:

$$x_{iik} = 0 \qquad \forall i \in N, \forall k \in V \qquad (3.23)$$

$$x_{n+1,j,k} = 0 \qquad \forall j \in N, \forall k \in V \qquad (3.24)$$

$$\sum_{j=2}^{c}\sum_{k=1}^{v}x_{1jk} \leq |V| \qquad (3.25)$$

$$\sum_{i=1}^{n}D_i y_{ik} \leq q_k^t \qquad \forall k \in V \qquad (3.26)$$

$$\sum_{j=1}^{n}\sum_{k=1}^{v}x_{ijk} = 1 \qquad \forall i \in C \qquad (3.27)$$

$$\sum_{k=1}^{v}y_{ik} = 1 \qquad \forall i \in C \qquad (3.28)$$

$$\sum_{i=1}^{n}x_{ijk} = y_{jk} \qquad \forall j \in C, \forall k \in V \qquad (3.29)$$

$$\sum_{j=1}^{n}x_{ijk} = y_{ik} \qquad \forall i \in C, \forall k \in V \qquad (3.30)$$

$$\sum_{j=1}^{n} x_{1jk} = 1 \qquad \forall k \in V \qquad (3.31)$$

$$\sum_{i=1}^{n} x_{ilk} - \sum_{j=1}^{n} x_{ljk} = 0 \qquad \forall l \in C, \forall k \in V \qquad (3.32)$$

$$\sum_{i=1}^{n} x_{i,n+1,k} = 1 \qquad \forall k \in V \qquad (3.33)$$

$$t_{ijk} = \frac{60 d_{ij}}{v_k} \qquad \forall i,j \in C, \forall k \in K \qquad (3.34)$$

$$s_{ik} = \left\lceil \frac{D_i}{q_k^d} \right\rceil \frac{2 h_i y_{ik}}{v_k^d} \qquad \forall i \in C, \forall k \in V \qquad (3.35)$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} x_{ijk} \left(t_{ijk} + s_{ik} + w_{ik} \right) \leq r_k \qquad \forall k \in V \qquad (3.36)$$

$$x_{ijk}(z_{ik} + w_{ik} + s_{ik} + t_{ijk} - z_{jk}) = 0 \qquad \forall i,j \in N, \forall k \in K \qquad (3.37)$$

$$a_i \leq z_{ik} + w_{ik} \leq b_i \qquad \forall i \in N, \forall k \in K \qquad (3.38)$$

$$a_i \leq z_{ik} + w_{ik} + s_{ik} \leq b_i \qquad \forall i \in N, \forall k \in K \qquad (3.39)$$

$$x_{ijk} \in \{0,1\} \qquad \forall i,j \in C, \forall k \in V \qquad (3.40)$$

$$y_{ik} \in \{0,1\} \qquad \forall i \in C, \forall k \in V \qquad (3.41)$$

Decision variables:

$$x_{ijk} = \begin{cases} 1, jika\ kendaraan - k\ melewati\ node - i\ ke\ node - j \\ 0, jika\ tidak \end{cases} \qquad (3.42)$$

$$y_{ik} = \begin{cases} 1, jika\ node - i\ dilayani\ oleh\ kendaraan - k \\ 0, jika\ tidak \end{cases} \qquad (3.43)$$

Fungsi objektif (3.22) bertujuan untuk meminimalkan biaya keseluruhan dari total konsumsi energi truk, total konsumsi energi *drone*, dan jumlah total truk. Persamaan (3.23) melarang kendaraan dari node-i untuk kembali ke node-i lagi. Persamaan (3.24) memastikan bahwa titik terakhir adalah *n+1* (depot). Persamaan (3.25) menunjukkan total kendaraan yang digunakan harus kurang dari total kendaraan tersedia (*V*). Persamaan (3.26) memastikan bahwa *demand* tidak melebihi kapasitas truk. Persamaan (3.27) – (3.30) memastikan bahwa setiap pelanggan hanya dikunjungi oleh satu truk dan dilayani hanya satu kali. Persamaan (3.31) – (3.33) memastikan bahwa setiap truk mulai dan berakhir di depot. Persamaan (3.34) dan (3.35) menyatakan travel time dengan truk-k dan

durasi pelayanan pada node-i oleh set kendaraan-k. Selanjutnya yaitu persamaan (3.36) menjamin waktu perjalanan maksimal tidak terlampaui. Sedangkan persamaan (3.37) – (3.39) digunakan untuk memodelkan batasan *time window*. Untuk persamaan (3.40) dan (3.41) menunjukkan bahwa nilai x_{ijk} dan y_{ik} adalah biner. Pada kasus ini, *decision variables* (3.42) dan (3.43) yang dibutuhkan adalah x_{ijk} dan y_{ik} jika bernilai 1 dan 0 dengan keterangan seperti persamaan di atas.

3.3 Pengembangan Algoritma

Basis dalam kasus MO-VRPTW-D adalah VRPTW dengan memodifikasi *service time* berdasarkan ketinggian *customer* dan faktor pemilihan kendaraan dari *set* kendaraan yang disediakan serta faktor multi-objektif. Sehingga, dalam pembuatan algoritma MO-VRPTW-D dengan SA perlu dilakukan terlebih dahulu pembuatan algoritma VRPTW. Selain itu, pada tahap awal pembentukan algoritma VRPTW diperlukan pula pembentukan algoritma PFIH untuk menentukan solusi awal yang layak. Berikut adalah tahapan dalam pengembangan algoritma MO-VRPTW-D dengan Algoritma *Simulated Annealing* dan *Ant Colony Optimization*.

3.3.1 Pengembangan Algoritma PFIH untuk Inisialisasi Solusi Awal

Algoritma PFIH dimaksudkan untuk mencari solusi awal yang layak dengan metode heuristik. Metode PFIH sendiri dipilih karena sudah banyak digunakan dan terbukti signifikan mampu menghasilkan solusi awal layak yang cukup bagus. Pada bab sebelumnya telah dijelaskan kerangka PFIH secara umum dalam membentuk solusi awal. Berikut adalah *pseudocode* PFIH yang dapat diterapkan untuk kasus MO-VRPTW dan telah dimodifikasi oleh Penulis.

Tabel 7 Algoritma *Push-Forward Insertion Heuristic* untuk Kasus MO-VRPTW-D

Algoritma 4: Algoritma *Push-Forward Insertion Heuristic* MO-VRPTW-D	
	Input: $xy, w_k^t, w_k^d, \delta_k, kota, kendaraan, it_{max}$
	Output: $rute_0, infeasibility, E_0$
1	$\alpha = 0.7, \beta = 0.1, \gamma = 0.2$
2	Input parameter kota
3	$rute_{now}$: rute awal beranggotakan matriks 0
4	$it = 1$
5	**while** $it \leq it_{max}$

```
Inisialisasi rute awal (rute kosong)
for i = 2: length(kota)
    Hitung biaya PFIH
end
Urutkan customer dengan biaya PFIH terkecil hingga terbesar
Masukkan seluruh customer ke kendaraan 1 sesuai urutan
Update feasibilitas
k = 2
while infeasibilility > 0
    Input parameter kendaraan
    Update feasibilitas1
    while infeasibility1 > 0
        Pilih 1 customer untuk dipindah ke set kendaraan k + 1
        Update infeasibility1 rute − k
    end
    k = k + 1
end
Update biaya
if it = 1
    rute_now = rute_0
else
    if k(it) < k(it − 1)
        rute_now = rute_0
    else
        if E_0(it) > E_0(it − 1)
            E_0(it) = E_0(it − 1)
            rute_now(it) = rute_now(it − 1)
        else
            rute_now = rute_0
        end
    end
end
```

38 **clear** $rute_0$

39 $it = it + 1$

40 **end**

41 $rute_0 = rute_{now}(it - 1)$

42 $E_0 = E_0(it - 1)$

Input yang dibutuhkan adalah data kendaraan dan data kota diantaranya adalah koordinat, *demand*, *time window*, jarak vertikal, kapasitas, kecepatan, dan *range* waktu maksimum. Hal pertama yang harus dilakukan adalah mendefinisikan nilai α = 0.7, β = 0.1, γ = 0.2 menurut Thangiah (1999). Perhitungan biaya dari depot untuk masing-masing pelanggan dilakukan untuk menentukan rute awal sehingga peluang untuk *feasible* adalah besar. Berikut adalah perhitungan biaya dari depot.

$$c_i = -\alpha d_{oi} + \beta b_i + \gamma \left(\frac{P_i}{360}\right) d_{oi} \tag{3.44}$$

Dengan d_{oi} adalah jarak dari depot ke pelanggan tersebut, b_i adalah waktu akhir pelanggan *i*, dan P_i adalah sudut koordinat polar untuk pelanggan *i* dari depot. Sehingga dapat disimpulkan bahwa biaya PFIH pelanggan dipengaruhi oleh jarak ke depot, waktu akhir, dan sudut polar. Pelanggan dengan biaya PFIH minimum diletakkan setelah depot dan diurutkan pada urutan selanjutnya berdasarkan biaya terkecil. Pada awal iterasi, seluruh *customer* diletakkan pada set kendaraan 1. Apabila *feasible*, maka solusi tersebut yang akan menjadi input di Algoritma SA. Namun apabila tidak, maka set kendaraan harus ditambah dan satu per satu *customer* dari set kendaraan 1 harus dipindah ke set kendaraan 2, dan seterusnya hingga *feasible*. Prinsip dari Algoritma PFIH untuk MO-VRPTW-D sendiri adalah melakukan perhitungan biaya pada setiap iterasi untuk solusi yang *feasible*. Solusi dari iterasi berikutnya akan diterima apabila menghasilkan biaya lebih kecil dari iterasi sebelumnya. Maka dari itu, metode PFIH tergolong sebagai metode heuristik. Solusi awal akan dipilih dari solusi pada iterasi terakhir, karena solusi dari iterasi terakhir adalah solusi paling bagus diantara solusi lain.

3.3.2 Pengembangan Algoritma SA untuk MO-VRPTW-D

Algoritma *Simulated Annealing* digunakan untuk membangun model MO-VRPTW-D dengan dibantu Algoritma PFIH dalam membentuk solusi awal.

Model MO-VRPTW-D yang dibangun adalah modifikasi konstrain dari model VRPTW pada subbab sebelumnya. Sebelum masuk tahap pengujian dengan data yang telah direkap, model MO-VRPTW-D dengan SA juga akan divalidasi terlebih dahulu untuk memastikan bahwa alur algoritma yang telah dibuat tersebut benar atau valid. Berikut adalah *pseudocode* SA yang dapat diterapkan untuk MO-VRPTW-D.

Tabel 8 Algoritma *Simulated Annealing* untuk Kasus MO-VRPTW-D

Algoritma 5: Algoritma *Simulated Annealing* MO-VRPTW-D

Input: $xy, w_k^t, w_k^d, \delta_k, kota, kendaraan, c, it_{max}$

Output: $rute, E_{SA}$

Input parameter kota dan kendaraan
$it_{PFIH} = 1$
$it = 1$
$it_{con} = 10$
$n = 1$
$n_{max} = 10$
Inisialisasi E_{SA} awal
$rute_0$: rute awal terbaik dari Algoritma PFIH
Inisialisasi rute awal
$\Delta E = 10$
$T_0 = E_{SA}$
$T = T_0$
while $it \leq it_{max}$
    if $\Delta E > 1e - 8$
        while $n \leq n_{max}$
            Inisialisasi rute1 dan rute2
            if $it = 1$
                $rute(it) = rute_0$
            else
                $rute(it) = rute(it - 1)$
            end
            $r_1 = rand$

```
        r_11 = pilih 2 customer random
        I = min(r_11)
        J = max(r_11)
        if r_1 ≤ 0.33
            flip customer
        elseif 0.33 < r_1 ≤ 0.67
            swap customer
        else
            slide customer
        end
        Update feasibilitas
        if infeasibility = 0
            Update biaya
            if E_SA(it) < E_SA(it − 1)
                rute = rute(it)
                E_SA = E_SA(it)
            else
                r = rand
                if r ≤ e^(−ΔE/kT)
                    rute = rute(it)
                    E_SA = E_SA(it)
                end
            end
        end
        n = n + 1
    end
else
    break
end
if it < it_con
    ΔE = 10
```

else

  $\Delta E = std(E_{SA}(it - (it_{con} - 1)) : E_{SA}(it))$

end

$T = c \cdot T$

$n = 1$

$it = it + 1$

end

for $j = 1 : (it - 1)$

  Hitung E_{SA}

end

$rute = rute(it - 1)$

$E_{SA} = E_{SA}(it - 1)$

Algoritma *Simulated Annealing* cukup baik dan tidak rumit untuk diterapkan di kasus diskret seperti MO-VRPTW-D. Prinsip utama dari Algoritma *Simulated Annealing* adalah *generate* rute pada temperatur tertentu dan jumlah siklus yang didefinisikan sebelum temperatur diturunkan untuk memperoleh solusi optimal. Mekanisme pembangkitan solusi untuk keluar dari jebakan optimum lokal memakai prinsip peluang distribusi energi Boltzman dimana pada temperatur awal yang cukup tinggi, partikel akan mudah untuk menerima solusi yang lebih buruk dalam artian solusi masih jauh dari konvergensi. Semakin turun temperatur, semakin ketat solusi untuk menerima solusi lebih buruk dan kemungkinan solusi yang dihasilkan tersebut sudah mendekati optimum global dan konvergensi. Peluan penerimaan solusi lebih buruk (disebut juga dengan kriteria metropolis) dibatasi dengan persamaan berikut.

$$P(E) \leq e^{\frac{-\Delta E}{kT}} \tag{3.45}$$

Apabila solusi dari pembangkitan rute *random* ternyata lebih buruk dari solusi sebelumnya, maka perlu dibangkitkan *random* yang kemudian dibandingkan dengan nilai *P(E).* Apabila nilainya lebih kecil atau sama dengan, maka terima solusi lebih buruk itu, sebaliknya jika nilainya lebih besar. $-\Delta E$ merupakan selisih dari solusi sekarang dan solusi sebelumnya. Nilai *k* disederhanakan menjadi 1 sedangkan nilai *T* adalah temperatur saat itu. Apabila rute baru diterima, diperlukan uji *feasibility*, apabila lolos, maka rute tersebut

diterima. Namun apabila rute baru yang dihasilkan tidak *feasible,* maka yang digunakan untuk operasi selanjutnya adalah rute lama. Prinsip dasar Algoritma *Simulated Annealing* untuk MO-VRPTW-D sebenarnya adalah menukar rute berbasis *flip, swap,* dan *slide.* Pada setiap iterasi, *customer* akan di *flip, swap,* dan *slide* sesuai dengan *random number* yang dibangkitkan disertai uji *feasibility*. Pemberhentian akan dilakukan apabila *stopping criteria* telah dicapai dimana dalam kasus ini yang menjadi *stopping criteria* adalah iterasi maksimum atau konvergensi sebesar 10^{-8} untuk 10 iterasi terakhir.

3.3.3 Pengembangan Algoritma ACO untuk MO-VRPTW-D

Berbeda dengan Algoritma *Simulated Annealing* yang membutuhkan bantuan Algoritma PFIH untuk membentuk solusi awal yang *feasible,* Algoritma *Ant Colony Optimization* tidak dibantu Algoritma PFIH karena inisialisasi rute yang *feasible* akan menggunakan Algoritma ACO itu sendiri sebelum menuju tahap selanjutnya yaitu menemukan hasil optimal untuk kasus MO-VRPTW-D. Model MO-VRPTW-D yang dibangun oleh Algoritma ACO akan dibandingkan dengan hasil dari Algoritma SA. Kedua metode ini sama-sama bagus untuk menangani problem diskret sehingga relevan untuk dibandingkan. Sama seperti Algoritma SA, sebelum masuk tahap pengujian dengan data yang telah direkap, model MO-VRPTW-D dengan ACO juga akan divalidasi terlebih dahulu untuk memastikan bahwa alur algoritma yang telah dibuat tersebut benar atau valid. Berikut adalah *pseudocode* ACO yang dapat diterapkan untuk MO-VRPTW-D.

Tabel 9 Algoritma *Ant Colony Optimization* untuk Kasus MO-VRPTW-D

Algoritma 6: Algoritma *Ant Colony Optimization* MO-VRPTW-D	
	Input: $xy, w_k^t, w_k^d, \delta_k, kota, kendaraan, N, it_{max}$
	Output: $rute, E_{ACO}$
1	Input parameter kota dan kendaraan
2	$\alpha = 1, \beta = 3, \rho = 0.5$
3	$it = 1$
4	$iter = jumlah\ kota - 2$
5	Inisialisasi τ_0
6	Inisialisasi E_{ACO}
7	**while** $it < it_{max}$

8 **for** $i = 1: jumlah\ kota$

9 **for** $j = 1: jumlah\ kota$

10 **for** $k = 1: jumlah\ kendaraan$

11 **for** $l = 1: jumlah\ semut$

12 Hitung matriks visibilitas

13 **end**

14 **end**

15 **end**

16 **end**

17 **for** $l = 1: jumlah\ semut$

18 $$p(1,j,k) = \begin{cases} \dfrac{\tau(1,j,k)^{\alpha}\eta(1,j,k)^{\beta}}{\sum_{u \in M_k}\tau(1,u,k)^{\alpha}\eta(1,u,k)^{\beta}}, jika\ s \in M_k \\ 0, jika\ tidak \end{cases}$$

19 **end**

20 $r = rand(N, 1)$

21 **for** $l = 1: jumlah\ semut$

22 **for** $i = 1: jumlah\ kota - 1$

23 **if** $r \leq cummulative\ probability\ dari\ kota\ pertama$

24 Masukkan *customer* baru sesuai *roulette wheel selection*

25 **break**

26 **end**

27 **end**

28 **end**

29 **for** $l = 1: jumlah\ semut$

30 Hitung nilai *feasibility1* untuk rute terbentuk

31 **end**

32 **while** $sum(infeasibility1) > 0$

33 Cari *customer* dengan metode sebelumnya hingga *feasible*

34 **end**

35 **for** $l = 1: jumlah\ semut$

36 Beri nilai 0 pada visibilitas kolom rute terpilih semua baris

end

while $iter > 0$

  for $l = 1: jumlah\ semut$

    Matriks visibilitas bernilai=0, $nveh = nveh + 1$

    $$p(i,j,k) = \begin{cases} \dfrac{\tau(i,j,k)^{\alpha}\eta(1,j,k)^{\beta}}{\sum_{u \in M_k}\tau(i,u,k)^{\alpha}\eta(i,u,k)^{\beta}}, jika\ s \in M_k \\ 0, jika\ tidak \end{cases}$$

  end

  $r = rand(N, 1)$

  for $l = 1: jumlah\ semut$

    for $i = 1: jumlah\ kota - 1$

      if $r \leq cummulative\ probability$

        Masukkan *customer* baru sesuai *roulette wheel selection*

        break

      end

    end

  end

  for $l = 1: jumlah\ semut$

    Hitung nilai *feasibility* untuk rute terbentuk

  end

  while $sum(infeasibility) > 0$

    Cari *customer* dengan metode sebelumnya

    for $l = 1: jumlah\ semut$

      Hitung nilai *feasibility* untuk rute terbentuk

    end

    if $sum(infeasibility) > 0$

      Periksa seluruh kemungkinan rute

      Beri nilai 0 pada matriks visibilitas untuk rute *infeasible*

    end

  end

  for $i = 1: jumlah\ kota$

```
        for j = 1: jumlah kota
            for k = 1: jumlah kendaraan
                for l = 1: jumlah semut
                    Update matriks visibilitas
                end
            end
        end
    end
    iter = iter − 1
end
for l = 1: jumlah semut
    Hitung biaya dari masing-masing semut
end
for l = 1: jumlah semut
    for l = 1: jumlah semut
        for l = 1: jumlah semut
            τ_ijk = (1 − ρ)τ_ijk + Δτ_ijk
        end
    end
end
end
iter = jumlah kota − 2
it = it + 1
end
Hitung biaya rute semut dan rute hasil konversi matriks feromon
E_ACO = min(E_semut, E_feromon)
rute = rute(E_ACO)
```

Nilai visibilitas $\eta(i,j,k)$ dihitung dari $\frac{1}{d_{ijk}}$ dimana untuk anggota matriks yang baris dan kolomnya sama akan diberi nilai 0. Selanjutnya adalah dengan membentuk τ_0 bernilai 0,01 (belum pernah dilewati semut dan akan bertambah jika dilewati sebelum berkurang karena penguapan). Peluang semut dari

customer-i menuju *customer-j* untuk masing-masing kendaraan-*k* dengan menggunakan rumus berikut dimana peluang untuk *customer* yang sudah dikunjungi akan diberi nilai 0 pada kolom tersebut.

$$p(i,j,k) = \begin{cases} \frac{\tau(i,j,k)^{\alpha}\eta(i,j,k)^{\beta}}{\sum_{u \in M_k}\tau(i,u,k)^{\alpha}\eta(i,u,k)^{\beta}}, jika\ s \in M_k \\ 0, jika\ tidak \end{cases} \quad (3.46)$$

Setelah peluang dikunjungi untuk masing-masing *customer-i,* perlu dilakukan perhitungan peluang kumulatif agar *roulette wheel selection* dapat dilakukan dengan menggunakan bilangan random. Bilangan akan menunjukkan *customer* mana yang harus dikunjungi setelahnya. Apabila *customer-j feasible* untuk dikunjungi, kolom pada *customer-j* akan diberi nilai 0 dan perhitungan selanjutnya akan dimulai dari *customer-j*. Namun jika tidak, kolom pada *customer-j* akan tetap diberi nilai 0 tetapi perhitungan akan diulang dari *customer-i* dengan tujuan agar tidak menuju ke *customer-j* pada perhitungan tersebut. Apabila seluruh *customer* telah dikunjungi, feromon harus di-*update* untuk mengetahui dari *N* semut, mayoritas urutan *customer* yang harus dikunjungi. Sehingga *customer* yang sering dikunjungi di awal akan memberikan feromon yang besar.

$$\tau_{ijk} = (1-\rho)\tau_{ijk} + \Delta\tau_{ijk} \quad (3.47)$$

Setelah feromon di-*update*, feromon baru akan digunakan di iterasi selanjutnya. Iterasi selanjutnya berlangsung sama seperti iterasi sebelumnya. Iterasi dilakukan hingga batas iterasi maksimum dicapai sehingga diharapkan hasil dari perhitungan sudah mencapai konvergen. Pada akhir iterasi, *customer* dengan feromon terbesar menunjukkan *customer* tersebut akan dikunjungi lebih awal.

3.4 Rekap Data

Data yang akan digunakan dalam penelitian Tugas Akhir ini adalah data sekunder dimana Penulis mengambil data sekunder yaitu data yang dibuat oleh Solomon dan Gehring & Homberger baik itu untuk validasi maupun eksperimen nantinya. Data sekunder yang dibuat oleh Solomon adalah data dengan jumlah 100 *customer* dimana data ini sudah pernah dilakukan penelitian sebelumnya dan sudah terdapat *best known solutions* apabila perlu dilakukan *benchmark*. Berbeda dengan data yang dibuat oleh Solomon, data yang dibuat Gehring & Homberger

adalah data dengan jumlah *customer* 200 – 1000 (hanya akan dipakai untuk 200 *customer*). Dalam subbab ini selain rekap data dengan parameter yang ada di kasus VRPTW, Penulis juga menyediakan beberapa data teknis yang dibutuhkan dalam perhitungan biaya dalam algoritma yang dibuat nantinya. Berikut adalah data teknis dan data parameter yang dibutuhkan di perhitungan dalam algoritma terkait yang telah dinormalisasi.

Tabel 10 Parameter Data Uji

No.	Parameter	Nilai	Satuan
1	*Weight coefficient* energi truk (α)	0,6	-
2	*Weight coefficient* energi *drone* (β)	0,1	-
3	*Weight coefficient setup* kendaraan (γ)	0,3	-
4	Biaya *setup* $T_k D_k$ (δ_k)	183,8 – 269,7	Euro
5	Jenis truk-k (T_k)	1 – 10	Jenis
6	Jenis *drone*-k (D_k)	1 – 10	Jenis
7	Konsumsi energi truk-k (w_k^t)	0,1042 – 0,1563	Euro/menit
8	Konsumsi energi *drone*-k (w_k^d)	0,0104 – 0,0156	Euro/menit
9	Kapasitas truk-k (q_k^t)	225 – 400	Unit
10	Kapasitas *drone*-k (q_k^d)	11 – 15	Unit
11	Kecepatan truk-k (v_k^t)	50 – 60	mil/jam
12	Kecepatan vertikal *drone*-k (v_k^d)	8 – 15	m/s
13	Waktu tempuh maksimal truk-k (r_k)	1000 – 1500	menit
14	Jarak vertikal *customer* (h)	5 – 14	meter

Note: data diambil dari *paper* dan referensi terkait lainnya, namun ada beberapa data yang di-*random* oleh Penulis, setiap jenis truk dan *drone* terdapat 10 unit

Tabel 11 Kombinasi Truk dan *Drone*

Truk						
k	T_k	w_k^t	q_k^t	v_k^t	r_k	δ_k^t
1	T1	0,1042	225	60	1000	153,2
2	T2	0,1042	225	60	1050	153,2
3	T3	0,125	250	58	1100	168,5
4	T4	0,125	250	58	1200	168,5

Truk						
k	T_k	w_k^t	q_k^t	v_k^t	r_k	δ_k^t
5	T5	0,1354	300	55	1250	183,8
6	T6	0,1354	300	55	1300	183,8
7	T7	0,1458	350	53	1350	199,2
8	T8	0,1458	350	53	1400	199,2
9	T9	0,1563	400	50	1450	214,5
10	T10	0,1563	400	50	1500	214,5

Drone					
k	D_k	w_k^d	q_k^d	v_k^d	δ_k^d
1	D1	0,01042	11	15	30,6
2	D2	0,01042	11	14	36,8
3	D3	0,01042	12	13	42,9
4	D4	0,0125	12	12	49
5	D5	0,01354	13	11	55,2
6	D6	0,01354	13	10	61,4
7	D7	0,01458	14	10	67,6
8	D8	0,01458	14	9	73,8
9	D9	0,01563	15	9	80
10	D10	0,01563	15	8	86,2

Note: Semakin besar kapasitas truk dan/atau *drone*: semakin besar konsumsi bahan bakar, semakin kecil kecepatan, semakin lama waktu tempuh maksimal, dan semakin besar biaya yang dikeluarkan

Data *weight coefficient* diperoleh dari prioritas objektif itu sendiri. Semakin prioritas objektif tersebut maka semakin besar bobot yang diberikan. Dalam hal ini, meminimasi konsumsi energi truk adalah objektif yang paling diprioritaskan. Biaya *setup* set kendaraan diperoleh dari biaya *setup* truk ditambah dengan biaya *setup drone* yang diambil dari rata-rata biaya sewa per operasi. Terdapat 10 jenis truk dan 10 jenis *drone* (masing-masing terdapat 10 unit) yang nantinya akan dikombinasikan sehingga terdapat 100 jenis set kendaraan tersedia. Konsumsi energi truk dan *drone* diperoleh dari modifikasi data pada Sacramento, et al. (2019) dimana Penulis memodifikasi untuk menambah variasi. Variasi biaya

tergantung pada spesifikasi truk atau *drone* dimana semakin tinggi spesifikasinya membutuhkan biaya operasi yang lebih mahal pula. Untuk kapasitas truk dan *drone* disesuaikan dengan data sekunder yaitu Solomon dan Gehring & Homberger *benchmark*. Kecepatan truk sendiri didapat dari Sacramento, et al. (2019), sedangkan untuk kecepatan vertikal *drone* sendiri berbeda dengan kecepatan horizontalnya karena beban yang dibawa secara vertikal lebih berat sehingga mengakibatkan kecepatannya melambat. Waktu tempuh maksimal untuk masing-masing jenis truk disesuaikan dengan dengan data sekunder sehingga tidak terdapat truk yang memiliki waktu tempuh yang sangat jauh dibandingkan truk lainnya. Data uji terakhir adalah jarak vertikal *customer* yang diperoleh dari *random number* dari 5 – 14 sesuai yang digunakan dari Han, et al. (2020).

3.4.1 Rekap Data Validasi

Data yang akan digunakan dalam validasi algoritma adalah data dengan 5 *customer*. Data ini nantinya yang akan diinput ke *software* LINGO sebagai *software* yang digunakan untuk optimasi metode eksak. Setelah itu, data juga akan di-*input* ke *software* MATLAB dengan metode SA untuk optimasi metaheuristik sebagai proses validasi. Berikut adalah rekap data untuk validasi dengan 5 *customer*.

Tabel 12 Data Validasi

i	x	y	D	a	b	h	Kombinasi
0	35	35	0	0	690	0	T1D1
1	41	49	10	322	513	7	
2	35	17	7	100	180	6	
3	55	45	13	232	378	8	
4	55	20	19	298	477	6	
5	15	30	26	68	132	14	

Note: diambil dari 6 data pertama Solomon *Benchmark R101-Type*

Tabel 13 Kombinasi Truk dan *Drone* untuk Validasi

k	T_k	D_k	w_k^t	w_k^d	q_k^t	q_k^d	v_k^t	v_k^d	r_k	δ_k
1	T1	D1	0,1042	0,0104	225	11	0,667	10	1000	183,8

Note: diambil dari Data Kombinasi Truk 1 dan *Drone* 1, kecepatan truk dalam satuan mil/menit, dimana: $\delta_k^t + \delta_k^d = \delta_k$

3.4.2 Solomon Benchmark

Data yang dibuat oleh Solomon pada tahun 1987 telah banyak digunakan sebagai penelitian sehingga sesuai untuk dijadikan *benchmark* karena telah terdapat *best known solutions* dari bermacam-macam penelitian. Data yang dibuat Solomon nantinya akan dibuat model skenario 1 – 18 dengan masing-masing jumlah *customer* adalah 25, 60, dan 100. Selain itu, data juga akan dimodifikasi sesuai kebutuhan penelitian Tugas Akhir dengan menambahkan jenis *demand* untuk setiap *customer*. Namun untuk proses sebagai bahan analisis performansi untuk kasus VRPTW tetap akan menggunakan data asli tanpa modifikasi. Berikut adalah rekap data untuk Solomon *Benchmark* dengan jumlah *customer* 25, 60, dan 100.

Tabel 14 Solomon *Benchmark*

No.	Tipe	Pelanggan
1	R101	25
2		50
3		100
4	C101	25
5		50
6		100
7	RC101	25
8		50
9		100
10	R201	25
11		50
12		100
13	C201	25
14		50
15		100
16	RC201	25
17		50
18		100

3.4.3 Gehring & Homberger Benchmark

Data yang dibuat oleh Gehring & Homberger pada tahun 2005 telah banyak dimanfaatkan untuk dijadikan *extended benchmark* untuk kasus VRPTW dan terdapat juga penelitian sebelumnya yang mencantumkan *best known solutions*. Data yang dibuat Gehring & Homberger nantinya akan dibuat model skenario 19 – 24 dengan jumlah *customer* adalah 200. Sama seperti skenario 1 – 18, data juga akan dimodifikasi sesuai kebutuhan penelitian Tugas Akhir dengan menambahkan jenis *demand* untuk setiap *customer* dan untuk bahan analisis performansi dalam kasus VRPTW juga akan tetap menggunakan data asli tanpa modifikasi. Berikut adalah rekap data untuk Gehring & Homberger *Benchmark* dengan jumlah *customer* 200.

Tabel 15 Gehring & Homberger *Benchmark*

No.	Tipe	Pelanggan
1	R121	200
2	C121	200
3	RC121	200
4	R221	200
5	C221	200
6	RC221	200

3.5 Validasi Algoritma

Validasi adalah tahap untuk memastikan bahwa algoritma yang dibuat telah berjalan sebagaimana mestinya dan metode dikatakan valid apabila *output* yang dihasilkan metode terkait adalah benar. Dalam hal ini, validasi Algoritma SA dan ACO untuk MO-VRPTW yang telah dibuat dengan *software* MATLAB perlu dilakukan menggunakan metode eksak. Dalam hal ini Penulis menggunakan data dengan 5 *customer* dikarenakan metode eksak membutuhkan waktu komputasi yang sangat lama apabila menggunakan data *customer* yang banyak. Apabila dengan data dengan *customer* sedikit saja sudah benar maka untuk data besar sudah dipastikan valid, hanya permasalahan waktu komputasi saja yang menjadi kendala untuk metode eksak. *Coding* VRPTW yang dibuat dengan LINGO adalah valid dimana *output* LINGO membentuk *routing* (dari depot kembali ke depot) dengan rute kendaraan adalah 1-6-3-5-4-2-7 dengan total *travel time* adalah

179,227 menit. Untuk *coding* MO-VRPTW-D sendiri menunjukkan hasil serupa namun dengan komposisi kendaraan yang sama pula seperti yang ditunjukkan pada tabel 16 dengan nilai fungsi objektif sebesar 62,6217.

Tabel 16 Rekap Hasil *Running* LINGO untuk MO-VRPTW-D

No.	Set Kendaraan	Rute	Total Muatan
1	Set Kendaraan 1	1-6-3-5-4-2-7	75
Total			**75**

3.5.1 Validasi Algoritma PFIH untuk Inisialisasi Solusi Awal

```
rute0 =

     1     6     3     4     5     2     1


infeasibility =

     0


E0 =

   64.6116
```

Gambar 16 Hasil *Running* Algoritma PFIH dengan MATLAB untuk MO-VRPTW-D

Berikut adalah hasil *running code* Algoritma PFIH untuk inisialisasi solusi awal kasus MO-VRPTW-D dengan data 5 *customer* menggunakan MATLAB. Dari hasil *running* dengan 1 iterasi, Algoritma PFIH untuk MO-VRPTW-D menggunakan MATLAB juga menunjukkan nilai *infeability=0* yang artinya rute pada solusi awal adalah rute yang *feasible* secara *capacity*, *serving time*, *time window*, dan *maximum range*. Solusi awal yang diberikan juga mendekati optimal global dengan objektif sebesar 64,6116 (optimal global adalah 62,6217) dikarenakan terdapat sedikit perbedaan rute yang nantikan akan dengan mudah diselesaikan oleh Algoritma SA. Dengan hal ini dapat dikatakan bahwa Algoritma PFIH untuk inisialisasi solusi awal kasus MO-VRPTW-D adalah valid karena memperoleh rute awal yang *feasible*.

3.5.2 Validasi Algoritma SA untuk MO-VRPTW-D

```
rute =

     1     6     3     5     4     2     1


E_SA =

   62.6217
```

Gambar 17 Hasil *Running* Algoritma SA dengan MATLAB untuk MO-VRPTW-D

Berikut adalah hasil *running code* untuk kasus MO-VRPTW-D dengan data 5 *customer* menggunakan MATLAB untuk Algoritma SA. Dari hasil *running* Algoritma SA untuk MO-VRPTW-D juga menunjukkan hasil yang sama sebesar 62,6217 dengan yang diperoleh dari metode eksak. Selain itu, variabel keputusan (rute dan pemilihan kendaraan) yang dihasilkan juga sesuai dengan metode eksak. Dengan hal ini dapat dikatakan bahwa Algoritma SA untuk MO-VRPTW-D adalah valid karena memperoleh hasil optimal global. Validasi juga dikatakan cukup hanya dengan menggunakan data 5 *customer* karena jika dengan data 5 *customer* saja sudah valid, maka untuk data besar sudah dipastikan valid.

3.5.3 Validasi Algoritma ACO untuk MO-VRPTW-D

```
rute =

     1     6     3     5     4     2     1


E_ACO =

   62.6217
```

Gambar 18 Hasil *Running* Metode ACO dengan MATLAB untuk MO-VRPTW-D

Berikut adalah hasil *running code* untuk kasus MO-VRPTW-D dengan data 5 *customer* menggunakan MATLAB untuk Algoritma ACO. Dari hasil *running* Algoritma ACO untuk MO-VRPTW-D juga menunjukkan hasil yang sama sebesar 62,6217. Hasil tersebut juga sama dengan yang diperoleh dari metode eksak dan Algoritma SA. Variabel keputusan (rute dan pemilihan kendaraan) yang dihasilkan juga sesuai dengan metode eksak dan SA. Dengan hal ini dapat dikatakan bahwa Algoritma ACO untuk MO-VRPTW-D adalah valid karena memperoleh hasil optimal global. Sama dengan metode metaheuristik lain

(termasuk SA), validasi juga dikatakan cukup hanya dengan menggunakan data 5 *customer* karena jika dengan data 5 *customer* saja sudah valid, maka untuk data besar sudah dipastikan valid.

3.6 Eksperimen dan Analisis

Eksperimen akan dilaksanakan sesuai dengan skenario beberapa data yang terdapat di subbab sebelumnya. Data yang diperoleh dari penelitian sebelumnya akan dijadikan acuan dalam penelitian ini. Penulis akan mencoba menguji algoritma dengan data *customer* berjumlah 25 – 200. Masing-masing skenario akan direkap hasilnya yang kemudian diolah untuk mendapatkan kesimpulan. Untuk setiap skenario, pengujian akan dilakukan lima kali dengan mengambil nilai rata-rata, hasil terbaik, dan standar deviasi yang nantinya akan digunakan untuk menghitung performansi dengan menggunakan metode RPD (*Relative Percentage Deviation*) dengan membandingkan antara SA dengan ACO (ACO sebagai pembanding). Dari hasil yang didapat nantinya juga akan dianalisis terkait improvisasi pada multi-objektif apakah memberikan dampak secara signifikan terhadap pemilihan kendaraan atau tidak.

$$RPD = \frac{Best\ ACO - Best\ SA}{Best\ SA} \times 100\% \tag{3.48}$$

3.7 Kesimpulan dan Saran

Setelah menguji algoritma dengan beberapa skenario, akan diperoleh hasil yang dapat diolah dan disajikan layaknya penelitian statistika. Dari hasil tersebut juga akan diperoleh kesimpulan dari Penelitian Tugas Akhir. Kesimpulan yang didapat nantinya dapat merangkum keseluruhan isi penelitian. Selain itu, kesimpulan yang diperoleh juga akan menjawab Tujuan pada Bab 1. Selain kesimpulan, Penulis juga akan memberikan saran untuk penelitian selanjutnya dengan tujuan mampu menyelesaikan kasus yang lebih sesuai dengan kondisi *real* dan tentunya menyelesaikan kasus baru hasil pengembangan kasus ini yang bisa memberikan alternatif baru dan solusi yang lebih optimal.

(Halaman ini sengaja dikosongkan)

BAB IV
EKSPERIMEN DAN ANALISIS

Pada bab ini akan dibahas mengenai Pengujian Algoritma dengan beberapa skenario yang disertai dengan perubahan paramtere, Rekap Hasil dari pengujian algoritma, dan Analisis berupa analisis hasil dan analisis performansi melalui perbandingan Algoritma *Simulated Annealing* dan *Ant Colony Optimization*.

4.1 Pengujian Algoritma

Pengujian dengan skema skenario ditujukan untuk menguji sensitvitas metode. Setiap skenario dengan jumlah pelanggan sama memiliki 6 variasi yaitu tipe R1, C1, RC1, R2, C2, dan RC2. Masing-masing tipe data memiliki karakteristik yang berbeda. Berikut adalah skenario data uji yang berjumlah 24 skenario dengan jumlah pelanggan 25 – 200. Beberapa faktor yang mempengaruhi dari tipe data tersebut adalah geografis, *time window*, dan *serving time* yang berbeda. Data geografis mempengaruhi karena set problem R1 dan R2 dihasilkan secara *random*, sedangkan set problem C1 dan C2 dihasilkan secara *cluster* geografis, serta set problem oleh RC1 dan RC2 adalah gabungan antara *random* dan *cluster*. Set problem R1, C1 dan RC1 memiliki *time window* yang ketat. Sebaliknya, set R2, C2 dan RC2 memiliki *time window* panjang yang memungkinkan banyak pelanggan (lebih dari 30) dilayani oleh satu kendaraan saja. Dengan total 50 kombinasi kendaraan tersedia, dipastikan seluruh kasus akan *feasible* bahkan untuk 200 pelanggan.

Tabel 17 Skenario Data Uji

No.	Skenario	Tipe	Pelanggan
1	25.1	R101	25
2	25.2	C101	25
3	25.3	RC101	25
4	25.4	R201	25
5	25.5	C201	25
6	25.6	RC201	25
7	50.1	R101	50
8	50.2	C101	50

No.	Skenario	Tipe	Pelanggan
9	50.3	RC101	50
10	50.4	R201	50
11	50.5	C201	50
12	50.6	RC201	50
13	100.1	R101	50
14	100.2	C101	100
15	100.3	RC101	100
16	100.4	R201	100
17	100.5	C201	100
18	100.6	RC201	100
19	200.1	R121	200
20	200.2	C121	200
21	200.3	RC121	200
22	200.4	R221	200
23	200.5	C221	200
24	200.6	RC221	200

Note: Solomon *Benchmark* (25 – 100 *customer*) dan Gehring & Homberger *Benchmark* (200 *customer*)

4.1.1 Pengujian Algoritma SA untuk MO-VRPTW-D

Parameter yang menjadi nilai input dalam Algoritma SA adalah parameter penurunan temperatur (c) dan jumlah iterasi (it_{max}). Parameter penurunan temperatur bernilai antara 0 sampai 1. Semakin besar nilai dari parameter penurunan temperatur (c), semakin lama akan mencapai konvergensi (waktu komputasi akan semakin lama) karena semakin banyak titik yang dicoba. Namun apabila nilainya terlalu kecil, akan banyak titik yang terlewat yang mungkin saja titik terlewat tersebut adalah solusi optimal. Untuk jumlah iterasi, nilainya semakin besar akan semakin baik karena memungkinkan lebih banyak solusi yang dihasilkan. Namun semakin banyak jumlah iterasi juga akan mempengaruhi waktu komputasi. Penulis menggunakan *paper* Redi et al. (2020) sebagai acuan dimana nilai c adalah 0,9 dan iterasi maksimumnya adalah 100. Jumlah tersebut dinilai ideal untuk kasus kecil maupun kasus besar.

4.1.2 Pengujian Algoritma ACO untuk MO-VRPTW-D

Parameter yang menjadi nilai input dalam Algoritma ACO adalah jumlah semut (N) dan jumlah iterasi (it_{max}). Selain itu, terdapat parameter lain yaitu α (faktor feromon), β (faktor visibilitas), dan ρ (faktor penguapan feromon). Semakin besar nilai α, maka faktor feromon akna lebih dominan. Semakin besar nilai β, maka faktor visibilitas akan semakin dominan. Nilai ρ berkisar antara 0 hingga 1 yang dimana nilainya semakin kecil, maka penguapan feromon akan berjalan lambat yang artinya solusi dari iterasi sebelumnya akan semakin dipertimbangkan dibanding nilai ρ besar. Jumlah semut mengindikasikan berapa solusi yang dibangkitkan sekaligus dalam sekali iterasi. Tentu jumlah semut yang terlalu banyak akan memakan waktu komputasi yang sangat lama. Pada kasus MO-VRPTW-D, penyebab utama waktu komputasi lama adalah kompleksitas kasus tersebut. Namun bukan berarti parameter tidak berpengaruh di sini. Semakin banyak jumlah semut tentu akan semakin bagus karena akan semakin banyak solusi yang dihasilkan. Penulis menggunakan buku karya Santosa & Ai (2017) sebagai acuan dimana nilai $\alpha = 1, \beta = 3$, dan $\rho = 0{,}5$ dimana parameter ini cukup ideal untuk kasus MO-VRPTW-D pada penelitian ini. Sedangkan nilai N dan it_{max} yang digunakan penulis adalah 10 dan 100 (mempertimbangkan waktu komputasi).

4.2 Rekap Hasil

$$h_w = \frac{t_{n-1,\frac{\alpha}{2}} \times s}{\sqrt{n}} \tag{4.1}$$

$$n' = \left(\frac{z_{1-\frac{\alpha}{2}} \times s}{h_w} \right)^2 \tag{4.2}$$

Persamaan (4.1) dan (4.2) adalah persamaan untuk mencari minimal replikasi pada setiap skenario. Pengujian dilakukan sebanyak 5 kali sebagai replikasi awal untuk meminimalisir munculnya hasil yang bagus hanya secara kebetulan. Untuk itu, diperlukan beberapa kali replikasi sesuai rumus agar mendapatkan hasil yang merepresentasikan model dengan perhitungan rata-rata dan standar deviasi yang bisa digunakan untuk membandingkan kedua metode. Menurut perhitungan, replikasi minimal yang harus dilakukan adalah sebanyak 3 kali, sehingga replikasi awal sudah dinilai cukup. Dari hasil pengujian, berikut

adalah rekap hasil pengujian Metode SA dan ACO untuk kasus MO-VRPTW-D pada masing-masing skenario dengan pelanggan 25 – 200.

Tabel 18 Rekap Hasil

Skenario	Rata-Rata		Standar Deviasi / Rata – Rata		*Best*
	SA	ACO	SA	ACO	
25.1	**157,026**	264,929	0,016695	**0,00789**	157,026
25.2	**179,014**	248,642	0,17118	**0,004536**	179,014
25.3	197,150	**195,594**	0,183098	**0,160548**	195,594
25.4	**221,035**	358,826	**0,007625**	0,094515	221,035
25.5	**195,670**	376,589	**0,003596**	0,075106	195,670
25.6	**169,822**	362,277	**0,004657**	0,080942	169,822
50.1	**303,704**	456,492	0,124329	**0,010169**	303,704
50.2	456,948	**421,409**	**0,071046**	0,084976	421,409
50.3	558,341	**455,041**	0,063716	**0,00698**	455,041
50.4	591,040	**439,331**	**0,063372**	0,083925	439,331
50.5	**588,765**	656,764	0,05465	**0,041485**	588,765
50.6	**549,974**	554,505	**0,07052**	0,072646	549,974
100.1	**642,763**	820,570	**0,04127**	0,044878	642,763
100.2	1403,980	**630,639**	0,096181	**0,05345**	630,639
100.3	1213,820	**814,057**	**0,026106**	0,076269	814,057
100.4	1045,375	**1010,757**	0,086686	**0,085096**	1010,757
100.5	1594,480	**1027,000**	**0,010858**	0,050113	1027,000
100.6	1279,560	**922,300**	**0,054435**	0,068294	922,300
200.1	4271,800	**2066,860**	**0,028195**	0,034053	2066,860
200.2	4624,920	**2292,580**	0,070746	**0,023983**	2292,580
200.3	8015,500	**3195,020**	0,034935	**0,009533**	3195,020
200.4	6204,120	**3177,660**	0,032249	**0,015087**	3177,660
200.5	6315,180	**3062,280**	0,05049	**0,024623**	3062,280
200.6	6214,680	**2362,420**	0,05459	**0,051668**	2362,420

Note: nilai terbaik dari masing-masing skenario di-*bold*

4.3 Analisis

Pada subbab Analisis akan membahas penjelasan lebih lanjut terkait hasil pengujian pada masing-masing metode. Metode SA dan ACO dipilih karena mempunyai *track record* bagus untuk menyelesaikan problem diskret. Namun dalam hal ini akan dibuktikan juga metode mana yang memiliki hasil lebih bagus untuk kasus MO-VRPTW-D. Berikut akan disajikan analisis hasil dari Metode SA, Metode ACO, serta perbandingan antara Metode SA dan Metode ACO.

4.3.1 Analisis Hasil Pengujian Algoritma SA untuk MO-VRPTW-D

Dalam proses perhitungannya, Algoritma SA dibantu algoritma berbasis metode heuristik yaitu Algoritma PFIH. Solusi yang dihasilkan oleh Algoritma PFIH sudah cukup bagus karena solusi tersebut adalah lokal optimal. Namun dengan melanjutkan menggunakan Algoritma SA akan memperoleh hasil yang lebih optimal. Dari hasil perhitungan, Algoritma SA menghasilkan solusi dengan standar deviasi rendah pada Skenario 25.1, 25.4, 25.5, dan 25.6. Hal tersebut dikarenakan problem memiliki jumlah node yang tidak terlalu banyak serta tidak terlalu kompleks. Selain itu, waktu komputasi yang dibutuhkan oleh Algoritma SA juga tidak terlalu lama. Pada skenario lain, nilai standar deviasi yang tinggi disebabkan oleh penggunaan jumlah kendaraan yang berbeda. Terkadang menghasilkan solusi yang baik dikarenakan pada saat inisialisasi solusi awal hanya diperlukan jumlah kendaraan yang sedikit. Algoritma SA baru tidak cukup baik untuk menghasilkan solusi optimal pada skenario 100 pelanggan. Solusi yang dihasilkan sangat bervariasi sehingga menghasilkan standar deviasi yang tinggi pula. Selain itu, pada skenario 100 pelanggan dan 200 pelanggan, Penulis juga harus mengurangi jumlah iterasi guna memangkas waktu komputasi yang sangat lama. Kendati demikian, Algoritma SA cukup bagus secara keseluruhan.

4.3.2 Analisis Hasil Pengujian Algoritma ACO untuk MO-VRPTW-D

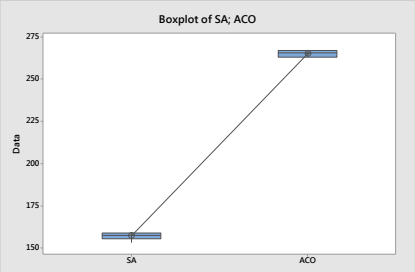
Algoritma ACO cukup rumit untuk kasus MO-VRPTW-D. Namun dibalik kerumitan tersebut ternyata menghasilkan solusi yang cukup bagus. Terbukti pada Skenario 25.1, 25.2, 50.1, dan 50.3 menghasilkan standar deviasi yang rendah. Artinya, Algoritma ACO tidak memiliki kesulitan ketika jumlah node dalam kasus berjumlah 50. Pada skenario lain, solusi yang dihasilkan begitu fluktuatif yang disebabkan oleh jumlah semut dalam iterasi kurang sesuai. Waktu

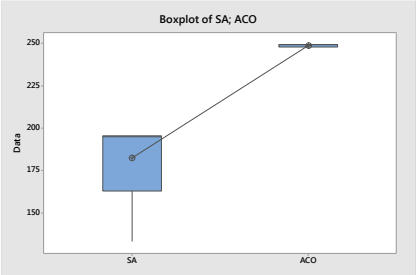
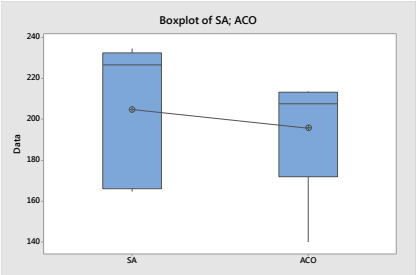
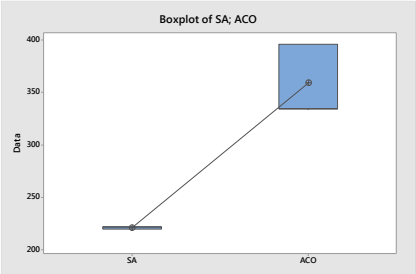
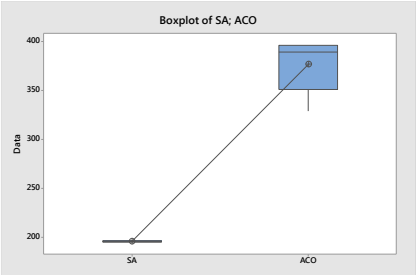
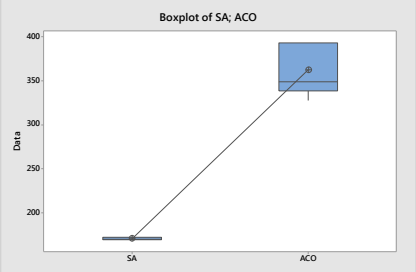
komputasi yang dibutuhkan Algoritma ACO sangat lama karena kompleksitas kasus dan kompleksitas algoritma tersebut. Kompleksitas algoritma yang dimaksud adalah *syntax* yang diinput ke *software*. Pembangkitan solusi dalam Algoritma ACO turut menyumbang penyebab waktu komputasi yang lama. Pembangkitan solusi yang secara simultan (sebanyak *N* semut) dan proses iterasi yang panjang adalah penyebabnya. Tidak semua solusi pada MO-VRPTW-D adalah *feasible,* sehingga perlu waktu yang cukup lama untuk melakukan proses iterasi ulang agar solusi yang dihasilkan *feasible*. Namun, ketika Algoritma ACO menyelesaikan skenario dengan jumlah pelanggan 100, solusi yang dihasilkan cukup bagus dan hal ini sejalan dengan waktu komputasi yang lama. Secara keseluruhan, Algoritma ACO mampu menyelesaikan kasus MO-VRPTW-D dengan bagus dengan kelemahannya adalah waktu komputasi yang lama dan algoritma yang rumit.

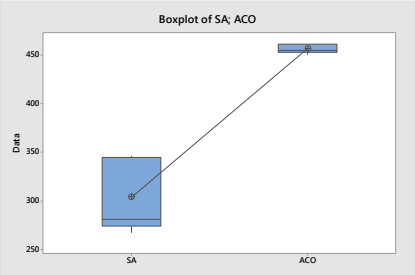
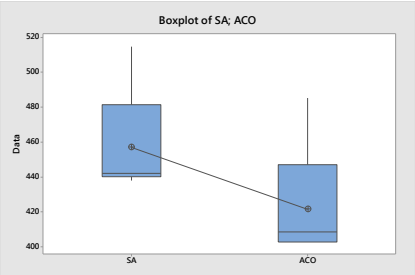
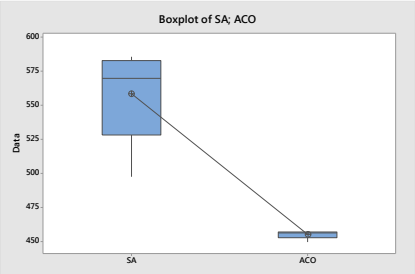
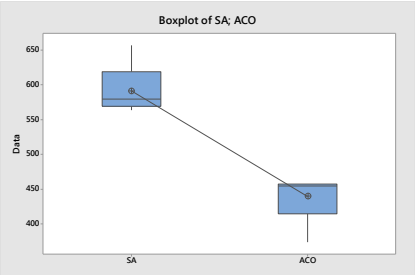
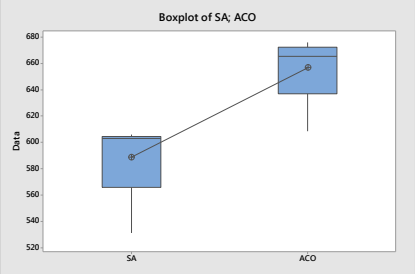
4.3.3 Analisis Perbandingan Metode SA dan ACO

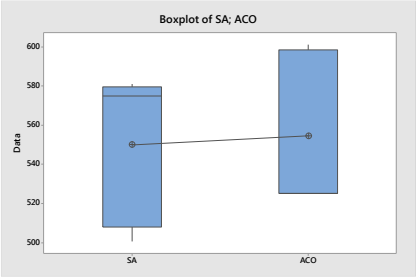
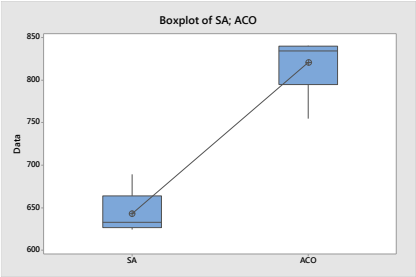
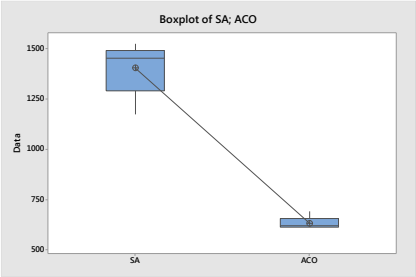
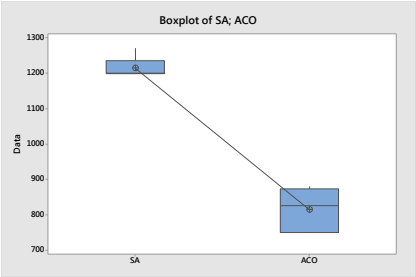
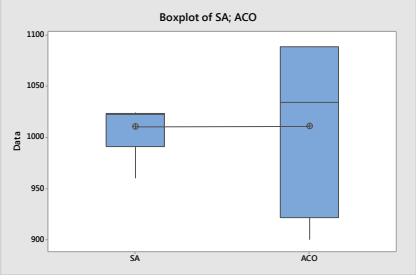
Tujuan membandingkan kedua metode adalah untuk mengetahui performansi dari masing-masing algoritma dikarenakan penelitian ini tidak memiliki data hasil *benchmark* yang sesuai. Selain itu, perbandingan antara dua metode juga akan membuktikan metode mana yang sebaiknya diterapkan untuk kasus MO-VRPTW-D. Metode yang baik adalah metode yang memiliki standar deviasi kecil yang menggambarkan bahwa metode tersebut memiliki tingkat konvergensi yang bagus sehingga rentang nilainya terbatas. Berikut adalah rekap perbandingan Metode SA dan Metode ACO disajikan dalam *boxplot* dari setiap skenario menggunakan ANOVA.

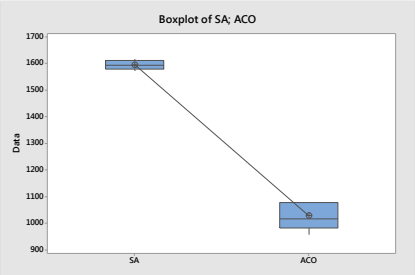
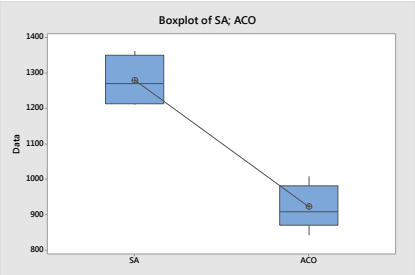
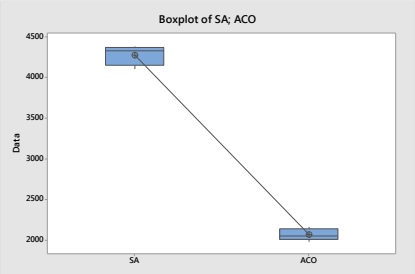
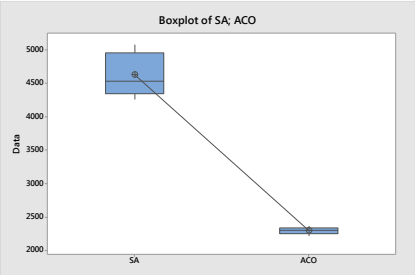
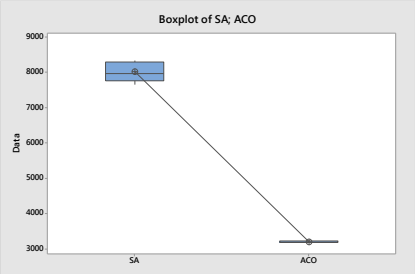
Tabel 19 Perbandingan SA dan ACO untuk MO-VRPTW-D

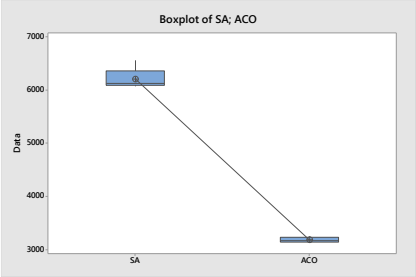
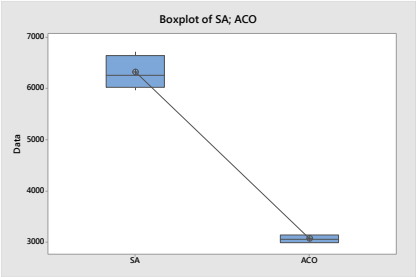
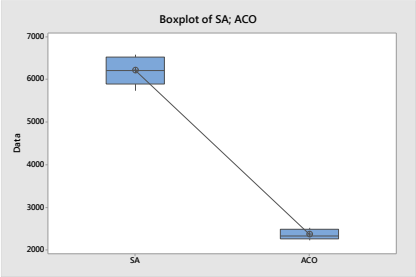
No.	Skenario	*Box Plot*	RPD	Terbaik
*1	Skenario 25.1	 **Gambar 19 SA vs ACO Skenario 25.1**	71,47%	SA

No.	Skenario	*Box Plot*	RPD	Terbaik
2	Skenario 25.2	 **Gambar 20 SA vs ACO Skenario 25.2**	85,32%	SA
3	Skenario 25.3	 **Gambar 21 SA vs ACO Skenario 25.3**	-15,07%	*ACO
4	Skenario 25.4	 **Gambar 22 SA vs ACO Skenario 25.4**	52,48%	SA
5	Skenario 25.5	 **Gambar 23 SA vs ACO Skenario 25.5**	68,79%	SA
6	Skenario 25.6	 **Gambar 24 SA vs ACO Skenario 25.6**	93,70%	SA

No.	Skenario	*Box Plot*	RPD	Terbaik
*7	Skenario 50.1	 **Gambar 25 SA vs ACO Skenario 50.1**	68,56%	SA
8	Skenario 50.2	 **Gambar 26 SA vs ACO Skenario 50.2**	-8,07%	*ACO
9	Skenario 50.3	 **Gambar 27 SA vs ACO Skenario 50.3**	-9,64%	ACO
10	Skenario 50.4	 **Gambar 28 SA vs ACO Skenario 50.4**	-33,76%	ACO
11	Skenario 50.5	 **Gambar 29 SA vs ACO Skenario 50.5**	14,55%	SA

No.	Skenario	*Box Plot*	RPD	Terbaik
12	Skenario 50.6	 **Gambar 30 SA vs ACO Skenario 50.6**	4,88%	*SA
13	Skenario 100.1	 **Gambar 31 SA vs ACO Skenario 100.1**	20,92%	SA
*14	Skenario 100.2	 **Gambar 32 SA vs ACO Skenario 100.2**	-48,06%	ACO
15	Skenario 100.3	 **Gambar 33 SA vs ACO Skenario 100.3**	-37,63%	ACO
16	Skenario 100.4	 **Gambar 34 SA vs ACO Skenario 100.4**	-6,27%	*ACO

No.	Skenario	*Box Plot*	RPD	Terbaik
17	Skenario 100.5	 **Gambar 35 SA vs ACO Skenario 100.5**	-39,10%	ACO
18	Skenario 100.6	 **Gambar 36 SA vs ACO Skenario 100.6**	-30,52%	ACO
*19	Skenario 200.1	 **Gambar 37 SA vs ACO Skenario 200.1**	-51,77%	ACO
20	Skenario 200.2	 **Gambar 38 SA vs ACO Skenario 200.2**	-47,98%	ACO
21	Skenario 200.3	 **Gambar 39 SA vs ACO Skenario 200.3**	-58,55%	ACO

No.	Skenario	*Box Plot*	RPD	Terbaik
22	Skenario 200.4	 **Gambar 40 SA vs ACO Skenario 200.4**	-48,40%	ACO
23	Skenario 200.5	 **Gambar 41 SA vs ACO Skenario 200.5**	-49,88%	ACO
24	Skenario 200.6	 **Gambar 42 SA vs ACO Skenario 200.6**	-61,27%	ACO

*: akan dibandingkan dengan *Single Objective Optimization*

Dari total 24 skenario, secara garis besar, kedua algoritma mampu menyelesaikan perhitungan dengan baik. Pada pengujian, terdapat juga perhitungan ANOVA dan RPD diantara kedua algoritma. Perhitungan ANOVA bertujuan untuk mengetahui apakah terdapat perbedaan signifikan antara kedua algoritma, bisa jadi signifikan lebih baik atau signifikan lebih buruk. Sedangkan perhitungan RPD hanya mengacu pada hasil terbaik pada setiap skenario. Pada tahap awal dengan jumlah customer 25, Algoritma SA lebih unggul dibanding Algoritma ACO. Terbukti menurut perhitungan RPD, 5 dari 6 skenario didapatkan solusi yang lebih baik menggunakan Algoritma SA. Dari 5 skenario tersebut, Algoritma SA signifikan lebih baik dibanding Algoritma ACO. Namun pada Skenario 25.3, tidak terdapat perbedaan signifikan antara Algoritma SA dan Algoritma ACO. Pada skenario dengan jumlah pelanggan 50, Algoritma SA dan

ACO bersaing untuk menghasilkan solusi lebih baik. Algoritma SA unggul pada 2 skenario (50.1 dan 50.5), Algoritma ACO juga unggul pada 2 skenario (50.3 dan 50.4), sedangkan untuk Skenario 50.2 dan 50.6 tidak terdapat perbedaan signifikan antara keduanya. Pada skenario dengan jumlah pelanggan lebih banyak, Algoritma ACO lebih diunggulkan untuk menghasilkan solusi lebih baik. Terbukti pada skenario dengan 100 pelanggan, Algoritma ACO mampu menghasilkan solusi yang lebih baik sebanyak 4 dari 6 skenario. Algoritma SA hanya unggul pada Skenario 100.1 dimana terdapat 1 skenario dimana tidak terdapat perbedaan signifikan antara Algoritma SA dan Algoritma ACO yaitu Skenario 100.4. Pada skenario dengan jumlah pelanggan 200, Penulis terpaksa untuk menurunkan parameter pengujian demi menghemat waktu komputasi. Skenario dengan 200 pelanggan cukup memakan waktu untuk kedua algoritma, dikarenakan membutuhkan waktu lama untuk memperoleh solusi *feasible* pada setiap iterasi. Pada skenario ini, bisa dibilang solusi dari Algoritma ACO jauh lebih baik dibanding Algoritma SA. Pada seluruh skenario dengna 200 pelanggan, solusi Algoritma ACO selalu lebih baik secara signifikan maupun hasil terbaik. Waktu komputasi Algoritma ACO pada seluruh skenario lebih lama dibandingkan Algoritma SA, wajar jika hasilnya akan lebih baik. Secara keseluruhan dapat diambil kesimpulan bahwa pada perhitungan RPD, Algoritma ACO lebih baik untuk kasus MO-VRPTW-D dibanding Algoritma SA dengan hasil yang terbaik yaitu 62,5% dari total skenario. Berikut disajikan perbandingan skenario terbaik dari masing-masing jumlah pelanggan antara *Multi-Objective Optimization* dan *Single Objective Optimization.*

Tabel 20 *Single Objective Optimization* dan *Multi-Objective Optimization* pada Skenario Terpilih

Skenario	Rata-Rata				Standar Deviasi / Rata − Rata			
	MO	Obj. 1	Obj. 2	Obj. 3	MO	Obj. 1	Obj. 2	Obj. 3
25.1	157,026	162,604	166,152	164,512	0,0079	0,0073	0,0238	0,0172
50.1	303,704	387,479	426,255	401,401	0,0102	0,0769	0,0644	0,0714
100.2	630,639	676,855	1613,8	1534,7	0,0535	0,0548	0,0496	0,0417
200.1	2066,86	2041,18	3737,64	3716,48	0,0282	0,0312	0,028	0,0239

*: nilai terkecil

Simulated Annealing
Ant Colony Optimization

BAB V
KESIMPULAN DAN SARAN

Pada bab ini akan dibahas mengenai Kesimpulan dari Laporan Tugas Akhir yang menjawab Tujuan pada Bab I serta Saran dari Penulis untuk.penelitian-penelitian selanjutnya.

5.1 Kesimpulan

Berdasarkan Tujuan yang telah dijelaskan pada Bab I, berikut Kesimpulan yang didapat dari pengerjaan Tugas Akhir.

1. Metode *Weighted-Sum* dipilih menyelesaikan kasus *multi-objective* pada *Multi-Objective Vehicle Routing Problem with Time Window and Drones* (MO-VRPTW-D) karena menghasilkan solusi lebih baik daripada Metode *Lexicographic* dengan nilai objektif sebesar 196,37
2. Algoritma *Simulated Annealing* dan *Ant Colony Optimization* berhasil dikembangkan untuk menyelesaikan kasus *Multi-Objective Vehicle Routing Problem with Time Window and Drones* (MO-VRPTW-D) dimana Algoritma *Simulated Annealing* membutuhkan bantuan Algoritma *Push-Forward Insertion Heuristic* untuk membentuk solusi awal, sedangkan Algoritma *Ant Colony Optimization* mampu menghasilkan solusi awal sendiri walaupun membutuhkan waktu komputasi yang lebih lama
3. Algoritma *Simulated Annealing* dan *Ant Colony Optimization* yang dihasilkan sudah tervalidasi sesuai dengan konstrain pada kasus *Multi-Objective Vehicle Routing Problem with Time Window and Drones* (MO-VRPTW-D) dengan nilai objektif sebesar 62,6217 (sesuai dengan metode eksak) dan untuk Algoritma *Push-Forward Insertion Heuristic* menghasilkan solusi awal sebesar 64,6116 dengan nilai *infeasibility* sebesar 0 (rute awal yang dihasilkan adalah rute yang *feasible*)
4. Dari hasil eksperimen dan perhitungan ANOVA untuk kasus *Multi-Objective Vehicle Routing Problem with Time Window and Drones* (MO-VRPTW-D), Algoritma *Simulated Annealing* (SA) mampu menghasilkan solusi lebih baik pada 8 skenario, Algoritma *Ant Colony Optimization* (ACO) mampu menghasilkan solusi lebih baik pada 12 skenario,

sedangkan untuk skenario lainnya tidak terdapat perbedaan signifikan antara kedua metode

5 Dari perhitungan *Relavite Percentage Deviation* (RPD) pada Algoritma *Simulated Annealing* (SA) dan *Ant Colony Optimization* (ACO) untuk kasus *Multi-Objective Vehicle Routing Problem with Time Window and Drones* (MO-VRPTW-D), Algoritma ACO menghasilkan solusi lebih baik dari Algoritma SA untuk 15 skenario dari total 24 skenario (62,5%), sehingga secara keseluruhan dapat disimpulkan bahwa Algoritma ACO lebih baik daripada Algoritma SA untuk kasus MO-VRPTW-D

5.2 Saran

Berikut merupakan Saran yang diberikan oleh Penulis untuk perbaikan pada penelitian selanjutnya.

1. Penelitian saat ini mengasumsikan bahwa kecepatan truk dan *drone* adalah linear, sehingga pada penelitian selanjutnya dapat dilakukan dengan memodifikasi skenario kecepatan truk dan *drone* dari skenario statis menjadi skenario dinamis
2. Penelitian selanjutnya dapat dilakukan dengan mempertimbangkan faktor utilitas muatan pada truk menggunakan *goal programming* untuk menghindari penambahan truk yang hanya untuk mengirimkan sedikit barang dan lebih memilih untuk tidak mengirimkannya dengan konsekuensinya adalah biaya penalti
3. Penelitian selanjutnya dapat dilakukan dengan menggabungkan antara skenario pelanggan horizontal dan skenario pelanggan vertikal sehingga akan mendekati kondisi *real*, namun model akan semakin kompleks seiring bertambahnya konstrain
4. Pada penelitian selanjutnya diharapkan dapat mempertimbangkan faktor durasi maksimal dari penerbangan *drone* serta *setup time drone* karena hal tersebut mempengaruhi waktu truk meninggalkan *customer-i* serta waktu truk tiba di *customer-j*
5. Penelitian selanjutnya diharapkan dapat menghasilkan *code* yang lebih efisien dalam hal waktu komputasi namun tetap menghasilkan solusi yang mendekati optimal

DAFTAR PUSTAKA

Afifi, S., Dang, D. and Moukrim, A., 2013. *A Simulated Annealing Algorithm for the Vehicle Routing Problem with Time Windows and Synchronization Constraints*. Catania: Learning and Intelligent Optimization (LION 7).

Astuti, S., 2012. *Aplikasi Algoritma Genetika Hibrida Pada Vehicle Routing Problem with Time Windows*. *Undergraduate Thesis*. Universitas Indonesia.

Ballou, R. H., 2004. *Business Logistics/Supply Chain Management*. (5th ed). s.l: Prentice Hall.

Bateman, T. & A., 2021. *Drone Delivery of Vaccine Doses Speeds Up COVID-19 Vaccinations in Remote Areas of Ghana.* [Online] Tersedia di: <https://www.euronews.com/next/2021/06/04/drone-delivery-of-vaccine-doses-speeds-up-covid-19-vaccinations-in-remote-areas-of-ghana> [Diakses 8 Agustus 2021].

Bus-Truck Indonesia. 2021. *Gabungan Pengiriman Barang dengan Van dan Drone*. [Online] Tersedia di: <https://bus-truck.id/berita/horsefly-gabungan-pengiriman-barang-dengan-van-dan-drone/8244> [Diakses 3 November 2021].

Cordeau, J., Desaulniers, G., Desrosiers, J., Solomon, M. and Soumis, F., 2000. *The VRP with Time Windows*. Philadelphia.

Corrigan, F., 2020. *Drones for Deliveries from Medicine to Post, Packages and Pizza.* [Online] Tersedia di: <https://www.dronezon.com/drones-for-good/drone-parcel-pizza-delivery-service/> [Diakses 8 Agustus 2021]

Dell'Amico, M., Montemanni, R. and Novellani, S., 2021. *Exact Models for the Flying Sidekick Traveling Salesman Problem*. *International Transactions in Operational Research*.

Ed. 2021. *Miller-Tucker-Zemlin formulation — AIMMS How-To*. [Online] Tersedia di: <https://how-to.aimms.com/Articles/332/332-Miller-Tucker-Zemlin-formulation.html> [Diakses 3 November 2021].

Ed. 2021. *Truk Mitsubishi Fuso Terbaru: Beli / Sewa*. [Online] Tersedia di: <https://www.deliveree.com/id/truk-mitsubishi-fuso-terbaru/> [Diakses 3 November 2021].

El-Sherbeny, N., 2010. *Vehicle Routing with Time Windows: An Overview of Exact, Heuristic and Metaheuristic Methods*. *Journal of King Saud University - Science*, 22(3), pp.123-131.

Eydi, A. and Ghasemi-Nezhad, S., 2021. *A Bi-Objective Vehicle Routing Problem with Time Windows and Multiple Demands*. *Ain Shams Engineering Journal*, 12(3), pp.2617-2630.

Ghoseiri, K. and F. Ghannad, S., 2008. *Hybrid Genetic Algorithm for Vehicle Routing and Scheduling Problem*. *Journal of Applied Sciences*, 9(1), pp.79-87.

Han, Y., Li, J., Liu, Z., Liu, C. and Tian, J., 2020. *Metaheuristic Algorithm for Solving the Multi-Objective Vehicle Routing Problem with Time Window and Drones*. *International Journal of Advanced Robotic Systems*, 17(2), p.172988142092003.

Inayati, S., 2020. Penyelesaian Masalah Optimisasi Multiobjektif Nonlinear Menggunakan Pendekatan Pareto Front dalam Metode Pembobotan. Jurnal Matematika Integratif, 16(2), p.139.

Iswardani, Kurnia. (2015). *Penerapan Ant Colony Optimization pada Vehicle Routing Problem Time Windows (Studi Kasus: CV Yufa Barokah). Undergraduate Thesis*, Institut Teknologi Sepuluh Nopember, Surabaya.

Kallehauge, B., 2006. *On the Vehicle Routing Problem with Time Windows*. Ph.D. Technical University of Denmark.

Kallehauge, B., Larsen, J., Madsen, O. and Solomon, M., n.d. *Vehicle Routing Problem with Time Windows*. pp.69 - 71.

Kho, J., 2021. *Mengenal Delivery Drone Untuk Keperluan Komersial Di Indonesia*. [Online] Tersedia di: <https://www.simplidots.com/mengenal-delivery-drone-untuk-keperluan-komersial-di-indonesia/> [Diakses 8 Agustus 2021].

Kitjacharoenchai, P. and Lee, S., 2019. *Vehicle Routing Problem with Drones for Last Mile Delivery*. *Procedia Manufacturing*, 39, pp.314-324.

Laporte, G. (1992). *The Vehicle Routing Problem*: *An Overview of Exact and Approximate Algorithms*. *European Journal of Operational Research*, 345-358.

Murray, C. C., & Chu, A. G. (2015). *The Flying Sidekick Traveling Salesman Problem: Optimization of Drone-Assisted Parcel Delivery*. *Transportation Research Part C*: *Emerging Technologies*, 54, 86–109.

Othman, W., Abd Wahab, A., Alhady, S. and Wong, H., 2018. *Solving Vehicle Routing Problem using Ant Colony Optimisation (ACO) Algorithm. International Journal of Research and Engineering*, 5(9), pp.500-507.

Pratama, G., Alfaridzi, E., Awaluddin, A., Mara, S. and Rifai, A., 2020. *Studi Komparasi Penggunaan Drone untuk Logistik Last-Mile*. *Seminar Nasional*.

Prawira, HI. (2020). *Pengembangan Algoritma Particle Swarm Optimization dan Simulated Annealing Untuk Menyelesaikan Vehicle Routing Problem with Drone. Thesis*, Institut Teknologi Sepuluh Nopember, Surabaya.

Rahayu, R., Novianingsih, K. and H., H., 2018. PENYELESAIAN MASALAH PENUGASAN MULTI OBJEKTIF DENGAN METODE WEIGHTED-SUM DAN METODE $\boldsymbol{\varepsilon}$-CONSTRAINT. Bandung: Departemen Pendidikan Matematika FPMIPA UPI.

Redi, A., Maula, F., Kumari, F., Syaveyenda, N., Ruswandi, N., Khasanah, A. and Kurniawan, A., 2020. *Simulated Annealing Algorithm for Solving the Capacitated Vehicle Routing Problem: A Case Study of Pharmaceutical Distribution*. *Jurnal Sistem dan Manajemen Industri*, 4(1), pp.41-49.

Rose, C. (2013). *Amazon's Jeff Bezos Looks to the Future*. Tersedia di: <https://www.cbsnews.com/news/amazons-jeff-bezos-looks-to-the-future/> [Diakses 8 Agustus 2021].

Santosa, B., 2008. MATLAB untuk Statistika & Teknik Optimasi. 1st ed. Yogyakarya: Graha Ilmu, pp.1 - 31.

Santosa, B. & Ai, T. J., 2017. *Pengantar Metaheuristik: Implementasi dengan Matlab*. (1st ed). Surabaya: ITS Tekno Sains.

Sacramento, D., Pisinger, D., & Ropke, S. (2019). *An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones*. *Transportation Research Part C: Emerging Technologies*, 102(March), 289–315. https://doi.org/10.1016/j.trc.2019.02.018

Sidik, F., 2018. *Ini Teknologi Drone JD.com Untuk Optimalisasi Pengiriman Barang.* [Online] Tersedia di: <https://teknologi.bisnis.com/read/20181118/105/861000/ini-teknologi-drone-jd.com-untuk-optimalisasi-pengiriman-barang> [Diakses 8 Agustus 2021].

Sintef.no. 2021. *100 customers*. [online] Tersedia di: <https://www.sintef.no/projectweb/top/vrptw/solomon-benchmark/100-customers/> [Diakses 3 November 2021].

Soenandi, I., Joice, J. and Marpaung, B., 2019. *Optimasi Capacitated Vehicle Routing Problem with Time Windows dengan Menggunakan Ant Colony Optimization. Jurnal Sistem dan Manajemen Industri*, 3(1), p.59.

Stanimirovic, I., 2012. *COMPENDIOUS LEXICOGRAPHIC METHOD FOR MULTI-OBJECTIVE OPTIMIZATION*. 1st ed. Niš: FACTA UNIVERSITATIS.

Tam, V. and Ma, K., 2004. *An Effective Search Framework Combining Meta-Heuristics to Effectively Solve the Vehicle Routing Problems with Time Windows*. *Artificial Intelligence Review*, 21(2), pp.87-112.

Wahyuningsih, S., Satyananda, D., Octoviana, L. and Nurhakiki, R., n.d. *Vehicle Routing Problem with Time Windows Variants and Its Application in Distribution Optimization*. Malang: Jurusan Matematika FMIPA, Universitas Negeri Malang, pp.27 - 30.

Wang, Z., & Sheu, J. B. (2019). *Vehicle routing problem with drones. Transportation Research Part B: Methodological*, 122, 350–364. https://doi.org/10.1016/j.trb.2019.03.005

Wen, T., Zhang, Z. and Wong, K., 2016. *Multi-Objective Algorithm for Blood Supply via Unmanned Aerial Vehicles to the Wounded in an Emergency Situation*. *PLOS ONE*, 11(5), p.e0155176.

LAMPIRAN

Rekap Data

https://intip.in/DataTugasAkhir/

Rekap Hasil

- *Rekap Hasil Pengujian Algoritma SA untuk MO-VRPTW-D*

1. Skenario 25.1

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	157,3707	157,09528	2,275488548
2	158,5502		
3	158,9587		
4	153,2261		
5	157,3707		

2. Skenario 25.2

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	195,4094	182,2933	27,5324
2	194,722		
3	192,7287		
4	195,5237		
5	133,0826		

3. Skenario 25.3

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	234,396	204,5996	35,4222
2	226,5098		
3	164,68		
4	167,1781		
5	230,2339		

4. Skenario 25.4

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	220,2282	220,8737	1,5036
2	221,8346		

Percobaan	Hasil	Rata-Rata	Standar Deviasi
3	222,1728		
4	221,6014		
5	218,5315		

5. Skenario 25.5

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	194,7799	195,4919	0,7278
2	194,9696		
3	196,0203		
4	196,4816		
5	195,2081		

6. Skenario 25.6

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	172,7187	170,4013	1,4654
2	170,9013		
3	169,9233		
4	169,1839		
5	169,2794		

7. Skenario 50.1

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	346,2891	303,7044	37,7594
2	281,1093		
3	267,1579		
4	342,8566		
5	281,1093		

8. Skenario 50.2

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	437,877	456,9479	32,4642
2	441,9837		
3	448,2536		
4	441,9837		

Percobaan	Hasil	Rata-Rata	Standar Deviasi
5	514,6416		

9. Skenario 50.3

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	497,3164	558,3413	35,5751
2	579,6641		
3	569,9122		
4	559,1011		
5	585,7128		

10. Skenario 50.4

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	574,2373	591,0404	37,4551
2	656,9638		
3	579,6513		
4	580,594		
5	563,7554		

11. Skenario 50.5

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	605,9836	588,7650	32,1763
2	531,322		
3	603,1515		
4	600,2164		
5	603,1515		

12. Skenario 50.6

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	500,5428	549,9742	38,7840
2	515,3566		
3	577,9793		
4	581,1753		
5	574,8168		

13. Skenario 100.1

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	689,2644	642,7628	26,5267
2	632,6604		
3	638,729		
4	624,3239		
5	628,8364		

14. Skenario 100.2

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	1523,9	1403,9800	135,0364
2	1460,9		
3	1174,1		
4	1453,6		
5	1407,4		

15. Skenario 100.3

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	1199,2	1213,8200	31,6883
2	1200,1		
3	1270,5		
4	1199,2		
5	1200,1		

16. Skenario 100.4

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	1024	1010,16	28,1181
2	1022,5		
3	1021,9		
4	959,875		
5	1022,5		

17. Skenario 100.5

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	1617,4	1594,4800	17,3129
2	1586,7		

Percobaan	Hasil	Rata-Rata	Standar Deviasi
3	1571,7		
4	1592,6		
5	1604		

18. Skenario 100.6

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	1211	1279,5600	69,6532
2	1214,6		
3	1339,1		
4	1270,7		
5	1362,4		

19. Skenario 200.1

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	4381,1	4271,8000	120,4432
2	4101,3		
3	4191,8		
4	4324,5		
5	4360,3		

20. Skenario 200.2

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	4531,5	4624,9200	327,1939
2	5075,2		
3	4432,4		
4	4254,5		
5	4831		

21. Skenario 200.3

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	8337,4	8015,5000	280,0241
2	8249,8		
3	7960,6		
4	7879,9		

Percobaan	Hasil	Rata-Rata	Standar Deviasi
5	7649,8		

22. Skenario 200.4

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	6067,3	6204,1200	200,0740
2	6111,3		
3	6166,9		
4	6118,7		
5	6556,4		

23. Skenario 200.5

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	5964,6	6315,1800	318,8544
2	6074,9		
3	6713,1		
4	6568,3		
5	6255		

24. Skenario 200.6

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	6482,6	6214,6800	339,2594
2	6581,2		
3	6213,5		
4	5734,1		
5	6062		

- *Rekap Hasil Pengujian Algoritma ACO untuk MO-VRPTW-D*

1. Skenario 25.1

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	262,7371	264,9285	2,090249766
2	266,5429		
3	267,1594		
4	265,4660		

Percobaan	Hasil	Rata-Rata	Standar Deviasi
5	262,7371		

2. Skenario 25.2

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	249,1948	248,6423	1,1278
2	249,2143		
3	248,9535		
4	249,2143		
5	246,6344		

3. Skenario 25.3

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	203,9994	195,5943	31,4022
2	207,6324		
3	139,8627		
4	212,9338		
5	213,5430		

4. Skenario 25.4

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	333,2084	358,8259	33,9144
2	395,9729		
3	334,4877		
4	395,9729		
5	334,4877		

5. Skenario 25.5

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	395,6685	376,5885	28,2842
2	395,9642		
3	389,3564		
4	373,1755		
5	328,778		

6. Skenario 25.6

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	348,8612	362,2766	29,3235
2	348,8612		
3	392,9747		
4	327,7112		
5	392,9747		

7. Skenario 50.1

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	454,9355	456,4921	4,6423
2	454,9355		
3	461,1383		
4	450,3129		
5	461,1383		

8. Skenario 50.2

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	402,5592	421,4086	35,8096
2	408,3332		
3	402,5592		
4	485,2582		
5	408,3332		

9. Skenario 50.3

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	456,0818	455,0413	3,1764
2	449,3979		
3	456,8224		
4	456,0818		
5	456,8224		

10. Skenario 50.4

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	373,408	439,3314	36,8709
2	456,9847		

Percobaan	Hasil	Rata-Rata	Standar Deviasi
3	454,6398		
4	454,6398		
5	456,9847		

11. Skenario 50.5

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	668,6675	656,7636	27,2461
2	675,8277		
3	608,6292		
4	665,3467		
5	665,3467		

12. Skenario 50.6

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	524,9554	554,5047	40,2824
2	525,1973		
3	601,1593		
4	596,014		
5	525,1973		

13. Skenario 100.1

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	834,0696	820,5700	36,8258
2	754,9069		
3	840,6048		
4	834,0696		
5	839,1993		

14. Skenario 100.2

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	619,6042	630,6393	33,7074
2	690,4779		
3	609,8409		
4	613,6694		

Percobaan	Hasil	Rata-Rata	Standar Deviasi
5	619,6042		

15. Skenario 100.3

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	747,9802	814,0572	62,0875
2	879,5533		
3	751,2487		
4	865,8706		
5	825,6332		

16. Skenario 100.4

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	942,901	1.010,7563	86,0112
2	899,684		
3	1034,088		
4	1089,101		
5	1088,009		

17. Skenario 100.5

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	1079,5	1.027,0000	51,4663
2	1015,6		
3	1006,7		
4	1076,1		
5	957,1		

18. Skenario 100.6

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	898,6	922,3000	62,9880
2	841,4		
3	907,6		
4	955,2		
5	1008,7		

19. Skenario 200.1

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	2047,1	2066,86	70,3831
2	2037,4		
3	1978,2		
4	2160,2		
5	2111,4		

20. Skenario 200.2

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	2280,7	2292,58	54,9821
2	2301,4		
3	2301,3		
4	2213		
5	2366,5		

21. Skenario 200.3

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	3247,3	3195,02	30,4589
2	3195,1		
3	3171,2		
4	3179,5		
5	3182		

22. Skenario 200.4

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	3142,1	3177,66	47,9418
2	3244		
3	3130,6		
4	3210,1		
5	3161,5		

23. Skenario 200.5

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	3137	3062,28	75,4010
2	3049,2		

Percobaan	Hasil	Rata-Rata	Standar Deviasi
3	2989,5		
4	3144,1		
5	2991,6		

24. Skenario 200.6

Percobaan	Hasil	Rata-Rata	Standar Deviasi
1	2461,8	2362,42	122,0620
2	2514,3		
3	2291,1		
4	2220,6		
5	2324,3		

LINGO *Code* untuk VRPTW

```
MODEL:
!Vehicle Routing Problem with Time Windows;

SETS:
N/1..7/:d,a,b,s;
V/1..1/:;
Link1(N,N,V):x;
Link2(N,N):t;
Link3(N,V):z,w,f,g;
ENDSETS

DATA:
Q=225;
t=
      0  22.8473  27.0000  33.5410  37.5000  30.9233        0
22.8473        0  48.8364  21.8403  48.3038  48.3038  22.8473
27.0000  48.8364        0  51.6140  30.3356  35.7805  27.0000
33.5410  21.8403  51.6140        0  37.5000  64.0800  33.5410
37.5000  48.3038  30.3356  37.5000        0  61.8466  37.5000
30.9233  48.3038  35.7805  64.0800  61.8466        0  30.9233
      0  22.8473  27.0000  33.5410  37.5000  30.9233        0;
d=0 10 7 13 19 26 0;
a=0 322 100 232 298 68 0;
b=690 513 180 378 477 132 690;
s=0 10 10 10 10 10 0;
ENDDATA

!Objective Function;
MIN=@SUM(Link1(i,j,k):t(i,j)*x(i,j,k));

!Kendaraan k dilarang dari node-i menuju node-i lagi;
@FOR(N(i):
      @FOR(V(k):
            x(i,i,k)=0
      )
```

```
);

!Kendaraan dipastikan berakhir di depot;
@FOR(N(i)|i#EQ#@SIZE(N):
	@FOR(N(j):
		@FOR(V(k):
			x(i,j,k)=0
		)
	)
);

!Jumlah kendaraan<=V;
@SUM(Link1(i,j,k)|i#EQ#1   #AND#   j#GT#1   #AND#   j#LT#@SIZE(N):
x(i,j,k))<=@SIZE(V);

!Setiap node dikunjungi sekali;
@FOR(N(i)|i#GT#1 #AND# i#LT#@SIZE(N):
	@SUM(Link1(i,j,k):x(i,j,k))=1
);

!Capacity constraints;
@FOR(V(k):
	@SUM(N(i):d(i)*@SUM(N(j):x(i,j,k)))<=Q
);

!Berangkat dan kembali ke depot;
@FOR(V(k):
	@SUM(Link1(i,j,k)|i#EQ#1:x(i,j,k))=1
);

@FOR(N(l)|l#GT#1 #AND# l#LT#@SIZE(N):
	@FOR(V(k):
		@SUM(Link1(i,l,k):x(i,l,k))=@SUM(Link1(l,j,k):
			x(l,j,k))
	)
);

@FOR(V(k):
	@SUM(Link1(i,j,k)|j#EQ#@SIZE(N):x(i,j,k))=1
);

!Time window constraints;
@FOR(N(i):
	@FOR(N(j):
		@FOR(V(k):
			x(i,j,k)*(z(i,k)+w(i,k)+s(i)+t(i,j)-z(j,k))=0
		)
	)
);

@FOR(N(i):
	@FOR(V(k):
		f(i,k)=z(i,k)+w(i,k);
		g(i,k)=z(i,k)+w(i,k)+s(i);
	)
);

@FOR(N(i):
	@FOR(V(k):
```

```
            @BND(a(i),f(i,k),b(i));
            @BND(a(i),g(i,k),b(i));
      )
);

!Binary constraints;
@FOR(N(i):
      @FOR(N(j):
            @FOR(V(k):
                  @BIN(x(i,j,k))
            )
      )
);

END
```

LINGO *Code* untuk MO-VRPTW-D

- *Weighted-Sum Method*

```
!Multi-Objective Function;
MIN=Biaya1+Biaya2+Biaya3;
Biaya1=@SUM(Link1(i,j,k)|i#NE#j:x(i,j,k)*wt(k)*t(i,j,k));
Biaya2=@SUM(Link3(i,k)|i#GT#1 #AND# i#LT#@SIZE(N):wd(k)*s(i,k));
Biaya3=@SUM(Link1(i,j,k)|i#EQ#1 #AND# j#GT#2 #AND# j#LT#@SIZE(N):
x(i,j,k));
```

- *Lexicographic Method*

1. *Objective 1*

```
!Objective 1 Function;
MIN=@SUM(Link1(i,j,k)|i#NE#j:x(i,j,k)*wt(k)*t(i,j,k));


!Subject to;

!Konsumsi energi drone;

@SUM(Link3(i,k)|i#GT#1 #AND# i#LT#@SIZE(N): wd(k)*s(i,k))>=1.11;


!#Setup kendaraan;

@SUM(Link1(i,j,k)|i#EQ#1 #AND# j#GT#2 #AND# j#LT#@SIZE(N):

x(i,j,k))>=1;
```

2. *Objective 2*

```
!Objective 2 Function;
MIN=@SUM(Link3(i,k)|i#GT#1 #AND# i#LT#@SIZE(N):wd(k)*s(i,k));


!Subject to;

!Konsumsi energi truk;

@SUM(Link1(i,j,k)|i#NE#j:x(i,j,k)*wt(k)*t(i,j,k))>=85.43;
```

```
!#Setup kendaraan;
@SUM(Link1(i,j,k)|i#EQ#1 #AND# j#GT#2 #AND# j#LT#@SIZE(N):
x(i,j,k))>=1;
```

3. *Objective 3*

```
!Objective 3 Function;
MIN=@SUM(Link1(i,j,k)|i#EQ#1 #AND# j#GT#2 #AND# j#LT#@SIZE(N):
x(i,j,k));

!Subject to;
!Konsumsi energi truk;
@SUM(Link1(i,j,k)|i#NE#j:x(i,j,k)*wt(k)*t(i,j,k))>=85.43;

!Konsumsi energi drone;
@SUM(Link3(i,k)|i#GT#1 #AND# i#LT#@SIZE(N): wd(k)*s(i,k))>=1.11;
```

- *MO-VRPTW-D dengan Weighted-Sum Method*

```
MODEL:
!Multi-Objective Vehicle Routing Problem with Time Window and
Drones;

SETS:
N/1..7/:d,a,b,h;
V/1..1/:wt,wd,qt,qd,r,vt,vd,delta;
Link1(N,N,V):x,t;
Link2(N,N):dist;
Link3(N,V):z,w,y,s,f,g;
ENDSETS

DATA:
alpha=0.6;
beta=0.1;
gamma=0.3;
delta=@OLE("E:/Teknik Industri/Semester 7/Tugas Akhir/Laporan/Fix/
Fix/Validasi.xlsx","delta");
wt=@OLE("E:/Teknik Industri/Semester 7/Tugas Akhir/Laporan/Fix/
Fix/Validasi.xlsx","wt");
wd=@OLE("E:/Teknik Industri/Semester 7/Tugas Akhir/Laporan/Fix/
Fix/Validasi.xlsx","wd");
qt=@OLE("E:/Teknik Industri/Semester 7/Tugas Akhir/Laporan/Fix/
Fix/Validasi.xlsx","qt");
qd=@OLE("E:/Teknik Industri/Semester 7/Tugas Akhir/Laporan/Fix/
Fix/Validasi.xlsx","qd");
vt=@OLE("E:/Teknik Industri/Semester 7/Tugas Akhir/Laporan/Fix/
Fix/Validasi.xlsx","vt");
vd=@OLE("E:/Teknik Industri/Semester 7/Tugas Akhir/Laporan/Fix/
Fix/Validasi.xlsx","vd");
r=@OLE("E:/Teknik Industri/Semester 7/Tugas Akhir/Laporan/Fix/
Fix/Validasi.xlsx","range");
d=@OLE("E:/Teknik Industri/Semester 7/Tugas Akhir/Laporan/Fix/
Fix/Validasi.xlsx","d");
```

```
a=@OLE("E:/Teknik  Industri/Semester  7/Tugas  Akhir/Laporan/Fix/
Fix/Validasi.xlsx","a");
b=@OLE("E:/Teknik  Industri/Semester  7/Tugas  Akhir/Laporan/Fix/
Fix/Validasi.xlsx","b");
h=@OLE("E:/Teknik  Industri/Semester  7/Tugas  Akhir/Laporan/Fix/
Fix/Validasi.xlsx","h");
dist=
0        15.2315  18.0000  22.3607  25.0000  20.6155        0
15.2315        0  32.5576  14.5602  32.2025  32.2025  15.2315
18.0000  32.5576        0  34.4093  20.2237  23.8537  18.0000
22.3607  14.5602  34.4093        0  25.0000  42.7200  22.3607
25.0000  32.2025  20.2237  25.0000        0  41.2311  25.0000
20.6155  32.2025  23.8537  42.7200  41.2311        0  20.6155
      0  15.2315  18.0000  22.3607  25.0000  20.6155        0;
ENDDATA

!Multi-Objective Function;
MIN=alpha*Biaya1+beta*Biaya2+gamma*Biaya3;
Biaya1=@SUM(Link1(i,j,k)|i#NE#j:x(i,j,k)*wt(k)*t(i,j,k));
Biaya2=@SUM(Link3(i,k)|i#GT#1 #AND# i#LT#@SIZE(N):wd(k)*s(i,k));
Biaya3=@SUM(Link1(i,j,k)|i#EQ#1 #AND# j#GT#2 #AND# j#LT#@SIZE(N):
delta(k)*x(i,j,k));

!Kendaraan k dilarang dari node-i menuju node-i lagi;
@FOR(N(i):
      @FOR(V(k):
            x(i,i,k)=0
      )
);

!Kendaraan dipastikan berakhir di depot;
@FOR(N(i)|i#EQ#@SIZE(N):
      @FOR(N(j):
            @FOR(V(k):
                  x(i,j,k)=0
            )
      )
);

!Jumlah kendaraan<=V;
@SUM(Link1(i,j,k)|i#EQ#1   #AND#   j#GT#1   #AND#   j#LT#@SIZE(N):
x(i,j,k))<=@SIZE(V);

!Capacity constraints;
@FOR(V(k):
      @SUM(Link3(i,k):d(i)*y(i,k))<=qt(k)
);

!Setiap node dikunjungi sekali;
@FOR(N(i)|i#GT#1 #AND# i#LT#@SIZE(N):
      @SUM(Link1(i,j,k):x(i,j,k))=1
);

@FOR(N(i)|i#GT#1 #AND# i#LT#@SIZE(N):
      @SUM(Link3(i,k):y(i,k))=1
);

@FOR(V(k):
      @FOR(N(j)|j#GT#1 #AND# j#LT#@SIZE(N):
```

```
            @SUM(Link1(i,j,k):x(i,j,k))=y(j,k)
      )
);

@FOR(V(k):
      @FOR(N(i)|i#GT#1 #AND# i#LT#@SIZE(N):
            @SUM(Link1(i,j,k):x(i,j,k))=y(i,k)
      )
);

!Berangkat dan kembali ke depot;
@FOR(V(k):
      @SUM(Link1(i,j,k)|i#EQ#1:x(i,j,k))=1
);

@FOR(N(l)|l#GT#1 #AND# l#LT#@SIZE(N):
      @FOR(V(k):
            @SUM(Link1(i,l,k):x(i,l,k))=@SUM(Link1(l,j,k):
                  x(l,j,k))
      )
);

@FOR(V(k):
      @SUM(Link1(i,j,k)|j#EQ#@SIZE(N):x(i,j,k))=1
);

!Time window constraints;
@FOR(N(i):
      @FOR(N(j):
            @FOR(V(k):
                  t(i,j,k)=dist(i,j)/vt(k)
            )
      )
);

@FOR(N(i):
      @FOR(V(k):
            s(i,k)=@ROUNDUP(d(i)/qd(k),0)*(2*h(i)*y(i,k)/vd(k))
      )
);

@FOR(V(k):
      @SUM(Link1(i,j,k):(x(i,j,k)*(w(i,k)+s(i,k)+t(i,j,k))))<=r(k)
);

@FOR(N(i):
      @FOR(N(j):
            @FOR(V(k):
                  x(i,j,k)*(z(i,k)+w(i,k)+s(i,k)+t(i,j,k)-
                        z(j,k))=0
            )
      )
);

@FOR(N(i):
      @FOR(V(k):
            f(i,k)=z(i,k)+w(i,k);
            g(i,k)=z(i,k)+w(i,k)+s(i,k);
      )
```

```
);

@FOR(N(i):
	@FOR(V(k):
		@BND(a(i),f(i,k),b(i));
		@BND(a(i),g(i,k),b(i));
	)
);

!Binary constraints;
@FOR(N(i):
	@FOR(N(j):
		@FOR(V(k):
			@BIN(x(i,j,k))
		)
	)
);

@FOR(N(i):
	@FOR(V(k):
		@BIN(y(i,k))
	)
);

END
```

MATLAB *Function*

- *feasibilitas.m*

```
function
[rute,total_time,s,t,infeasibility,infeas1,infeas2,infeas3,infeas4]=feas
ibilitas(xy,rute,kendaraan,kota)
d=squareform(pdist(xy));
[nveh,~]=size(kendaraan);
[ncit,~]=size(kota);

% Infeasibility time
%travel time all nodes k
for i=1:ncit
    for j=1:ncit
        for k=1:nveh
            t(i,j,k)=60*d(i,j)/kendaraan(k,3);
        end
    end
end

%travel time each route
time=0;
for k=1:nveh
    for i=1:ncit
        while i<=ncit
            if rute(k,i+1)==0
                rute(k,i+1)=rute(k,i);
            else
                time(k,i)=t(rute(k,i),rute(k,i+1),k);
            end
```

```
            i=i+1;
        end
    end
end

%normalisasi rute
rute1=norm_rute(rute);

%serving time
s=zeros(nveh,ncit+1);
h=kota(:,4);
D=kota(:,1);
for k=1:nveh
    for i=1:ncit+1
        if rute1(k,i)==0
            s(k,i)=0;
        else

s(k,i)=(ceil(D(rute1(k,i))./kendaraan(k,2)))*(2*h(rute1(k,i)).*sign(rute
1(k,i))./kendaraan(k,4));
        end
    end
end

%time window
first_tw=reshape(kota(rute(:,:),2),nveh,ncit+1);
arrival=zeros(nveh,ncit+1);
leave=zeros(nveh,ncit+1);
for k=1:nveh
    i=1;
    while i<=ncit+1
        if i==1
            arrival(k,i)=0;
            leave(k,i)=max(first_tw(k,i+1),time(k,i))-time(k,i);
        elseif i==2
            arrival(k,i)=max(first_tw(k,i),time(k,i-1));
            leave(k,i)=arrival(k,i)+s(k,i);
        else
            arrival(k,i)=max(first_tw(k,i),arrival(k,i-1)+s(k,i-
1)+time(k,i-1));
            leave(k,i)=arrival(k,i)+s(k,i);
        end
        i=i+1;
    end
end

% Infeasibility capacity
%truck load
load_t=0;
for k=1:nveh
    for i=1:ncit
        if rute1(k,i)==0
            load_t(k,i)=0;
        else
            load_t(k,i)=kota(rute1(k,i),1);
        end
    end
```

```
end

% Infeasibility
%infeasibility capacity
infeas1=sum(find(kendaraan(:,1)-sum(load_t')')<0);

%infeasibility time window
last_tw=reshape(kota(rute(:,:),3),nveh,ncit+1);
infeas2=sum(sum(leave-last_tw>0));

%infeasibility serving time
infeas3=sum(sum(s(:,1:ncit)-(kota(:,3)-kota(:,2))'>0));

%infeasibility range for each vehicle
total_time=max(leave')'-min(leave')';
infeas4=sum(find(kendaraan(:,5)-total_time<0));

% Infeasibility
infeasibility=infeas1+infeas2+infeas3+infeas4;

end
```

- *norm_rute.m*

```
function rute1=norm_rute(rute)
rute1=rute;
[panjang,lebar]=size(rute1);
y=zeros(panjang,lebar-1);
for k=1:panjang
    y(k,:)=[diff(rute1(k,:)')'];
    rute1(k,find(y(k,1:end-1)==0)+1)=0;
end
end
```

- *norm_rute2.m*

```
function rute2=norm_rute2(rute)
rute2=rute;
[nveh,ncit]=size(rute2);
for i=2:ncit-2
    for j=1:nveh
        if rute2(j,i)==0
            [~,x]=find(rute2(j,i:ncit-1)~=0);
            x=x+i-1;
            rute2(j,[i min(x)])=rute2(j,[min(x) i]);
        end
    end
end
y=find(sum(rute2)==0);
rute2(:,y)=[];
end
```

- *norm_rute3.m*

```
function rute3=norm_rute3(rute,kendaraan,kota)
[nveh,~]=size(kendaraan);
[ncit,~]=size(kota);
```

```
rute3=[ones(nveh,1) zeros(nveh,ncit-1) ones(nveh,1)];
rute=[1 rute];
rute(rute==1)=0;
x=find(rute==0);
for i=1:nveh
    if i==nveh
        x(i+1)=length(rute);
        rute3(i,2:(x(i+1)-x(i)+1))=rute(x(i)+1:x(i+1));
    else
        rute3(i,2:(x(i+1)-x(i)))=rute(x(i)+1:x(i+1)-1);
    end
end
end
```

- *norm_rute4.m*

```
function rute4=norm_rute4(tau,xy,kendaraan,kota)
[ncit,~]=size(kota);
[nveh,~]=size(kendaraan);
rute4=zeros(nveh,ncit+1);
rute4(:,[1 end])=1;
i=1;
j=1;
idk=1;
tau(:,1,:)=0;
while j<=nveh
    while i<=ncit-1
        if sum(sum(tau(:,:,j)))==0
            i=ncit;
        else
            [~,idk]=max(tau(idk,:,j));
            rute4(j,i+1)=idk;

[~,~,~,~,infeasibility,~,~,~,~]=feasibilitas(xy,rute4,kendaraan,kota);
            if infeasibility>0
                rute4(j,i+1)=0;
                tau(:,idk,j)=0;
            else
                tau(:,idk,:)=0;
                i=i+1;
            end
        end
    end
    if j==1
        i=min(find(rute4(j,2:end)==0));
    else
        idk_i=max(find(rute4(j,2:ncit)~=0));
        i=idk_i+1;
    end
    j=j+1;
end
end
```

- *biaya.m*

```
function [rute,E]=biaya(xy,wt,wd,rute,kendaraan,kota,delta)
[nveh,~]=size(kendaraan);
[ncit,~]=size(kota);
```

```
alpha=.6;
beta=.1;
gamma=.3;
[rute,~,s,t,~]=feasibilitas(xy,rute,kendaraan,kota);

%decision variables
x=zeros(ncit,ncit,nveh);
for i=1:ncit
    for j=1:ncit
        for k=1:nveh
            x(rute(k,i),rute(k,i+1),k)=1;
        end
    end
end
for i=1:ncit
    for j=1:ncit
        for k=1:nveh
            if i==j
                x(i,j,k)=0;
            end
        end
    end
end

%konsumsi energi truk
cons=sum(sum(x.*t));
[panjang,~]=size(rute);
cons=reshape(cons,panjang,1,1);
biaya1=sum(wt.*cons);

%konsumsi energi drone
biaya2=sum(wd.*sum(s')');

%setup cost set kendaraan
biaya3=sum(delta.*sign(reshape(sum(sum(x)),panjang,1)));

%total biaya
E=alpha*biaya1+beta*biaya2+gamma*biaya3;

end
```

- *PFIH_Iter.m*

```
function [rute0,biaya]=PFIH_iter(xy,wt,wd,delta,kota,kendaraan,it_PFIH)
[rute1,~,biaya1]=PFIH_MOVRPTWD(xy,wt,wd,delta,kota,kendaraan,it_PFIH);
[rute2,~,biaya2]=PFIH_MOVRPTWD(xy,wt,wd,delta,kota,kendaraan,it_PFIH);
[rute3,~,biaya3]=PFIH_MOVRPTWD(xy,wt,wd,delta,kota,kendaraan,it_PFIH);
[rute4,~,biaya4]=PFIH_MOVRPTWD(xy,wt,wd,delta,kota,kendaraan,it_PFIH);
[rute5,~,biaya5]=PFIH_MOVRPTWD(xy,wt,wd,delta,kota,kendaraan,it_PFIH);
[rute6,~,biaya6]=PFIH_MOVRPTWD(xy,wt,wd,delta,kota,kendaraan,it_PFIH);
[rute7,~,biaya7]=PFIH_MOVRPTWD(xy,wt,wd,delta,kota,kendaraan,it_PFIH);
[rute8,~,biaya8]=PFIH_MOVRPTWD(xy,wt,wd,delta,kota,kendaraan,it_PFIH);
[rute9,~,biaya9]=PFIH_MOVRPTWD(xy,wt,wd,delta,kota,kendaraan,it_PFIH);
[rute10,~,biaya10]=PFIH_MOVRPTWD(xy,wt,wd,delta,kota,kendaraan,it_PFIH);
[nveh1,~]=size(rute1);
[nveh2,~]=size(rute2);
[nveh3,~]=size(rute3);
```

```
[nveh4,~]=size(rute4);
[nveh5,~]=size(rute5);
[nveh6,~]=size(rute6);
[nveh7,~]=size(rute7);
[nveh8,~]=size(rute8);
[nveh9,~]=size(rute9);
[nveh10,~]=size(rute10);
nvehx=[nveh1 nveh2 nveh3 nveh4 nveh5 nveh6 nveh7 nveh8 nveh9 nveh10];
[~,idknveh]=min(nvehx);
if idknveh==1
    rute0=rute1;
    biaya=biaya1;
elseif idknveh==2
    rute0=rute2;
    biaya=biaya2;
elseif idknveh==3
    rute0=rute3;
    biaya=biaya3;
elseif idknveh==4
    rute0=rute4;
    biaya=biaya4;
elseif idknveh==5
    rute0=rute5;
    biaya=biaya5;
elseif idknveh==6
    rute0=rute6;
    biaya=biaya6;
elseif idknveh==7
    rute0=rute7;
    biaya=biaya7;
elseif idknveh==8
    rute0=rute8;
    biaya=biaya8;
elseif idknveh==9
    rute0=rute9;
    biaya=biaya9;
else
    rute0=rute10;
    biaya=biaya10;
end
end
```

MATLAB *Code* untuk Inisialisasi Solusi Awal Menggunakan Algoritma PFIH

```
function
[rute0,infeasibility,E0]=PFIH_MOVRPTWD(xy,wt,wd,delta,kota,kendaraan,itm
ax)
a1=.7; %alpha
a2=.1; %beta
a3=.2; %gamma
d=squareform(pdist(xy,'euclidean')); %Jarak antar customer dalam mil
D=kota(:,1); %Demand tiap customer
a=kota(:,2); %Jam buka tiap customer
b=kota(:,3); %Jam tutup tiap customer
h=kota(:,4); %Ketinggian tiap customer
[ncit,~]=size(kota);
[nveh,~]=size(kendaraan);
```

```
rute_now=[ones(nveh,1,itmax) zeros(nveh,ncit-1,itmax)
ones(nveh,1,itmax)];
it=1;
while it<=itmax
    kendaraan0=kendaraan(1,:);
    rute0=[1 zeros(1,ncit-1) 1];
    for i=2:ncit %biaya penyisipan
        cos_theta(i)=(xy(i,1)-xy(1,1))/d(1,i);
        c(i)=-a1*(d(1,i))+a2*b(i)+a3*(acosd(cos_theta(i))/360)*(d(1,i));
    end
    cos_theta=cos_theta(2:end);
    c=c(2:end);
    [~,idk]=sort(c);
    rute0(1,2:ncit)=idk+1;
    [rute0,~,~,~,infeasibility]=feasibilitas(xy,rute0,kendaraan0,kota);
    clear z
    z=2;
    while infeasibility>0
        rute0(z,:)=[1 zeros(1,ncit-1) 1];
        kendaraan1=kendaraan(1:z,:);
        qt=kendaraan1(:,1); %Kapasitas truk
        qd=kendaraan1(:,2); %Kapasitas drone
        vt=kendaraan1(:,3); %Kecepatan truk
        vd=kendaraan1(:,4); %Kecepatan drone
        r=kendaraan1(:,5); %Batas total waktu maksimum truk
        [~,~,~,~,infeasibility1]=feasibilitas(xy,rute0(z-
1,:),kendaraan1(z-1,:),kota);
        while infeasibility1>0
            rute_inf=find(rute0(z-1,2:end-1)~=0)+1;
            rute_rot=randperm(length(rute_inf));
            rute0([z-1 z],rute_inf(rute_rot(1)))=rute0([z z-
1],rute_inf(rute_rot(1)));
            [rute0(z-1,:),~,~,~,infeasibility1]=feasibilitas(xy,rute0(z-
1,:),kendaraan1(z-1,:),kota);
            rute0(z-1,:)=norm_rute(rute0(z-1,:));
        end

[rute0,~,~,~,infeasibility]=feasibilitas(xy,rute0,kendaraan1,kota);
        rute0=norm_rute(rute0);
        z=z+1;
    end
    [rute0,E0(it)]=biaya(xy,wt(1:z-1),wd(1:z-1),rute0,kendaraan(1:z-
1,:),kota,delta(1:z-1));
    rute0=norm_rute(rute0);
    if it==1
        [nveh1,~]=size(rute0);
        rute_now(1:nveh1,:,it)=rute0;
    else
        if sum(sum(rute_now(:,:,it)')'~=2)<sum(sum(rute_now(:,:,it-
1)')'~=2)
            [nveh1,~]=size(rute0);
            rute_now(1:nveh1,:,it)=rute0;
        else
            if E0(it)>E0(it-1)
                E0(it)=E0(it-1);
                rute_now(:,:,it)=rute_now(:,:,it-1);
            else
```

```
                [nveh1,~]=size(rute0);
                rute_now(1:nveh1,:,it)=rute0;
            end
        end
    end
    clear rute0 nveh1
    it=it+1;
end
rute0=rute_now(:,:,it-1);
rute0((sum(rute0')'==2),:)=[];
E0=E0(it-1);
end
```

MATLAB *Code* untuk MO-VRPTW-D Menggunakan Algoritma SA

```
function [rute,E_SA]=SA_MOVRPTWD(xy,wt,wd,delta,kota,kendaraan,c,itmax)
d=squareform(pdist(xy,'euclidean')); %Jarak antar customer dalam mil
D=kota(:,1); %Demand tiap customer
a=kota(:,2); %Jam buka tiap customer
b=kota(:,3); %Jam tutup tiap customer
h=kota(:,4); %Ketinggian tiap customer
qt=kendaraan(:,1); %Kapasitas truk
qd=kendaraan(:,2); %Kapasitas drone
vt=kendaraan(:,3); %Kecepatan truk
vd=kendaraan(:,4); %Kecepatan drone
r=kendaraan(:,5); %Batas total waktu maksimum truk
it_PFIH=1;
it=1;
itcon=10;
n=1;
nmax=10;
E_SA=zeros(itmax,1);
[rute0,E_SA(1,1)]=PFIH_iter(xy,wt,wd,delta,kota,kendaraan,it_PFIH);
[ncit,~]=size(kota);
[nveh,~]=size(rute0);
rute=zeros(nveh,ncit+1,itmax);
deltaE=10;
rute(:,:,1)=rute0;
T0=E_SA(1,1);
T=T0;
while it<=itmax
    if deltaE>1e-8
        while n<=nmax
            rute1=zeros(nveh,ncit+1);
            rute2=rute1;
            if it==1
                rute(:,:,it)=rute0;
            else
                rute(:,:,it)=rute(:,:,it-1);
            end
            r1=rand; %flip/swap/slide customer
            rute1=rute(:,:,it);
            [r,c]=size(rute1);
            rute1=reshape((rute1(:,2:end))',1,r*(c-1));
            rute1=norm_rute2(rute1);
            rute1(rute1==0)=[];
            rute1(end)=[];
```

```
            r11=sort(ceil((ncit-1)*rand(1,2)));
            I=r11(1);
            J=r11(2);
            if r1<=.33
                rute1(I:J)=fliplr(rute1(I:J)); %flip customer
            elseif r1<=.67 && r1>.33
                rute1([I J])=rute1([J I]); %swap customer
            else
                rute1([I:J])=rute1([(I+1):J I]); %slide customer
            end
            rute1=norm_rute3(rute1,kendaraan(1:nveh,:),kota);

[rute2,~,~,~,infeasibility]=feasibilitas(xy,rute1,kendaraan(1:nveh,:),ko
ta);
            if infeasibility==0
                rute2=norm_rute(rute2);

[~,E_SA1(it,1)]=biaya(xy,wt(1:nveh,:),wd(1:nveh,:),rute2,kendaraan(1:nve
h,:),kota,delta(1:nveh,:));
                if E_SA1(it,1)<E_SA(it,1)
                    rute(:,:,it)=rute2;
                    E_SA(it,1)=E_SA1(it,1);
                else
                    r=rand;
                    if r<=exp((E_SA(it,1)-E_SA1(it,1))/T) %kriteria
metropolis
                        rute(:,:,it)=rute2;
                        E_SA(it,1)=E_SA1(it,1);
                    end
                end
            end
            n=n+1;
        end
    else
        break
    end
    if it<itcon
        deltaE=10;
    else
        deltaE=std(E_SA((it-(itcon-1)):it,1));
    end
    T=c*T;
    n=1;
    it=it+1;
end
for j=1:it-1

[rute1(:,:,j),E_SA(j,1)]=biaya(xy,wt(1:nveh,:),wd(1:nveh,:),rute(:,:,j),
kendaraan(1:nveh,:),kota,delta(1:nveh,:));
end
E_SA(E_SA==0)=[];
E_SA=E_SA(it-1,1);
rute=rute(:,:,it-1);
rute=norm_rute(rute);
rute=norm_rute2(rute);
rute(sum(rute')==2,:)=[];
end
```

MATLAB *Code* untuk MO-VRPTW-D Menggunakan Algoritma ACO

```
function
[rute,E_ACO]=ACO_MOVRPTWD(xy,wt,wd,delta,kota,kendaraan,N,itmax)
d=squareform(pdist(xy,'euclidean')); %Jarak antar customer dalam mil
D=kota(:,1); %Demand tiap customer
a=kota(:,2); %Jam buka tiap customer
b=kota(:,3); %Jam tutup tiap customer
height=kota(:,4); %Ketinggian tiap customer
qt=kendaraan(:,1); %Kapasitas truk
qd=kendaraan(:,2); %Kapasitas drone
vt=kendaraan(:,3); %Kecepatan truk
vd=kendaraan(:,4); %Kecepatan drone
r=kendaraan(:,5); %Batas total waktu maksimum truk
it=1;
[ncit,~]=size(kota);
[nveh,~]=size(kendaraan);
rute=zeros(nveh,ncit+1,N,itmax);
rute(:,[1 end],:,:)=1;
iter=ncit-2;
alpha=1;
beta=3;
rho=.5;
tau=.01*ones(ncit,ncit,nveh);
E_ACO=zeros(N,itmax);
while it<=itmax
    for i=1:ncit
        for j=1:ncit
            for k=1:nveh
                for l=1:N
                    if d(i,j)==0
                        h(i,j,k,l)=0; %visibilitas
                    else
                        h(i,j,k,l)=1/d(i,j);
                    end
                end
            end
        end
    end
    h(:,1,:,:)=0;
    depot=zeros(N,ncit);
    for l=1:N %probabilitas transisi dari kota pertama

depot(l,:)=((tau(1,:,1).^alpha).*(h(1,:,1,l).^beta))./sum(((tau(1,:,1).^
alpha).*(h(1,:,1,l).^beta))');
    end
    depot=cumsum(depot')';
    r=rand(N,1);
    city1=zeros(N,1);
    for l=1:N
        for i=1:ncit-1
            if r(l)<=depot(l,i+1)
                city1(l,1)=i+1;
                break
            end
        end
    end
```

```
    rute1=[ones(N,1) zeros(N,ncit-1) ones(N,1)];
    rute1(:,2)=[city1];
    infeasibility1=zeros(N,1);
    for l=1:N

[~,~,~,~,infeasibility1(l,1),~,~,~,~]=feasibilitas(xy,rute1(l,:),kendara
an(1,:),kota);
    end
    while sum(infeasibility1)>0
        idk1=find(infeasibility1~=0);
        r(idk1,1)=rand(sum(infeasibility1~=0),1);
        for l=[idk1']
            for i=1:ncit-1
                if r(l)<=depot(l,i+1)
                    city1(l,1)=i+1;
                    break
                end
            end
        end
        rute1(:,2)=[city1];
        for l=1:N

[~,~,~,~,infeasibility1(l,1),~,~,~,~]=feasibilitas(xy,rute1(l,:),kendara
an(1,:),kota);
        end
    end
    for l=1:N
        rute(1,:,l,it)=rute1(l,:);
    end
    for l=1:N
        h(:,rute1(l,2),1,l)=0;
    end
    nveh0=ones(N,1);
    while iter>0
        rw=zeros(N,ncit);
        for l=1:N %probabilitas transisi
            if sum(sum(h(:,:,nveh0(l,1),l)')')==0
                nveh0(l,1)=nveh0(l,1)+1;

rw(l,:)=((tau(1,:,nveh0(l,1)).^alpha).*(h(1,:,nveh0(l,1),l).^beta))./sum
(((tau(1,:,nveh0(l,1)).^alpha).*(h(1,:,nveh0(l,1),l).^beta))');
            else
                if rute(nveh0(l,1),ncit-iter,l,it)==0

rw(l,:)=((tau(1,:,nveh0(l,1)).^alpha).*(h(1,:,nveh0(l,1),l).^beta))./sum
(((tau(1,:,nveh0(l,1)).^alpha).*(h(1,:,nveh0(l,1),l).^beta))');
                else
                    rw(l,:)=((tau(rute(nveh0(l,1),ncit-
iter,l,it),:,nveh0(l,1)).^alpha).*(h(rute(nveh0(l,1),ncit-
iter,l,it),:,nveh0(l,1),l).^beta))./sum(((tau(rute(nveh0(l,1),ncit-
iter,l,it),:,nveh0(l,1)).^alpha).*(h(rute(nveh0(l,1),ncit-
iter,l,it),:,nveh0(l,1),l).^beta))');
                end
            end
        end
        rw=cumsum(rw')';
        r=rand(N,1);
```

```
        city=zeros(N,1);
        for l=1:N
            for i=1:ncit-1
                if r(l)<=rw(l,i+1)
                    city(l,1)=i+1;
                    break
                end
            end
        end
        for l=1:N
            rute(nveh0(l,1),ncit-iter+1,l,it)=city(l,1);
        end
        infeasibility=zeros(N,1);
        for l=1:N

[~,~,~,~,infeasibility(l,1),~,~,~,~]=feasibilitas(xy,rute(:,:,l,it),kend
araan,kota);
        end
        while sum(infeasibility)>0
            rw=zeros(N,ncit);
            for l=1:N %probabilitas transisi
                if sum(sum(h(:,:,nveh0(l,1),l)')')==0
                    nveh0(l,1)=nveh0(l,1)+1;

rw(l,:)=((tau(1,:,nveh0(l,1)).^alpha).*(h(1,:,nveh0(l,1),l).^beta))./sum
(((tau(1,:,nveh0(l,1)).^alpha).*(h(1,:,nveh0(l,1),l).^beta))');
                else
                    if rute(nveh0(l,1),ncit-iter,l,it)==0

rw(l,:)=((tau(1,:,nveh0(l,1)).^alpha).*(h(1,:,nveh0(l,1),l).^beta))./sum
(((tau(1,:,nveh0(l,1)).^alpha).*(h(1,:,nveh0(l,1),l).^beta))');
                    else
                        rw(l,:)=((tau(rute(nveh0(l,1),ncit-
iter,l,it),:,nveh0(l,1)).^alpha).*(h(rute(nveh0(l,1),ncit-
iter,l,it),:,nveh0(l,1),l).^beta))./sum(((tau(rute(nveh0(l,1),ncit-
iter,l,it),:,nveh0(l,1)).^alpha).*(h(rute(nveh0(l,1),ncit-
iter,l,it),:,nveh0(l,1),l).^beta))');
                    end
                end
            end
            rw=cumsum(rw')';
            idk=find(infeasibility~=0);
            r(idk,1)=rand(sum(infeasibility~=0),1);
            for l=1:length(idk)
                for i=1:ncit-1
                    if r(idk(l))<=rw(idk(l),i+1)
                        city(idk(l),1)=i+1;
                        break
                    end
                end
            end
            for l=1:N
                rute(nveh0(l,1),ncit-iter+1,l,it)=city(l,1);
            end
            for l=1:N
```

```
[~,~,~,~,infeasibility(l,1),~,~,~,~]=feasibilitas(xy,rute(:,:,l,it),kend
araan,kota);
            end
            if sum(infeasibility)>0
                idk=find(infeasibility~=0);
                clear infeas_veh infeas_veh1 rute2 rb_kosong rute_belum
                rute2=zeros(N,ncit+1);
                rb_kosong=zeros(length(idk),1);
                for l=1:length(idk)

rb_kosong(l,1)=length([find(sum(h(:,:,nveh0(l,1),idk(l)))~=0)]);
                end
                rute_belum=zeros(length(idk),max(rb_kosong));
                for l=1:N
                    rute2(l,:)=rute(nveh0(l,1),:,l,it);
                end
                for l=1:length(idk)

rute_belum(l,1:length([find(sum(h(:,:,nveh0(l,1),idk(l)))~=0)]))=[find(s
um(h(:,:,nveh0(l,1),idk(l)))~=0)];
                end
                [~,col_rb]=size(rute_belum);
                if col_rb>0
                    if col_rb==1
                        idk2=find(rute_belum==0);
                        sum_rb=rute_belum==0;
                    else
                        idk2=find(sum(rute_belum')'==0);
                        sum_rb=sum(rute_belum')'==0;
                    end
                    if sum(sum_rb)>0
                        for l=1:length(idk2)

h(:,:,nveh0(idk(idk2(l)),1),[idk(idk2(l))])=0; %masalah utama
                            rute([nveh0(idk(idk2(l)),1)
nveh0(idk(idk2(l)),1)+1],ncit-
iter+1,idk(idk2(l)),it)=rute([nveh0(idk(idk2(l)),1)+1
nveh0(idk(idk2(l)),1)],ncit-iter+1,idk(idk2(l)),it);

nveh0(idk(idk2(l)),1)=nveh0(idk(idk2(l)),1)+1;
                        end
                        rute_belum([idk2],:)=[];
                        idk([idk2],:)=[];
                    end
                    clear r_zero c_zero
                    [r_zero,c_zero]=find(rute_belum==0);
                    for i=1:length(r_zero)

rute_belum(r_zero(i),c_zero(i))=rute_belum(r_zero(i),c_zero(i)-1);
                        while rute_belum(r_zero(i),c_zero(i))==0

rute_belum(r_zero(i),c_zero(i))=rute_belum(r_zero(i),c_zero(i)-1);
                        end
                    end
                    for l=1:length(idk)
                        for i=1:col_rb
```

```
                            if rute_belum(l,i)~=0
                                rute2(idk(l),ncit-
iter+1)=rute_belum(l,i);

[~,~,~,~,infeas_veh(i),~,~,~,~]=feasibilitas(xy,rute2(idk(l),:),kendaraa
n(nveh0(l,1),:),kota);
                            end
                        end
                    end
                end
                exist infeas_veh;
                if ans==1
                    infeas_veh1=infeas_veh==0;
                    if sum(infeas_veh1)==0
                        for l=1:length(idk)
                            h(:,:,nveh0(idk(l),1),[idk(l)])=0;
                            rute([nveh0(idk(l),1)
nveh0(idk(l),1)+1],ncit-iter+1,idk(l),it)=rute([nveh0(idk(l),1)+1
nveh0(idk(l),1)],ncit-iter+1,idk(l),it);
                            nveh0(idk(l),1)=nveh0(idk(l),1)+1;
                            for k=1:N

[~,~,~,~,infeasibility(k,1),~,~,~,~]=feasibilitas(xy,rute(:,:,k,it),kend
araan,kota);
                            end
                        end
                        for i=1:N
                            if infeasibility(i,1)==0
                                h(:,rute(nveh0(i,1),ncit-
iter+1,i,it),:,i)=0;
                            else
                                if iter==1
                                    h(:,rute(nveh0(i,1),ncit-
iter+1,i,it),:,i)=0;
                                end
                            end
                        end
                    else
                        for l=1:length(idk)
                            h(:,:,nveh0(idk(l),1),[idk(l)])=0;
                            rute([nveh0(idk(l),1)
nveh0(idk(l),1)+1],ncit-iter+1,idk(l),it)=rute([nveh0(idk(l),1)+1
nveh0(idk(l),1)],ncit-iter+1,idk(l),it);
                            nveh0(idk(l),1)=nveh0(idk(l),1)+1;
                            for k=1:N

[~,~,~,~,infeasibility(k,1),~,~,~,~]=feasibilitas(xy,rute(:,:,k,it),kend
araan,kota);
                            end
                        end
                    end
                else
                    for l=1:length(idk)
                        h(:,:,nveh0(idk(l),1),[idk(l)])=0;
                        rute([nveh0(idk(l),1) nveh0(idk(l),1)+1],ncit-
iter+1,idk(l),it)=rute([nveh0(idk(l),1)+1 nveh0(idk(l),1)],ncit-
iter+1,idk(l),it);
```

```
                        nveh0(idk(l),1)=nveh0(idk(l),1)+1;
                        for k=1:N

[~,~,~,~,infeasibility(k,1),~,~,~,~]=feasibilitas(xy,rute(:,:,k,it),kend
araan,kota);
                        end
                    end
                end
            end
        end
        for i=1:ncit
            for j=1:nveh
                for k=1:N
                    if rute(j,i,k,it)~=0
                        h(:,rute(j,i,k,it),:,k)=0;
                    end
                end
            end
        end
        iter=iter-1;
    end
    for i=1:N

[~,E_ACO(i,it)]=biaya(xy,wt(1:nveh,:),wd(1:nveh,:),rute(:,:,i,it),kendar
aan(1:nveh,:),kota,delta(1:nveh,:));
    end
    for i=1:ncit
        for j=1:ncit
            for k=1:nveh
                tau(i,j,k)=tau(i,j,k).*(1-rho);
            end
        end
    end
    delta_tau=zeros(ncit,ncit,nveh);
    for l=1:N
        rute1=norm_rute2(rute(:,:,l,it));
        [rr1,cr1]=size(rute1);
        for x=1:rr1
            for y=1:cr1-1
                if rute1(x,y)==0
                    continue
                elseif rute1(x,y+1)==0

delta_tau(rute1(x,y),1,x)=delta_tau(rute1(x,y),1,x)+1./E_ACO(l,it);
                else

delta_tau(rute1(x,y),rute1(x,y+1),x)=delta_tau(rute1(x,y),rute1(x,y+1),x
)+1./E_ACO(l,it);
                end
            end
        end
        clear rute1 rr1 cr1
    end
    for i=1:ncit
        for j=1:ncit
            for k=1:nveh
                if i==j
```

```
                    delta_tau(i,j,k)=0;
                end
            end
        end
    end
    tau=tau+delta_tau; %update pheromone
    iter=ncit-2;
    it=it+1;
end
rute4=norm_rute4(tau,xy,kendaraan(1:nveh,:),kota);
[~,E_ACO4]=biaya(xy,wt(1:nveh,:),wd(1:nveh,:),rute4,kendaraan(1:nveh,:),
kota,delta(1:nveh,:));
if E_ACO4<min(min(E_ACO))
    E_ACO=E_ACO4;
    rute=rute4;
else
    [N_idk,it_idk]=find(E_ACO==min(min(E_ACO)));
    E_ACO=E_ACO(N_idk(1),it_idk(1));
    rute=rute(:,:,N_idk(1),it_idk(1));
end
rute=norm_rute2(rute);
rute(sum(rute')==2,:)=[];
end
```

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Penulis memiliki nama lengkap Meidani Nuzul Tri Pamungkas (lahir di Kabupaten Tuban, 2 Mei 2000) adalah Mahasiswa Program Sarjana (S1) Departemen Teknik Sistem dan Industri – ITS Angkatan 2018. Penulis adalah anak dari pasangan Lilik Guntomo (Bapak) dan Ninik Suntari (Ibu) dengan status sebagai anak ketiga dari tiga bersaudara. Penulis menempuh pendidikan formal di SDN Bektiharjo 5 (2006 – 2012), SMP Negeri 1 Tuban (2012 – 2015), SMA Negeri 1 Tuban (2015 – 2018), dan Departemen Teknik Sistem dan Industri – ITS (2018 – 2022). Semasa pendidikan dasar dan menengah, Penulis aktif dalam kegiatan organisasi, kepanitiaan, dan perlombaan. Salah satu puncak kejayaan Penulis dalam bidang perlombaan adalah mendapatkan Medali Perak OSN Astronomi 2017 Pekanbaru, Riau (mewakili Jawa Timur dengan predikat Juara 1 OSN Astronomi tingkat Provinsi Jawa Timur) dan sempat mengikuti Pelatihan Nasional Tahap 1 *International Olympiad on Astronomy and Astrophysics 2018* di Wisma Kartini, Bandung selama satu bulan. Selama kuliah, Penulis lebih aktif dalam kegiatan akademisi (Asisten Dosen Matematika 1 & 2 dan Asisten Lab. QMIPA), organisasi kemahasiswaan (Kepala Departemen PSDM ITS Astronomy Club 2019, Ketua UKM ITS Astronomy Club 2020, dan Sekretaris Departemen PSDM Forum Daerah Rumah Rotan 2020/2021), magang (*Human Capital of* PT Semen Indonesia (Kerja Praktik 2021) dan *Logistic Operation of* PT Paragon Technology and Innovation (Program Kampus Merdeka 2021)), *freelance* (Manager HRD *of* CV IRIT 2020 dan Tutor Profesional Olimpiade Astronomi tingkat Kota, Provinsi, dan Nasional), dan *research project*. Penulis juga penerima Beasiswa Unggulan *Batch* 1 2019 dan Beasiswa LPDP untuk Program Kampus Merdeka 2021 dari Kemdikbud RI. Saat ini Penulis sedang menyelesaikan Tugas Akhir berjudul "*Multi-Objective Vehicle Routing Problem with Time Window and Drones* (MO-VRPTW-D) Menggunakan Algoritma *Simulated Annealing* dan *Ant Colony Optimization* untuk *Last-Mile Delivery*". Apabila pembaca ingin memberi kritik dan saran atau ingin melakukan diskusi lebih lanjut terkait Tugas Akhir Penulis, dapat disampaikan melalui email: meidani.18024@mhs.its.ac.id.

(Halaman ini sengaja dikosongkan)