TUGAS AKHIR – KS 141501

PENGEMBANGAN METODE *ANT COLONY OPTIMIZATION* DAN *TABU SEARCH* UNTUK MENYELESAIKAN *VEHICLE ROUTING PROBLEM WITH TIME WINDOWS*

DEVELOPMENT METHOD ANT COLONY OPTIMIZATION AND TABU SEARCH FOR COMPLETING THE VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

WILDAN HABIBY FANANI
NRP 5212 100 056

Dosen Pembimbing
Edwin Riksakomara, S.Kom., M.T.
Amalia Utamima, S.Kom., M.B.A.

JURUSAN SISTEM INFORMASI
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

TUGAS AKHIR – KS 141501

PENGEMBANGAN METODE *ANT COLONY OPTIMIZATION* DAN *TABU SEARCH* UNTUK MENYELESAIKAN *VEHICLE ROUTING PROBLEM WITH TIME WINDOWS*

WILDAN HABIBY FANANI
NRP 5212 100 056

Dosen Pembimbing
Edwin Riksakomara, S.Kom., M.T.
Amalia Utamima, S.Kom., M.B.A.

JURUSAN SISTEM INFORMASI
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

TUGAS AKHIR – KS 141501

DEVELOPMENT METHOD ANT COLONY OPTIMIZATION AND TABU SEARCH FOR COMPLETING THE VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

WILDAN HABIBY FANANI
NRP 5212 100 056

Supervisor
Edwin Riksakomara, S.Kom., M.T.
Amalia Utamima, S.Kom., M.B.A.

JURUSAN SISTEM INFORMASI
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

LEMBAR PENGESAHAN

PENGEMBANGAN METODE *ANT COLONY OPTIMIZATION* DAN *TABU SEARCH* UNTUK MENYELESAIKAN *VEHICLE ROUTING PROBLEM WITH TIME WINDOWS*

TUGAS AKHIR
Disusun untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Jurusan Sistem Informasi
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :

WILDAN HABIBY FANANI
NRP 5212 100 056

Surabaya, 27 Juli 2016

KETUA
JURUSAN SISTEM INFORMASI

Dr. Ir. Aris Tjahyanto, M.Kom.
NIP 19650310 199102 1 001

LEMBAR PERSETUJUAN

PENGEMBANGAN METODE *ANT COLONY OPTIMIZATION* DAN *TABU SEARCH* UNTUK MENYELESAIKAN *VEHICLE ROUTING PROBLEM WITH TIME WINDOWS*

TUGAS AKHIR

Disusun untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Jurusan Sistem Informasi
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :

WILDAN HABIBY FANANI
NRP 5212 100 056

Disetujui Tim Penguji : Tanggal Ujian : 27 Juli 2016
Periode Wisuda : September 2016

Edwin Riksakomara, S.Kom., M.T. **(Pembimbing I)**

Amalia Utamima, S.Kom., M.B.A. **(Pembimbing II)**

Wiwik Anggraeni, S.Si., M.Kom. **(Penguji I)**

Faisal Mahananto, S.Kom, M.Eng, Ph.D **(Penguji II)**

PENGEMBANGAN METODE ANT COLONY OPTIMIZATION DAN TABU SEARCH UNTUK MENYELESAIKAN VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

Nama Mahasiswa : Wildan Habiby Fanani
NRP : 5212 100 056
Jurusan : Sistem Informasi FTIF-ITS
Pembimbing 1 : Edwin Riksakomara, S.Kom., M.T.
Pembimbing 2 : Amalia Utamima, S.Kom., M.B.A.

ABSTRAK

Vehicle Routing Problem with Time Windows (VRPTW) merupakan salah satu dari beberapa permasalahan yang sering terjadi pada suatu sistem dalam penyaluran logistik. Sistem pengiriman barang kepada pelanggan yang terdapat pada *e-commerece* merupakan bagian dari permasalahan tersebut. Dimana setiap depot yang memiliki sejumlah kendaraan dengan kapasitas tertentu dapat melayani semua *customer* pada lokasi tertentu. Dengan ketentuan setiap *customer* memiliki jumlah permintaan serta batasan waktu yang berbeda-beda. Serta dalam setiap pengiriman memiliki tujuan untuk meminimalkan biaya distribusi tanpa mengabaikan batasan yang ada.

VRPTW dapat diselesaikan menggunakan metode eksak, heuristik, maupun meta-heuristik. Dalam tugas akhir ini VRPTW diselesaikan menggunakan *Ant Colony Optimization* (ACO) yang berdasarkan pada observasi perilaku koloni semut dalam menentukan jalur untuk mencari lokasi makanan yang kemudian disempurnakan menggunakan *Tabu Search* (TS) dalam pengambilan keputusan.

Penyelesaian VRPTW menggunakan algoritma ACOTS ini diujicobakan pada data set Solomon Problem, yang merupakan standar permasalahan internasional VRPTW. Hasil dari uji

coba tersebut menunjukkan bahwa algoritma ACOTS memberikan hasil solusi yang mendekati optimal atau mendekati solusi terbaik pada Solomon Data Set.

Kata kunci : ***Ant Colony Optimization, Tabu Search, Vehicle Routing Problem with Time Windows***

DEVELOPMENT METHOD ANT COLONY OPTIMIZATION AND TABU SEARCH FOR COMPLETING THE VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

Student Name : Wildan Habiby Fanani
NRP : 5212 100 056
Departement : Sistem Informasi FTIF-ITS
Supervisor 1 : Edwin Riksakomara, S.Kom., M.T.
Supervisor 2 : Amalia Utamima, S.Kom., M.B.A.

ABSTRACT

Vehicle Routing Problem with Time Windows (VRPTW) is one of several problems that often occur in a system in logistics. Delivery systems to customers contained in the e-commerece is part of the problem. Wherein each depot which has a number of vehicles with a certain capacity to service all customers in a specific location. Provided each customer has a number of requests and the time limits vary. As well as in each shipment has a goal to minimize distribution costs without ignoring boundaries.

VRPTW can be solved using exact methods, heuristic, and meta-heuristics. In this final task VRPTW solved using Ant Colony Optimization (ACO), which is based on observation of the behavior of ant colonies in determining the path to find the location of the food which is then refined using Tabu Search (TS) in decision making.

Completion VRPTW with ACOTS algorithm is tested on a data set Solomon Problem, which is the standard international issues VRPTW. Results from these trials show that the algorithm ACOTS results nearly optimal solution or approach the best solution to the Solomon Data Set.

Keyword* : *Ant Colony Optimization, Tabu Search, Vehicle Routing Problem with Time Windows

DAFTAR ISI

DAFTAR GAMBAR

**halaman ini sengaja dikosongkan*

DAFTAR TABEL

BAB VI
KESIMPULAN DAN SARAN

Bab ini berisikan kesimpulan dari hasil penelitian dan juga saran perbaikan untuk penelitian kedepannya beserta masalah yang duhadapi selama mengerjakan penelitian tugas akhir ini.

6.1. Kesimpulan

Hasil uji coba dan analisis yang telah dilakukan dapat diambil beberapa kesimpulan sebagai berikut:

1. *Algoritma gabungan Ant Colony Optimization* dan *Tabu Search* (ACOTS) dapat berjalan pada Vehicle Routing Problem with Time Windows (VRPTW). Dibuktikan dengan hasil validasi algoritma yang berjalan dengan lancar serta pengujian menggunakan 25 titik yang mendapatkan nilai error yaitu 0.27% terhadap solusi eksisting
2. Algoritma ACOTS dapat memberikan solusi yang optimal pada permasalahan VRPTW pada data set solomon C101, C105, dan C106 dengan 25 titik.
3. Pembatasan bilangan random yang digunakan pada semut untuk menentukan tujuan selanjutnya mempengaruhi tingkat optimasi algoritma. Karena semakin terbatas semut tersebut memilih tujuan, semakin sedikit pula variasi rute yang diberikan oleh algorimta sehingga kemungkinan solusi tidak optimal semakin besar.
4. Subrute yang dihasilkan dapat mewakili sebagai jumlah kendaraan yang digunakan untuk melakukan perjalanan. Sehingga dapat menjadi acuan ketika ingin melakukan perhitungan lebih lanjut terkait biaya yang akan dikeluarkan.

6.2. Saran

Saran untuk penelitian ini adalah:

1. Pengolahan data set untuk perhitungan jarak antar dua titik masih menggunakan Ms. Excel dengan perhitungan manual. Sehingga dikhawatirkan terdapat kesalahan perhitungan yang menyebabkan kesalahan pula pada model *software*.
2. Dalam penelitian ini menggunakan *software* MATLAB dengan semua script ada pada satu file serta belum menerapkan OOP. Sehingga ketika terdapat kesalahan terdapat pada salah satu script akan mengalami kesulitan saat melakukan perbaikan.
3. *Ant Colony Optimization* dan *Tabu Search* dapat digabungkan dengan algoritma lain yang dapat meningkatkan kualitas hasil solusi.

DAFTAR PUSTAKA

Abdulkader, M. M., Gajpal, Y., & ElMekkawy, T. Y. (2015). Hybridized ant colony algorithm for the Multi Compartment Vehicle Routing Problem. *ScienceDirect*.

Agustina, P. (2008). *Implementation of Algoritm Ant Colony System (ACS) for Open Vehicle Routing Problem (OVRP) to Determine Distribution Route of Newspaper (Case Study: Harian Surya).* Surabaya: Institut Teknologi Sepuluh Nopember.

Bell, J. E., & McMullen, P. R. (2004). Ant colony optimization techniques for the vehicle routing problem. *Advanced Engineering Informatics*, 41-48.

Brasysy, O., & Grendeau, M. (2001). *Genetic Algoritm for the Vehicle Routing Problem with Time Windows.* Arpakannus.

Haynes, D. D., & Corns, S. M. (2015). Agorithm for Tabu - Ant Colony Optimizer. *IEEE*.

Hindriyanto. (2012, September 3). *Metaheuristics : Tabu Search*. (duniaku) Dipetik Februari 18, 2016, dari https://hindriyanto.wordpress.com/2012/09/03/metaheuristics-tabu-search/

Iswardani, K. (2015). *Penerapan Ant Colony Optimization Pada Vehicle Routing Problem Time Windows (Case Study: CV. Yufa Barokah).* Surabaya: Institut Teknologi Sepuluh Nopember.

Joe, L. (2014, April 30). *Slideshare.* Dipetik February 24, 2016, dari http://www.slideshare.net/LindaJoe/energyaware-task-scheduling-using-antcolony-optimization-in-cloud

Kallehauge, B., Larsen, J., Madsen, O. B., & Solomon, M. M. (2002). Vehicle Routing with Time Windows. Dalam *Desaulniers G. et al., editor. Column Generation.* New York: Springer.

Katagiri, H., Hayashida, T., Nishizaki, I., & Guo, Q. (2012). A hybrid algorithm based on tabu search and ant colony

optimization fo k-minimum spanning tree problem. *ScienceDirect*.

Liu, Qi, & Chen. (2006). Optimization of Vehicle Routing Problem Based on Ant Colony System. Dalam *D. S. Huang, K. Li, and G. W. Irwin (Eds.): Computational Intelligence.*

Priyandari. (2009, September 9). *Tabu Search – Introduction*. (Universitas Sebelas Maret) Dipetik Februari 18, 2016, dari http://priyandari.staff.uns.ac.id/200909/tabu-search-introduction/

Solomon. (2008, April 18). *VRPTW Solomon Benchmark*. (Sintef) Dipetik Februari 26, 2016, dari https://www.sintef.no/projectweb/top/vrptw/solomon-benchmark/100-customers/

Toth, P., & Vigo, D. (2002). *The Vehicle Routing Problem.* Philadelpia: SIAM.

BIODATA PENULIS

Penulis lahir di Lamongan pada tanggal 21 Maret 1994 merupakan anak pertama dari dua bersaudara. Penulis telah menempuh pendidikan formal di MI Muhammadiyah 04 Blimbing Paciran, SMP Negeri 2 Paciran, dan SMA Negeri 1 Babat Lamongan. Setelah selesai menempuh pendidikan SMA, penulis melanjutkan pendidikan di Jurusan Sistem Informasi tahun akngkatan 2012. Selama menempuh pendidikan di perguruan tinggi, penulis aktif dalam berorganisasi, dibuktikan dengan menjadi Kadep Syiar Kajian Islam Sistem Informasi.

Selain itu, penulis juga aktif dalam kegiatan kepanitian seperti Panitia YELS 2013. Pada semester delapan perkuliahan, penulis mulai mengerjakan Tugas Akhir di Laboratorium Rekayasa Data dan Intelegensia Bisnis dibawah bimbingan Bapak Edwin Riksakomara, S.Kom., M.T., serta Ibu Amalia Utamima, S.Kom., M.B.A. dengan topik mengenai optimasi. Semoga penelitian Tugas Akhir ini mampu memberikan kontribusi positif bagi semua pihak terkait. Untuk mendapatkan informasi lebih lanjutmengenai tugas akhir ini, dapat menghubungi penulis melalu email wildan.byfan@outlook.com

BAB I
PENDAHULUAN

1.1. Latar Belakang

Saat ini *e-commerce* di Indonesia mengalami pertumbuhan yang sangat pesat. Mulai dari usaha rumahan hingga toko besar telah membuka lapak online mereka. Bagi kustomer, kehadiran toko online tentu sangat memudahkan mereka agar tidak pelru keluar rumah untuk mendapatkan barang. Selain itu *e-commerce* erat kaitannya dengan jasa ekspedisi yang akan mengirim barang menuju kustomer. Dengan demikian semakin banyaknya transaksi yang ada, maka semakin banyak pula kustomer yang akan dituju.

Dari sinilah terdapat permasalahan yang mengakibatkan banyaknya keterlambatan pengiriman. Salah satu penyebabnya adalah jalur distribusi barang yang kurang tepat dalam menjalankan proses pengiriman. Disisi lain terdapat beberapa pelanggan yang memiliki batasan waktu penerimaan, sehingga ketika terdapat keterlambatan tersebut akan menimbulkan sebuah kerugian. Hal tersebut dapat digolongkan sebagai *Vehicle Routing Problem with Time Windows* (VRPTW). Permasalahan tersebut muncul dimana suatu pengiriman harus menentukan siapa pengirimnya dan rute yang dituju dengan melihat batasan kapasitas setiap pengirim serta waktu yang menjadi target. Oleh sebab itu pemilihan jalur distribusi serta siapa yang mengirim adalah salah satu solusi untuk meningkatkan efektifitas serta efisiensi pengiriman. Sehingga jumlah pekerja, waktu, dan biaya yang dikeluarkan menjadi lebih minimal.

Metode yang akan digunakan adalah *Ant Colony Optimization* (ACO) dan *Tabu Search* (TS). ACO merukapan metode yang meniru koloni semut ketika mencari sumber makanan dan kembali ke sarangnya yang secara alami mencari rute terpendek. Sedangkan TS merupakan metode yang digunakan

untuk permasalah optimasi kombinatorial. Kedua metode tersebut sangat cocok dan berkaitan dengan permasalahan yang ada (Abdulkader, Gajpal, & ElMekkawy, 2015). Oleh karena itu, peneliti akan menggunakan metode gabungan ACOTS untuk menyelesaikan permasalah VRPTW sehingga diharapkan mendapatkan hasil yang optimal.

1.2. Perumusan Masalah

Dari uraian diatas, dirumuskan sebuah permasalahan yang akan dibahas pada tugas akhir ini adalah:

1. Bagaimana mengaplikasikan algoritma gabungan *Ant Colony Optimization* dan *Tabu Search* (ACOTS) pada *Vehicle Routing Problem with Time Windows* (VRPTW)?
2. Bagaimana mendapatkan solusi yang optimal pada VRPTW menggunakan ACOTS?

1.3. Batasan Tugas Akhir

Batasan permasalahan pada tugas akhir ini adalah sebagai berikut:

1. Penelitian ini menggunakan data set *solomon problem* dari website Sintef : *Research, Technology and Innovation* (Solomon, 2008).
2. Jumlah kendaraan, kapasitas kendaraan, serta waktu *loading* dan *unloading* barang disesuaikan dengan data set yang ada.

1.4. Tujuan Tugas Akhir

Berdasarkan latar belakang permasalaan yang telah dijelaskan sebelumnya, tujuan dari tugas akhir ini adalah:

1. Mengembangkan algoritma gabungan *Ant Colony Optimization* dan *Tabu Search* (ACOTS) pada *Vehicle Routing Problem with Time Windows* (VRPTW).
2. Mendapatkan solusi yang optimal pada VRPTW menggunakan ACOTS.

1.5. Manfaat Tugas Akhir

Manfaat dari tugas akhir yang peneliti lakukan adalah sebagai berikut:

1. Mendapatkan rute tercepat sehingga dapat menimimalkan biaya yang dikeluarkan.
2. Mampu mengetahui jumlah kendaraan serta waktu yang dibutuhkan.

1.6. Relevansi

Topik yang diangkat pada tugas akhir ini mengenai optimasi pengelolaan rute dengan batasan waktu untuk meminimalkan biaya yang digunakan. Topik tersebut berkaitan dengan sistem pendukung keputusan serta riset operasi lanjut. Pada pohon penelitian laboratorium Rekayasa Data dan Intelegensi Bisnis (RDIB), topik ini termasuk pada kategori *optimization*.

**halaman ini sengaja dikosongkan*

BAB II
TINJAUAN PUSTAKA

Pada bagian ini akan dibahas mengenai tinjauan pustaka dan teori-teori yang mendukung dalam pengerjaan tugas akhir. Teori-teori tersebut antara lain: *Vehicle Routing Problem with Time Windows*, *Ant Colony Optimization*, *Tabu Search*.

2.1. Studi Sebelumnya

1. "*Algorithm for a Tabu – Ant Colony Optimizer*" oleh David D. Haynes and Steven M. Corns (Haynes & Corns, 2015). Pada penelitian tersebut dijelaskan bahwa algoritma Tabu-ACO dapat berjalan dengan baik dalam *Steiner Tree Problem* (STP) di mana *leaf node* yang dirasa tidak optimal dapat dipangkas dari *local search*. Pengukuran STP sangat berhubungan dengan beberapa jalur dan terdiri dari sedikit *leaf node*.
2. "*A hybrid algorithm based on tabu search and ant colony optimization for k-minimum spanning tree problems*" oleh Hideki Katagiri, Tomohiro Hayashida, Ichiro Nishizaki, Qingqiang Guo (2012) (Katagiri, Hayashida, Nishizaki, & Guo, 2012). Pada penelitian ini menjelaskan tentang metode gabungan (*hybrid*) *tabu search* (TS) dan *ant colony optimization* (ACO) serta membandingan *hybrid* tersebut dengan metode utama mereka, yaitu TS dan ACO untuk studi kasus *k-minimun spanning tree problem*. Hasil yang didapatkan dalam paper ini adalah metode *hybrid* memberikan nilai yang lebih optimal dibandingkan dengan metode awalnya.
3. "*Hybridized ant colony algorithm for the Multi Compartment Vehicle Routing Problem*" oleh Abdulkader, Yuvraj Gajpal, Tarek Y. ElMekkawy (2015) (Abdulkader, Gajpal, & ElMekkawy, 2015). Pada penelitian ini peneliti mengusulkan untuck menggunakan algoritma *ant colony* yang dikombinasikan dengan prosedur *local search* untuk memecahkan *Multi*

Compartment Vehicle Routing Problem. Percobaan tersebut menunjukkan bahwa rata-rata algoritma *hybrid* yang diusulkan menghasilkan hasil yang lebih baik serta hanya memerlukan sedikit waktu komputasi. Selain itu, algoritma tersebut dapat mempertahankan performanya dalam masalah yang lebih besar. Hasil percobaan juga menunjukkan bahwa metode *hybrid* diperlukan untuk meningkatkan kinerja algoritma *ant colony* itu sendiri.

2.2. Dasar Teori

Subbab ini berisikan dasar teori yang akan digunakan dalam penelitian tugas akhir ini, mencakup teori dan metode yang digunakan

2.2.1. Vehicle Routing Problem with Time Windows

Vehicle Routing Problem adalah permasalahan pencarian rute dalam menentukan sistem distribusi. Tujuan VRP adalah menentukan rute optimal untuk beberapa kendaraan yang telah diketahui kapasitasnya, agar dapat memenuhi *demand* dari *customer* dimana *demand* tersebut tidak melebihi kapasitas kendaraan (Agustina, 2008). Rute yang optimal adalah rute yang memiliki total jarak dan waktu perjalanan terpendek dalam memenuhi *demand customer* serta menggunakan kendaraan yang seminimal mungkin, sehingga dapat meminimalkan biaya distribusi.

Vehicle Routing Problem with Time windows (VRPTW) merupakan perluasan dari VRP yang paling sering ditemukan dalam pengambilan keputusan mengenai penentuan rute untuk pengambilan barang. Setiap kendaraan yang bertugas pada VRPTW hanya dapat keluar dari depot pada jam kerja dan harus kembali sebelum jam kerja berakhir (Brasysy & Grendeau, 2001).

Tujuan *VRPTW* adalah menentukan sejumlah rute untuk melayani seluruh kustomer dengan biaya distribusi serendah

mungkin tanpa melanggar batasan waktu dari depot dan waktu yang ditentukan oleh pihak pelanggan. Jumlah rute tidak boleh melebihi jumlah kendaraan (Iswardani, 2015).

Menurut Toth, P. dan Vigo, D (Toth & Vigo, 2002)., komponen-komponen yang terdapat dalam *Vehicle Routing Problem* adalah:

a. Jaringan jalan, biasanya direpresentasikan dalam sebuag diagram yang terdiri dari *arc* (lengkung/garis atau bagian-bagian jalan) dan *vertex* (titik lokasi *customer* dan depot). Setiap *arc* diasosiasikan dengan jarak, waktu, dan biaya untuk menuju *vertex* tertentu berdasarkan kondisi yang ada (jenis kendaraan, kondisi/karakteristik jalan, dan periode perlintasan).
b. *Customer*, ditandai dengan *vertex* (titik) dan biasanya memiliki hal-hal seperti berikut:
    1. Jumlah permintaan barang
    2. Periode pelayanan
    3. Waktu yang dibutuhkan untuk menurunkan atau memuat barang
    4. Pengelompokan titik
    5. Prioritas
c. Depot, juga ditandai dengan suatu titik yang merupakan awal dan akhir dari sebuah rute.
d. Kendaraan, merupakan alat yang digunakan untuk mendistribusikan barang dan memiliki hal sebagai berikut:
    1. Depot awal/akhir
    2. Kapasitas
    3. Kemungkinan untuk dipisah menjadi beberapa kompartemen untuk mengangkut barang dengan jenis yang berbeda-beda
    4. Alat yang tersedia untuk operasi
    5. Pengelompokan rute
    6. Biaya

e. Pengemudi, memiliki kendala seperti jam kerja harian, durasi maksimum perjalanan, serta lembur yang biasanya dikenakan pada kendaraan yang digunakan.
f. Batasan waktu, termasuk biaya tambahan jika tidak sesuai dengan waktu yang ditentukan.

Liu dkk. (Liu, Qi, & Chen, 2006) memodelkan VRPTW sebagai berikut: terdapat *n-customer* yang memiliki *demand*. *Demand* dari tiap *customer* akan dilayani oleh beberapa kendaraan dan *demand* tidak boleh melebihi kapasitas kendaraan. Kendaraan akan berangkat dari depot dan kembali ke depot tersebut. Kendaraan pertama akan melayani customer awal yang ditentukan dan beberapa customer lainnya selama tidak melanggar constraint yang ditentukan. Tujuannya untuk meminimasi waktu pada saat kendaraan meninggalkan depot sampai kendaraan terakhir kembali ke depot. Kendaraan dinotasikan dari 1 sampai m, customer dinotasikan dari 1 sampai n dan node yang harus dikunjungi mulai dari 0,1,2,3,…,n dimana 0 menunjukkan depot.

Formulasi matematis VRPTW menurut Kallehauge dkk. (Kallehauge, Larsen, Madsen, & Solomon, 2002) adalah sebagai berikut:
Meminimalkan:

$$z = \sum_{k \in V} \sum_{i \in N} \sum_{k \in N} c_{ij} x_{ijk}$$

Dengan batasan:

1. Setiap pelanggan dikunjungi tepat satu kali oleh suatu kendaraan:
   $$\sum_{k \in V} \sum_{i \in N} \sum_{k \in N} c_{ij} x_{ijk}$$
2. Total permintaan semua pelanggan dalam satu rute tidak melebihi kapasitas kendaraan:
   $$\sum_{i \in C} d_i \sum_{j \in N} x_{ijk} \leq q, \forall k \in V$$
3. Setiap rute awal dari depot 0:

$$\sum_{j \in N} x_{0jk} = 1, \forall k \in V$$

4. Setiap kendaraan yang mengunjungi suatu pelanggan pasti akan meninggalkan pelanggan tersebut:
$$\sum_{i \in N} x_{ihk} - \sum_{j \in N} x_{hjk} = 0, \forall h \in C, \forall k \in V$$
5. Setiap rute berakhir di depot n+1:
$$\sum_{i \in N} x_{i,n+1,k} = 1, \forall k \in V$$
6. Suatu kendaraan k yang menuju j dari i, tidak dapat tiba di j sebelum $s_{ik} + t_{ij}$. Jadi jika $x_{ijk} > 0$ maka $s_{ik} + t_{ij} \leq s_{jk}$, bentuk linearnya adalah:
$$s_{ik} + t_{ij} - M_{ij}(1 - x_{ijk}) \leq S_{jk}, \forall i, j \in N, \forall k \in V$$
Dengan M_{ij} adalah konstanta besar yang tidak kurang dari nilai maksimum dari $b_i + t_{ij} - a_j; (i,j) \in A$
7. Waktu pelayanan disetiap pelanggan memenuhi *time windows*:
$$a_i \leq s_{jk} \leq b_i, \forall i \in N, \forall k \in V$$
8. Perubahan x_{ijk} merupakan perubahan biner:
$$x_{ijk} \in \{0,1\}, \forall i, j \in N, \forall k \in V$$
Dimana:
V = himpunan kendaraan berkapasitas sama
C = himpunan pelanggan = {1,…,n}
A = himpunan titik jalur = $\{(i,j) | i, j \in N, i \neq j\}$
c_{ij} = jarak/biaya dari simpul i ke simpul j.
t_{ij} = waktu tempuh dari simpul i ke simpul j.
d_i = jumlah permintaan pelanggan i.
q = kapasitas kendaraan
$[a_i, b_i]$ = time windows dari simpul i.

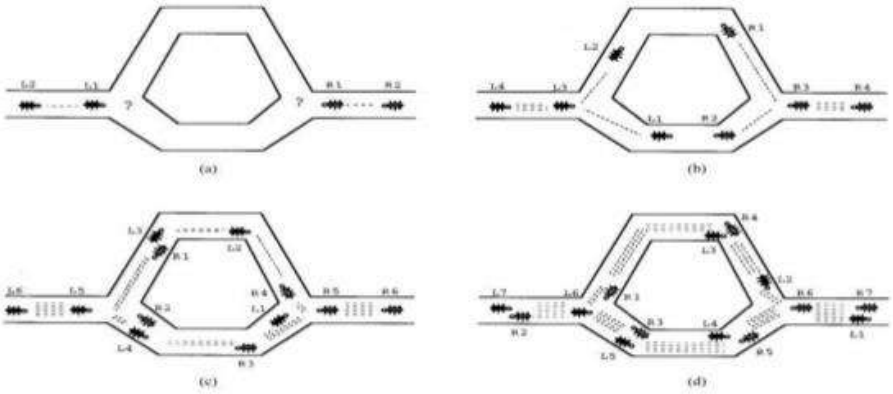
Untuk setiap $(i,j) \in A, i \neq n+1, j \neq 0$ dan untuk setiap kendaraan k didefinisikan berupa:

$$x_{ijk} = \begin{cases} 1, \text{jika kendaraan } k \text{ melalui } i \text{ lalu ke } j \\ 0, \text{selainnya} \end{cases}$$

s_{ik} = waktu bagi kendaraan k mulai melayani pelanggan i, dengan q dan di adalah bilangan integer tak negatif dan a_i, b_i, c_{ij}, dan t_{ij} adalah bilangan tak negatif. Pada simpul depat diasumsikan $a_0 = b_0 = a_{n+1} = 0$ dan $s_{0k} = 0$ setiap k.

2.2.2. Any Colony Optimization

Pada dasarnya konsep Ant Colony Optimization adalah meniru perilaku sekelompok semut untuk menemukan solusi terbaik dari permasalahan optimasi kombinatorial ketika sekelompok semut mencari makanan. Ketika mencari sumber makanan, semut-semut tersebut akan menyebar mencari jalur tercepat untuk mendapatkan makanan tersebut. Dalam peroses pencarian tersebut semut berkomunikasi dengan semut lain menggunakan zat kimia yang dinamakan *pheromone* seperti yang ada pada Gambar 2.1. Sehingga nantinya semut yang lain secara perlahan dan otomatis akan memilih jalur yang memiliki tingkat *pheromone* yang lebih besar. Karena semakin banyak *pheromone* pada sebuah jalur tentunya semakin banyak pula semut yang pernah melewati jalur tersebut.



Gambar 2.1 Ilustrasi ACO [11]

Seekor semut akan meninggalkan sejumlah *pheromone* dalam perjalanannya sehingga semut-semut yang lain dapat mengikuti jejak *pheromone* tersebut. Semakin banyak jumlah semut yang melewati rute tertentu, maka semakin kuat *pheromone* yang ada pada rute tersebut sehingga semut yang lain akan mengikuti dimana terdapat pheromone yang paling

kuat. *Pheromone* dalam rute tentunya mengalami penguapan, namun karena banyak yang melewati rute serta meninggalkan jejak lain maka nilai *pheromone* akan tetap kuat dan rute tersebut merupakan jarak terpendek yang dilalui. Hal inilah yang mendasari konsep *ant colony optimization* sehingga dapat digunakan untuk mencari rute terpendek serta menghasilkan biaya yang minimum (Bell & McMullen, 2004).

Agustina (Agustina, 2008) menyatakan bahwa terdapat tiga karakteristik utama dari ACO adalah aturan transisi status, aturan *pheromone local updating*, dan aturan *pheromone global updating*:

a. Aturan transisi status
   Aturan yang digunakan oleh semut untuk memutuskan kemana dia akan pergi. Semut tersebut dapat memilih lintasan baru (lintasan yang belum pernah dilewati semut) atau lintasan terbaik (lintasan yang memiliki jumlah *pheromone* terbanyak dan jarak terpendek) secara probabilitas.
b. Aturan *local updating*
   Aturan yang digunakan untuk menandai lintasan yang baru dilalui dengan meng-*update* jumlah *pheromone* yang ada pada sebuah lintasan. Pada setiap lintasan yang dilewati semut akan mengalami penambahan serta pengurangan (penguapan) *pheromone*. Hal ini dapat digunakan untuk menentukan lintasan yang baik dan buruk.
c. Aturan *global updating*
   Aturan yang digunakan setelah semua semut membentuk jalur perjalanan. Pada aturan ini akan dilakukan penguranan *pheromone* pada semua lintasan, kemudian melakukan penambahan jumlah *pheromone* pada sisi-sisi yang termasuk dalam perjalanan dengan jarak terpendek.

2.2.3. Tabu Search

Tabu search (TS) pertama kali diperkenalkan oleh Glover sekitar tahun 1986. Glover menyatakan bahwa TS adalah salah satu prosedur metaheuristik tingkat tinggi untuk penyelesaian permasalahan optimisasi kombinatorial. TS ini dirancang untuk mengarahkan metode-metode lain (atau komponen proses TS itu sendiri) untuk keluar atau menghindari dari masuk dalam solusi optimal yang bersifat lokal. Kemampuan TS dalam menghasilkan solusi yang mendekati optimal telah dimanfaatkan dalam beragam permasalahan klasik dan parktis dari berbagai bidang mulai bidang penjadwalan hingga bidang telekomunikasi (Priyandari, 2009).

Tabu Search mempunyai tiga komponen penting:

1. Penggunaan struktur memori yang fleksibel, yang memungkinkan evaluasi kriteria dan informasi historis bisa di eksploitasi lebih baik dibandingkan dengan struktur memori yang statis.
2. Penggunaan mekanisme control yang didasarkan interaksi antara kondisi yang membatasi dan kondisi yang mendukung proses pencarian.
3. Penggabungan fungsi memori yang memiliki rentang waktu yang berbeda, dari memori jangka pendek sanpai dengan memori jangka panjang, untuk menerapkan strategi intensifikasi dan diversifikasi.

Didalam menerapkan *Tabu search*, ada beberapa bagian yang perlu diperhatikan (Hindriyanto, 2012), yaitu:

1. *Tabu list* : tujuan dari penggunaan *Tabu list* (memori jangka pendek) adalah untuk menghindari mengevaluasi kandidat solusi yang pernah dikunjungi sebelumnya. Akan tetapi, menampung semua kandidat solusi yang pernah dievaluasi kedalam *Tabu list* akan membuat *Tabu search* menjadi tidak efektif (memerlukan ukuran memori yang besar dan wakt yang lama untuk mengecek apakah suatu candidate solusi pernah dievaluasi atau belum).

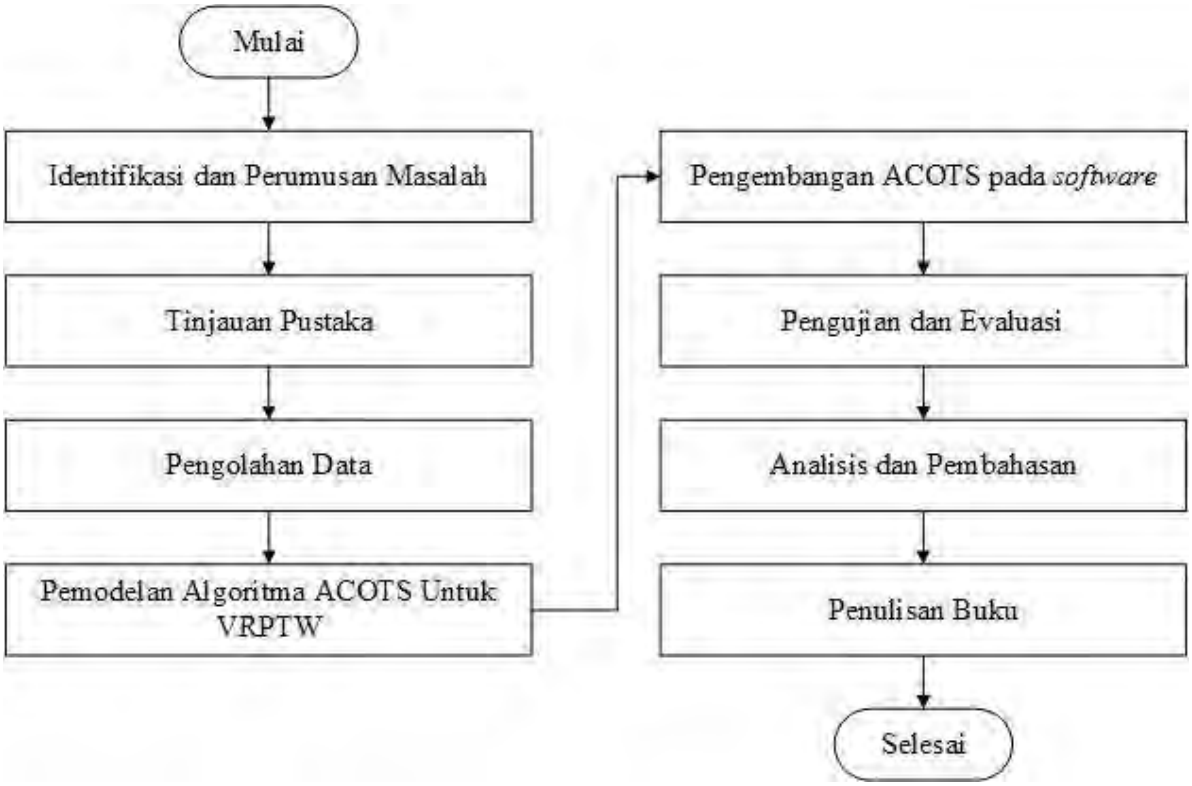
2. *Aspiration criteria*: umumnya diterapkan jika pergerakan tabu tersebut menghasilkan kandidat solusi yang memiliki nilai yang lebih baik daripada solusi terbaik yang telah dihasilkan.
3. Intensifikasi (memori jangka menengah): memori jangka menengah menyimpan sejumlah solusi yang berkualitas (*elite solution*) yang dihasilkan selama proses pencarian. Memori jangka menengah ini bertujuan untuk memberikan prioritas kepada atribut dari solusi berkualitas tersebut.
4. Diversifikasi (memori jangka panjang): memori jangka panjang menyimpan infromasi tentang kandidat solusi yang pernah dikunjungi. Berdasarkan infromasi tersebut, memori ini dapat mengeksplorasi area dalam ruang pencarian yang belum dikunjungi.

BAB III
METODOLOGI

Metodologi penelitian ini berisi tahapan-tahapan sistematis yang digunakan dalam melakukan penelitian. Tahapan-tahapan tersebut merupakan suatu kerangka berfikir yang dijadikan sebagai acuan agar proses penelitian berjalan secara sistematis, terstruktur dan terarah, serta dijadikan pedoman penelitian untuk mencapai tujuan yang telah ditetapkan sebelumnya.

3.1. Bagan Tahapan Pelaksanaan Tugas Akhir

Dalam pelaksanaan tugas akhir ini peneliti memiliki beberapa tahapan yang perlu dilaksanakan, berikut adalah bagan tahapan palaksanaan tugas akhir:



Gambar 3.1 Tahapan Pelaksanaan Tugas Akhir

3.2. Uraian Metodlogi

Berdasarkan pada Gambar 3.1 yang merupakan tahapan palaksanaan tugas akhir pada sub bab sebelumnya, di bawah ini merupakan penjelasan dari setiap prosesnya.

3.2.1. Identifikasi dan perumusan masalah

Pada tahapan pertama ini dilakukan identifikasi serta perumusan masalah yang terkait dengan latar belakang, tujuan serta manfaat yang akan didapat dalam penelitian. Pada tahapan ini diharapkan mampu menjadi awalan dalam memulai penelitian serta secara tidak langsung dapat menjadi masukan bagi tahapan selanjutnya.

3.2.2. Tinjauan pustaka

Pada tahapan ini dilakukan studi literatur terkait penelitian yang telah dilakukan oleh peneliti lain. Studi yang dilakukan terkait permasalahan yang diambil, metode yang digunakan, serta hasil yang didapat. Sehingga diharapkan dapat digunakan sebagai bahan acuan serta perbandingan agar penelitian lebih baik.

3.2.3. Pengolahan data

Data set yang telah diambil akan diproses sesuai dengan kebutuhan penelitian. Sehingga dapat mencakup fungsi tujuan serta batasan yang terdapat pada studi kasus.

3.2.4. Pemodelan algoritma untuk VRPTW

Pada tahapan ini adalah memodelkan algoritma gabungan antara *Ant Colony Optimizazion* dan *Tabu Search.* Model yang diterapkan adalah *flow diagram* dari algoritma gabungan yang akan diterapkan oleh peneliti untuk mendapatkan solusi.

3.2.5. Pengembangan ACOTS pada *software*

Model algoritma yang sudah dibuat sebelumnya, dijadikan acuan untuk pengembangan model pada *software*. Dimana *software* yang digunakan adalah MATLAB. Model *software* ini nantinya yang akan digunakan untuk memproses data yang

telah didapat sebelumnya. Input yang dibutuhkan adalah jarak *customer*, *time windows*, kapasitas kendaraan, *demand*, *loading/unloading*.

3.2.6. Pengujian dan evaluasi

Setelah model software selesai dibuat maka akan dilakukan pengujian. Tujuan pengujian adalah untuk memastikan bahwa program tersebut dapat berjalan serta menghasilkan *output* yang tepat. Dimana *output* yang dibutuhkan adalah jumlah pengirim, jarak rute, dan waktu. Kemudian dilakukan evaluasi ketika terdapat permasalahan dalam pelaksanaan pengujian.

3.2.7. Pembahasan dan penarikan kesimpulan

Pada tahapan ini dilakukan analisis dari hasil running model terkait tingkat optimasi yang didapatkan serta kelebihan dan kekurangan algoritma yang dijalankan. Hasil yang didapat akan dibandingkan dengan solusi yang telah ada pada *Solomon Dataset*. Penarikan kesimpulan terdiri dari poin-poin yang menjawab tujuan dari penelitian. Selain penarikan kesimpulan diberikan juga saran terkait penelitian yang dilakukan.

3.2.8. Penulisan buku

Dalam penulisan buku terdapat serangkaian proses yang peneliti laksanakan sehingga dapat terdokumentasi dengan baik.

**halaman ini sengaja dikosongkan*

BAB IV
IMPLEMENTASI

Pada bagian ini akan dilakukan pengolahan data yang dibutuhkan untuk pengembangan ACOTS pada VRPTW, validasi model *software* yang dibuat.

4.1. Pengolahan Data

Pada subbab ini akan dilakukan pengumpulan informasi terkait data-data yang dibutuhkan. Data-data tersebut diantaranya adalah lokasi titik depot dan *customer*, jumlah maksimal kendaraan, *time windows* depot dan *customer*, serta rute eksisting.

CUST NO.	XCOORD.	YCOORD.	DEMAND	READY TIME	DUE DATE	SERVICE TIME
1	40.00	50.00	0.00	0.00	1236.00	0.00
2	45.00	68.00	10.00	912.00	967.00	90.00
3	45.00	70.00	30.00	825.00	870.00	90.00
4	42.00	66.00	10.00	65.00	146.00	90.00
5	42.00	68.00	10.00	727.00	782.00	90.00
6	42.00	65.00	10.00	15.00	67.00	90.00

Gambar 4.1 Bagian data set

Pada Gambar 4.1 merupakan sebagian data VRPTW yang digunakan dalam pengujian program ACOTS. Terdapat beberapa keterangan yang menunjukkan bagian yang berfungsi sebagai koordinat maupun batasan, yaitu:

CUST NO = Nomor index setiap titik, dimana index 1 merupakan depot dan selain itu merupakan kustomer
XCOORD. = Titik koordinat pada sumbu x
YCOORD. = Titik koordinat pada sumbu y
DEMAND = Jumlah permintaan untuk masing-masing titik
READY TIME = Batas minimal kendaraan sampai pada titik
DUE DATE = Batas maksimal kendaraan sampai pada titik
SERVICE TIME = Waktu yang digunakan untuk bongkar muat pada suatu titik

4.1.1. Data demand dan time windows

Seperti pada Gambar 4.1, terdapat kolom *demand* dimana permintaan masing-masing *customer* berbeda-beda. Selain itu setiap *customer* juga dan memiliki *time windows* yaitu batas minimal serta maksimal suatu kendaraan sampai pada titik tertentu. Disamping itu, setiap kendaraan yang telah sampai pada titik tujuan, maka kendaraan tersebut akan berhenti pada titik tersebut untuk melakukan *loading/unloading*. Waktu yang digunakan untuk melakukan hal tersebut merupakan *Service Time*. Jadi pada dasarnya waktu yang dibutuhkan kendaraan untuk menuju suatu titik yaitu waktu perjalanan dan waktu servis, yang nantinya akan dibandingkan dengan *time windows* titik tersebut.

4.1.2. Lokasi *customer* dan depot

Pada Gambar 4.1 setiap *customer* memiliki lokasi yang ditunjukkan melalui titik koordinat x dan y . Dari data yang berupa titik koordinat tersebut akan diolah untuk mencari jarak pada setiap titik-titik koordinat antar *customer* dengan menggunakan rumus:

$$\sqrt{(X_a - X_b)^2 + (Y_a - Y_b)^2}$$

Dimana:

X_a = titik X awal
X_b = titik X tujuan

Y_a = titik Y awal
Y_b = titik Y tujuan

Dari rumus perhitungan antar dua titik tersebut, maka data yang akan diolah dalam *software* akan berbentuk *array* dengan ukuran nxn.

4.1.3. Solusi eksisting pada data set

Terdapat beberapa tipe data set yang ada pada VRPTW *solomon data sets* yaitu *Randomly*, *Clustered*, dan Mix. Setiap tipe data set terdapat dua varian serta setiap varian terdiri dari 8 hingga 12 nomor data set. Pada penelitian ini penulis menggunakan data set *Clustered* Varian 1 nomor C101, C105, dan C106.

Tabel 4.1 Solusi eksisting

Problem	NV	Distance
C101.25	3	191,3
C105.25	3	191,3
C106.25	3	191,3

Pada Tabel 4.1 terdapat kolom *problem* yang berisi nomor data set dengan akhiran angka 25 yang berarti hanya 25% data yang digunakan pada penelitian tersebut. Pada kolom selanjutnya yaitu *Number of Vehicles* yang berarti jumlah *vehicle* yang digunakan untuk menyelesaikan rute. Pada kolom selanjutnya yaitu *distance* yang merupakan total jarak yang ditempuh untuk menyelesaikan rute tersebut. Sehingga pada Tabel 4.1, menunjukkan bahwa solusi eksisting yang ada untuk C101.25, C105.25, dan C106.25 yaitu menggunakan 3 *vehicles* dengan total jarak tempuh 191,3 satuan jarak.

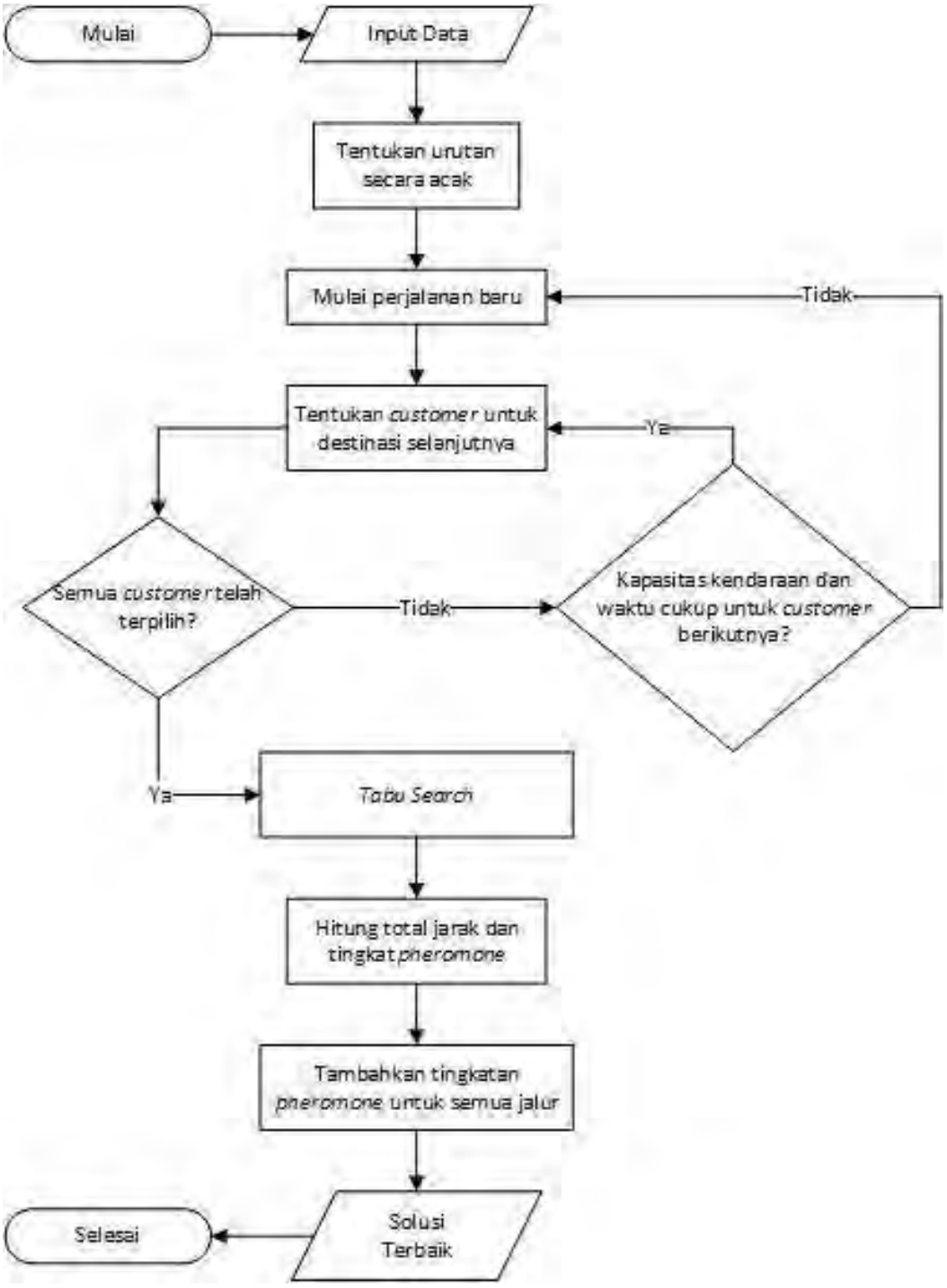
4.2. Pengembangan Algoritma ACOTS pada VRPTW

Pada penelitian ini akan dikembangkan algoritma penyelesaian untuk permasalahan VRPTW berdasarkan algoritma ACOTS:

1. Menentikan parameter yang digunakan yaitu jumlah semut, jumlah iterasi, kapasitas, jumlah titik, jarak antar titik, *time windows*, dan *demand* masing-masing *customer*. Selain itu terdapat parameter lain yaitu alpha, beta, jumlah *pheromone*, dan evaphor *pheromone*.
2. Menerapkan aturan *state transition*, dimana aturan tersebut digunakan untuk memilih titik yang akan dituju oleh semut.

3. Menerapkan aturan *local updating* yang digunakan untuk memodifikasi jumlah *pheromone*.
4. Mengulangi langkah 1-3 hingga semua semut mendapatkan rute untuk mengunjungi semua titik.
5. Membagi solusi sesuai batasan yang ditetapkan sesuai data set yang ada. Batasan yang digunakan adalah *time windows* dan kapasitas kendaran.
6. Menghitung total jarak untuk masing-masing rute pada setiap iterasi.
7. Menggunakan tabu search untuk melihat apakah terdapat solusi yang lebih baik pada satu iterasi.
8. Menerapkan aturan global updating.
9. Menyimpan solusi rute terbaik pada setiap iterasi.
10. Menentukan solusi terbaik dari yang terbaik dalam setiap iterasi.

Secara sistematis penerapan algoritma ACOTS dalam studi kasus VRPTW terdapat pada Gambar 4.2



Gambar 4.2 Flochart Algoritma ACOTS

4.3. Pengembangan ACOTS pada *Software*

Setelah *flow diagram* dibuat, maka selanjutnya adalah pengembangan model tersebut pada *software*. Sehingga nantinya dapat diaplikasikan sesuai dengan kebutuhan peneliti untuk menunjang penelitian. *software* yang digunakan oleh peneliti adalah MATLAB dengan rincian sebagai berikut.

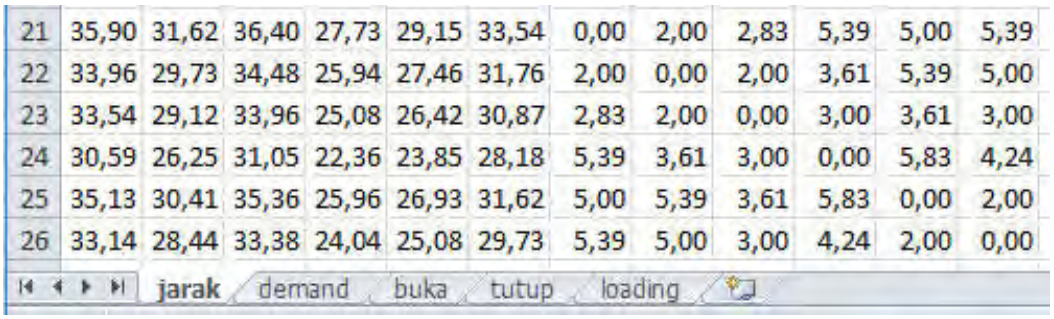
4.3.1. Input Data Set

Data set awal yang didapat terdiri dari maksimal kapasitas angkut, maksimal jumlah *vehicles*, dan tabel yang berisi titik-titik yang ditunjukkan melalui koordinat x dan y, *demand*, serta *ready time*, *due date*, dan *service time* setiap titik. Seluruh data tersebut disatukan menjadi sebuah *file* Ms.Excel sehingga dapat diproses dalam *software* MATLAB.

C101						
VEHICLE						
NUMBER	CAPACITY					
25	200					
CUSTOMER						
CUST NO.	XCOORD.	YCOORD.	DEMAND	READY TIM	DUE DATE	SERVICE TIME
0	40	50	0	0	1236	0
1	45	68	10	912	967	90
2	45	70	30	825	870	90
3	42	66	10	65	146	90
4	42	68	10	727	782	90
5	42	65	10	15	67	90
6	40	69	20	621	702	90
7	40	66	20	170	225	90
8	38	68	20	255	324	90
9	38	70	10	534	605	90
10	35	66	10	357	410	90

Gambar 4.3 Data set awal pada Ms. Excel

Gambar 4.3 merupakan data set awal yang telah diproses dan disimpan sebagai *file* Ms. Excel. Sehingga dapat mempermudah untuk melakukan proses pemisahan setiap parameter yang akan digunakan.



21	35,90	31,62	36,40	27,73	29,15	33,54	0,00	2,00	2,83	5,39	5,00	5,39
22	33,96	29,73	34,48	25,94	27,46	31,76	2,00	0,00	2,00	3,61	5,39	5,00
23	33,54	29,12	33,96	25,08	26,42	30,87	2,83	2,00	0,00	3,00	3,61	3,00
24	30,59	26,25	31,05	22,36	23,85	28,18	5,39	3,61	3,00	0,00	5,83	4,24
25	35,13	30,41	35,36	25,96	26,93	31,62	5,00	5,39	3,61	5,83	0,00	2,00
26	33,14	28,44	33,38	24,04	25,08	29,73	5,39	5,00	3,00	4,24	2,00	0,00

jarak | demand | buka | tutup | loading

Gambar 4.4 Data set yang telah diproses pada Ms. Excel

Gambar 4.4 merupakan hasil dari pengolahan data set yang diproses dengan membagi beberapa bagian tabel agar mempermudah dalam peroses input parameter.

4.3.2. Input Parameter

Sebelum melakukan eksekusi pada program, tentunya dibutuhkan beberapa parameter yang diperlukan agar program dapat berjalan sesuai dengan tujuan. Oleh sebab itu, setelah proses pengolahan data set pada Ms. Excel, data tersebut akan dibentuk dalam variabel-variabel tertentu.

```
filename='C100.xlsx';
distance=xlsread(filename,'jarak');
d=xlsread(filename,'demand');
si=xlsread(filename,'loading');
li=xlsread(filename,'tutup');
ei=xlsread(filename,'buka');
```

Gambar 4.5 Parameter pada data set

Pada Gambar 4.5 terdapat variabel *filename* menunjukkan *file* Ms. Excel tujuan yang akan diproses pada program. Kemudian terdapat beberapa variabel lain yaitu:

distance = jarak antar titik koordinat
d = *demand* setiap titik (kustomer)
si = waktu *loading* (*service time*)

li = waktu tutup (*due date*)
ei = waktu buka (*ready time*)
Selain parameter yang terdapat pada sebuah *file* tersebut, terdapat parameter lain yang yang mendukung kinerja program.

```
m = n_ants; %jumlah semut
n = length(distance); %jumlah tujuan
e = 0.5; %evaporation coefficient
alpha=1; %pangkat untuk ants' sight
beta=3; %pangkat untuk trace's effect
```

Gambar 4.6 Parameter penunjang acots

Pada Gambar 4.6 terdapat beberapa parameter yang mampu mendukung kinerja program, yaitu:

m = jumlah semut, dimana jika semakin banyak semut yang digunakan tentu semakin banyak pula variasi rute yang didapat.
n = jumlah titik tujuan, dimana semakin banyak titik tujuan, maka semakin banyak pula kemungkinan kombinasi yang akan muncul.
e = koefisien evaporasi, yaitu koefisien tingkat penguapan pada *pheromone*.
aplha = parameter pengendali *pheromone*, semakin besar alpha maka semakin besar pula pertimbangan semut menentukan tujuan berdasarkan *pheromone*.
beta = parameter pengendali visibilitas jarak, semakin besar beta maka semakin besar pula pertimbangan semut menentukan tujuan berdasarkan visibilitas jarak antar titik.

4.3.3. Keterangan Program

Setelah melakukan inisialisasi parameter pada program, maka selanjutnya adalah proses yang dijalankan oleh program itu sendiri. Terdapat serangkaian proses yang ada dalam pengembangan model pada *software* dimulai dari menghitung tingkat visibilitas jarak antar titik.

```
%menghitung visibility
for i=1:n
    for j=1:n
        if distance(i,j)==0
            h(i,j)=0;
        else
            h(i,j) = 1/distance(i,j); %inverse distance
        end
    end
end
```

Gambar 4.7 Tingkat visibilitas jarak antar titik

Pada Gambar 4.7 merupakan proses penentuan tingkat visibilitas jarak antar titik. Visibilitas tersebut berasal dari *invers* dari setiap jarak yang ada antara titik-titik koordinat yang saling berhubungan. Sehingga nantinya akan digunakan sebagi salah satu parameter keputusan semut dalam memilih rute selanjutnya.

```
for i=1:iter
    for i=1:m
        app(i,1)=1; %semua semut mulai dari kota 1
    end
    %rute semut
    for i=1:m %untuk semua semut
        cap=0;
        maxcap=Capacity;
        time=0;
        mh = h; % matriks invers jarak
        indpilih=(1:n);
```

Gambar 4.8 Iterasi serta penentuan node awal setiap semut

Pada Gambar 4.8 merupakan permulaan dari iterasi pembentukan rute, dimana setiap semut berada pada titik 1

atau depot. Kemudian rute selanjutnya dipilih oelh semut itu sendiri berdasarkan keputusan serta batasan yang ada.

```
c=app(i,j); %memilih satu kota
mh(:,c)=0; %jk sdh dipilih makan inv distance=0
temp=(t(c,:).^beta).*(mh(c,:).^alpha); %hitung jml pheromone
s=(sum(temp)); %jml tho
p=(1/s).*temp; %probabilitas
```

Gambar 4.9 Local updating

Pada Gambar 4.9 merupakan aturan *local updating* yang diterapkan pada program. Aturan tersebut berfungsi untuk menentukan jumlah *pheromone* sehingga dapat digunakan sebagai salah satu variabel keputusan untuk menentukan rute selanjutnya.

```
for k=1:n
    s=s+p(k);
    jalan=newjarak(c,k);
    time=timetemp+jalan;
    muat=d(k,1);
    cap=captemp+muat;

    if r<=s && indpilih(k,2)==0
        if time>=ei(k,1) && time<=li(k,1)
            if cap<=maxcap
                time=time+si(1,k);
                app(i,j+1)=k; %penempatan semut i di simpul berikutnya
                break
            else
                captemp=0;
                timetemp=0;
                s=0;
                k=1;
            end
        end
    end
```

Gambar 4.10 Penentuan semut dalam memilih rute

Pada Gambar 4.10 merupakan alur untuk menentukan keputusan rute selanjutnya berdasarkan parameter yang telah ditentukan terlebih dahulu. Setiap satu semut akan berjalan secara terus menerus hingga dipastikan bahwa seluruh titik telah dikunjungi serta mendapatkan banyak rute berdasarkan banyak semut.

```
if Ti(hh,lop)(next)<li(look(1,cc+1)) && carry(hh,lop)+d(look(1,cc+1))<=Q %d = demand
    carry(hh,lop)= carry(hh,lop)+d(look(1,cc+1));
    subrute(hh,lop)(next)=look(1,cc+1);
    zz2(hh,lop)=zz2(hh,lop)+distance(subrute(hh,lop)(next-1),subrute(hh,lop)(next));
else

    subrute(hh,lop)(next)=1;
```

Gambar 4.11 Pembagian rute berdasarkan batasan

Pada Gambar 4.11 merupakan proses pembagian rute berdasarkan batasan kapasitas dan waktu. Pembagian rute tersebut diterapkan pada seluruh semut pada satu iterasi. Sehingga dapat memunculkan jumlah kemungkinan kendaraan yang digunakan serta jarak yang ditempuh untuk setiap rute yang ada.

```
%menentukan rute terbaik
Rute= TotalJarak;
[rute_minimal,minIndex]=min(Rute);
bestrute = subrute(minIndex,:);
```

Gambar 4.12 Menentukan rute terbaik pada satu iterasi

Setelah mendapatkan total jarak pada masing-masing rute maka proses selanjutnya yaitu penentuan rute terbaik. Pada Gambar 4.12 merupakan proses penentuan rute terbaik dengan mencari jarak terpendek setiap rute yang dilalui pada satu iterasi.

```
if waktu(b,2)>=waktu(a,1)
    combine=[combine,b]; %menambah kombinasi semut
    waktu(a,1)=waktu(b,1);
    waktu(b,3)=2;
else
    if waktu(b,1)<=waktu(a,2)
        combine=[b,combine]; %menambah kombinasi semut
        waktu(a,2)=waktu(b,2);
        waktu(b,3)=2;
    end
```

Gambar 4.13 Tabu search

Setelah mendapatkan rute terbaik dengan total jarak serta jumlah kendaraan sementara yang digunakan. Maka pada Gambar 4.13 merupakan proses *tabu search* dimana mengkombinasikan beberapa kendaraan guna mendapatkan hasil yang lebih optimal.

```
for i=1:m
    for j=1:n
        dt=1/f(i);
        t(at(i,j),at(i,j+1))=(1-e)*t(at(i,j),at(i,j+1))+dt;
    end
end
```

Gambar 4.14 Global updating

Pada tahapan selanjutnya ialah *global updating* yang ditunjukkan pada Gambar 4.14 untuk seluruh titik, guna mendapatkan *update* visibilitas *pheromone* untuk menentukan rute pada iterasi selanjutnya.

```
tabuit=sum(tabuiter);

if tabuit<bestsol
    bestsol=tabuit;
    globsol=newglobsol;
end
```

Gambar 4.15 Menentukan rute paling optimal

Pada akhir iterasi terdapat sebuah keputusan pemilihan rute terbaik seperti pada Gambar 4.15 dengan membandingkan setiap rute terbaik pada setiap iterasi. Sehingga hasil akhir yang muncul merupakan rute yang terbaik dari yang terbaik.

4.3.4. Output Hasil

Dari beberapa proses yang telah perjalan pada program maka *output* yang dihasilkan yaitu: total jarak minimal, jumlah kendaraan yang digunakan, serta rute yang ditempuh setiap kendaraan tersebut. Hasil yang paling optimal akan dimunculkan sesuai dengan batasan-batasan yang telah ditentukan. Sehingga dapat dibandingkan dengan solusi eksisting yang ada pada data set tersebut.

4.4. Uji Validasi

Sebelum melakukan pengolahan data menggunakan algoritma ACOTS dengan menggunakan keseluruhan data, algoritma pada *software* akan diuji terlebih dahulu dengan menggunakan data set dengan skala kecil terlebih dahulu untuk menunjukkan apakah algoritma dapat menyelesaikan permasalahan dengan benar. Data yang digunakan untuk pengujian ini yaitu data yang terdiri dari 1 depot dan 5 *customer* dengan batasan yang sama dengan data set yang ada yaitu maksimal 25 kendaraan dengan kapasitas setiap kendaraan 200 satuan berat serta *time windows* masing-masing *customer*. Berikut adalah data set yang digunakan untuk melakukan uji validasi.

Tabel 4.2 Jarak antar titik

	1	2	3	4	5	6
1	0,00	18,68	20,62	16,12	18,11	15,13
2	18,68	0,00	2,00	3,61	3,00	4,24
3	20,62	2,00	0,00	5,00	3,61	5,83
4	16,12	3,61	5,00	0,00	2,00	1,00
5	18,11	3,00	3,61	2,00	0,00	3,00
6	15,13	4,24	5,83	1,00	3,00	0,00

Pada Tabel 4.2 merupakan tabel jarak antar titik yang awalnya merupakan titik koordinat (x,y) yang diproses hingga menjadi tabel jarak antar titik. Tabel tersebut yang digunakan unruk mengukur berapa jarak yang ditempuh pada setiap rute yang terpilih, sehingga dapat dibandingkan rute mana yang memiliki solusi optimal, yaitu jarak yang minimal.

Tabel 4.3 Data permintaan serta konstrain time windows

Cust. No	Demand	Ready Time	Due Date	Service Time
1	0	0	1236	0
2	10	912	967	90
3	30	825	870	90
4	10	65	146	90
5	10	727	782	90
6	10	15	67	90

Pada Tabel 4.3 merupakan data permintaan serta batasan waktu dan *service time*. Batasan tersebut digunakan untuk membatasi pergerakan semut yang bertujuan agar setiap perjalanan semut tidak melanggar konstrain yang ada. Sehingga solusi yang dihasilkan adalah solusi optimal dengan tidak melanggar suatu batasan apapun.

4.4.1. Penyelesaian dengan data set skala kecil

Pada subbab ini akan dilakukan perhitungan menggunakan data kecil, hasil dari perhitungan manual ini akan menghasilkan jarak distribusi minimum. Berikut adalah tahapan perhitungannya:

Langkah 1:
Rute diawali dari depot (Cust. 1), kemudian dihitung waktu sampai ke masing-masing kustomer. Setelah itu dilakukan pengecekan konstrain *time windows* pada setiap kustomer. Kemudian dipilih kustomer yang memiliki waktu sampai yang paling kecil seperti pada Tabel 4.4.

Tabel 4.4 Penentuan rute langkah 1

Pos	Time Start	Next Cust.	*Demand*	Waktu Sampai	Waktu Pergi	*Time Win.*	Cap.
Depot	0	2	10	18,68	108,68	tidak	ya
		3	30	20,62	110,62	tidak	ya
		4	10	16,12	106,12	tidak	ya
		5	10	18,11	108,11	tidak	ya
		6	10	15,13	105,13	ya	ya

Pada langkah 1, kustomer yang dipilih adalah kustomer 6, sehingga rute sementara adalah 1-6. Dari rute sementara tersebut didapatkan *demand* sementara yang diangkut adalah **10** dengan total jarak **15,13** serta waktu **15,13**.

Langkah 2:

Pada langkah sebelumnya telah terpilih kustomer 6. Kemudian dilakukan perhitungan waktu sampai dari kustomer yang terpilih menuju kustomer yang belum terpilih. Kemudian melakukan pengecekan terhadap *time windows* pada Tabel 4.5 untuk setiap kustomer yang akan dikunjungi dan dipilih kustomer yang memiliki waktu paling kecil.

Tabel 4.5 Penentuan rute langkah 2

Pos	Time Start	Next Cust.	*Demand*	Waktu Sampai	Waktu Pergi	*Time Win.*	Cap.
6	105,1	2	20	109,37	199,37	tidak	ya
		3	40	110,96	200,96	tidak	ya
		4	20	106,13	196,13	ya	ya
		5	20	108,13	198,13	tidak	ya

Pada tahapan ini yang terpilih adalah kustomer 4, sehingga rute sementara kali ini adalah 1-6-4 dengan *demand* yang diangkut adalah **20** dengan total jarak **16,13** serta waktu yang dibutuhkan adalah **106,13**.

Langkah 3:
Melakukan pengecekan terhadap kapasitas serta waktu yang dibutuhkan untuk melakukan tahap selanjutnya. Jika masih bisa, maka lakukan seperti langkah 2 hingga semua kustomer terkunjungi tanpa melanggar konstrain hingga kembali ke depot.

Tabel 4.6 Penentuan rute langkah 3

Pos	Time Start	Next Cust.	*Demand*	Waktu Sampai	Waktu Pergi	*Time Win.*	Cap.
4	196,1	2	30	199,74	289,74	tidak	ya
		3	50	201,13	291,13	tidak	ya
		5	30	198,13	288,13	tidak	ya

Pada Tabel 4.6 konstrain waktu tidak dapat terpenuhi karena kendaraan datang terlalu cepat dan tidak ada kustomer yang memenuhi *time windows*. Sehingga perlu dilakukan proses lebih lanjut untuk mencari jarak minimal serta selisih *time windows* yang kemudian menjadi *waiting time*.

Tabel 4.7 Perbandingan jarak dan waktu langkah 3

Cust.	Jarak	Waktu Sampai	Ready	End	Selisih
2	3,61	199,74	912	967	712,26
3	5,00	201,13	825	870	623,87
5	2,00	198,13	727	782	528,87

Pada Tabel 4.7 dapat dilihat bahwa jarak minimal dari titik sebelumnya menuju titik selanjutnya terdapat pada *cust.* 5. Selain itu selisih minimal antara waktu sampai dengan *ready time* terdapat pada nomor 5 pula. Sehingga rute terpilih untuk sementara adalah 1-6-4-5 dengan total jarak **18,13** serta waktu sampai **198,13** dan waktu tunggu **528,87**.

Langkah 4

Lakukan proses yang sama seperti sebelumnya sehingga didapat data sebagai berikut

Tabel 4.8 Penentuan rute langkah 4

Pos	Time Start	Next Cust.	*Demand*	Waktu Sampai	Waktu Pergi	*Time Win.*	Cap.
5	817	2	40	820,00	910,00	tidak	ya
		3	60	820,61	910,61	tidak	ya

Pada Tabel 4.8 konstrain waktu tidak dapat terpenuhi sehingga perlu dilakukan proses lebih lanjut untuk mencari jarak minimal serta selisih *time windows*.

Tabel 4.9 Perbandingan jarak dan waktu langkah 4

Cust.	Jarak	Waktu Sampai	Ready	End	Selisih
2	3,00	820,00	912	967	92,00
3	3,61	820,61	825	870	4,39

Pada Tabel 4.9 dapat dilihat bahwa jarak minimal dari titik sebelumnya menuju titik selanjutnya terdapat pada *cust.* 2. Namun selisih minimal antara waktu sampai dengan *ready time* terdapat pada nomor 3. Dikarenakan belum terdapat solusi yang pasti, maka dilakukan perbandingan ulang dengan menambahkan *service time* pada setiap kustomer. Kemudian dipilih nilai minimal dari selisih antara ***End – (Waktu Sampai + Service Time)***. Sehingga rute terpilih untuk sementara adalah 1-6-4-5-3 dengan total jarak **21,74** serta waktu sampai **820,61** dan waktu tunggu **4,39**.

Langkah 5

Selanjutnya hanya tersisa satu kustomer yang belum dikunjungi dan dapat dilihat pada tabel berikut

Tabel 4.10 Penentuan rute langkah 5

Pos	Time Start	Next Cust.	*Demand*	Waktu Sampai	Waktu Pergi	*Time Win.*	Cap.
3	910,6	2	70	912,61	1002,61	ya	ya

Dari Tabel 4.10 semua konstrain terpenuhi sehingga dapat langsung dipilih bahwa rute sementara adalah 1-6-4-5-3-2 dengan total jarak **23,74** serta waktu sampai **912,61**. Dikarenakan semua kustomer telah dikunjungi, maka semut akan kembali ke depot sehingga data skala kecil ini didapatkan rute optimal adalah 1-6-4-5-3-2-1 dengan jarak **42,52** satuan jarak serta waktu keseluruhan adalah **1019,29** serta total waktu tunggu yaitu **533,26** satuan waktu.

4.4.2. Perhitungan menggunakan *software*

Data set yang digunakan dalam perhitungan manual akan diselesaikan menggunakan algoritma ACOTS pada *software*. Berikut adalah langkah-langkah pengerjaannya:

Langkah 1:

Berikut adalah parameter yang digunakan untuk inisialisasi awal algoritma ACOTS:

α	= 1	(parameter pengendali intensitas *pheromone*)
β	= 3	(parameter pengendali visibilitas)
e	= 0.5	(parameter penguapan *pheromone*)
m	= 25	(jumlah maksimal semut)
iter	= 100	(jumlah maksimal iterasi)
Cap	= 200	(jumlah kapasitas maksimal *m*)
q0	= 0.96	(koefisien cost elemination)

Tingkat *pheromone* awal

```
t =

    0.0100    0.0100    0.0100    0.0100    0.0100    0.0100
    0.0100    0.0100    0.0100    0.0100    0.0100    0.0100
    0.0100    0.0100    0.0100    0.0100    0.0100    0.0100
    0.0100    0.0100    0.0100    0.0100    0.0100    0.0100
    0.0100    0.0100    0.0100    0.0100    0.0100    0.0100
    0.0100    0.0100    0.0100    0.0100    0.0100    0.0100
```

Gambar 4.16 Tingkat pheromone awal

Pada Gambar 4.16, peneliti menggunakan nilai *pheromone* awal yaitu 0.01 untuk semua titik. Sehingga pada penentuan rute awal, semut hanya memilih berdasarkan tingkat visibilitas jarak antar titik yang akan dituju. Selanjutnya akan dilakukan pembaruan nilai *pheromone* berdasarkan rute yang dipilih oleh semut dengan *local updating* maupun *global updating* guna menunjang faktor pemilihan jalur untuk semut itu sendiri.

Visibilitas jarak ($\eta_{ij} = 1/d_{ij}$)

```
h =

         0    0.0526    0.0476    0.0625    0.0556    0.0667
    0.0526         0    0.5000    0.2500    0.3333    0.2500
    0.0476    0.5000         0    0.2000    0.2500    0.1667
    0.0625    0.2500    0.2000         0    0.5000    1.0000
    0.0556    0.3333    0.2500    0.5000         0    0.3333
    0.0667    0.2500    0.1667    1.0000    0.3333         0
```

Gambar 4.17 Visibilitas jarak antar titik

Visibilitas jarak antar titik dihitung berdasarkan invers dari jarak antar titik itu sendiri. Seperti pada Gambar 4.17, menunjukkan nilai visibilitas tersebut untuk semua titik yang ada.

Langkah 2:

Jalankan fungsi *acotsvrptw* pada *command window* Matlab dengan memberikan parameter jumlah semut, iterasi, dan kapasitas. Sehingga didapatkan hasil sebagai berikut:

```
>> acotsvrptw(25,100,200)

bestsol =

    43


ans =

    [1x7 double]
```

Gambar 4.18 Hasil running data set kecil

Dari Gambar 4.18 tedapat variabel bestsol yang menunjukkan jarak minimal yang ditempuh oleh semut yaitu 43 satuan jarak. Sedangkan variabel ans menunjukkan bahwa hanya membutuhkan 1 semut untuk melakukan perjalanan menuju semua kustomer dengan tanpa melanggar konstrain. Berikut adalah rute yang dipilih oleh fungsi *acotsvrptw*.

ans{1,1} <1x7 double>

	1	2	3	4	5	6	7
1	1	6	4	5	3	2	1

Gambar 4.19 Solusi optimal data set kecil

Hasil rute yang didapat pada program tertera pada Gambar 4.19 yaitu dengan rute 1-6-4-5-3-2-1 dengan menggunakan 1 *vehicle* serta menempuh jarak 43 satuan jarak.

4.5. Perhitungan Error

Setelah melakukan uji validasi terhadap program, maka dibutuhkan pengujian pula terhadap solusi yang dihasilkan. Hasil yang dikeluarkan oelh program dibandinkan dengan hasil perhitungan manual maupun eksisteing yang ada. Perbandingan yang dilakukan menggunakan rumus *absolute presentate error*:

$$\text{Error} = \left(\frac{|x-y|}{x}\right) 100\%$$

Dimana:

x = total jarak eksisting

y = total jarak yang dihasilkan program

Sehingga jika dilakukan proses perbandingan untuk data skala kecil dengan nilai eksisting **42.52** dan hasil dari program **43** satauan jarak, maka error yang dihasilkan adalah **1.13%**. Dilihat dari nilai error yang dihasilkan, dapat dikatakan bahwa program dapat berjalan dengan baik, sehingga dilakukan percobaan dengan data set yang lebih besar.

4.6. Implementasi pada C101.25

Pada subbab ini akan dilakukan pengujian kembali menggunakan 25 titik dan 1 depot dari data set yang ada. Dimana hasil dari pengujian tersbeut akan dibandingkan dengan solusi eksisting yang sudah ada.

4.6.1. Parameter input model

Parameter yang digunakan untuk implementasi program pada data set tertera pada Tabel 4.11 berikut:

Tabel 4.11 Parameter input pada model 25 titik

Parameter Input	Nilai
Jumlah Semut (m)	25
Jumlah titik (n)	26
Kapasitas kendaraan (Capacity)	200
Pheromone awal (t)	0.01
Evaporasi (e)	0.5
Qo	0.96
α (alpha)	1
β (beta)	3
Jumlah iterasi	100
Demand	Terdapat di Lampiran 1
Time Windows	Terdapat di Lampiran 1
Service Time	Terdapat di Lampiran 1
Jarak	Terdapat di Lampiran 1

Pada pengujian untuk 25 titik dan 1 depot ini menggunakan 25 semut yang akan melakukan pencarian rute. Dengan masing-masing semut memiliki kapasitas angkut 200 satuan berat. Pada pengujuan kali ini menggunakan 100 kali iterasi untuk mengoptimalkan hasil pencarian rute. Serta parameter lainnya yang telah ditentukan dalam data set sebelumnya.

4.6.2. Hasil imlementasi pada C101.25

Model *software* yang telah dibuat akan diimplementasikan menggunakan data set yang telah diproses sebelumnya. *Output* dari model *software* adalah jumlah kendaraan yang digunakan, total jarak keseluruhan, dan rute setiap kendaraan.

```
bestsol =

  191.8136


ans =

    [1x13 double]
    [1x8  double]
    [1x10 double]
```

Gambar 4.20 Hasil pengujian dengan 25 titik

Dari Gambar 4.20 di atas terdapat informasi bahwa solusi optimal adalah **191.82** satuan jarak serta menggunakan 3 *vehicles* untuk menempuh jarak tersebut. Detail tujuan untuk masing-masing kendaraan adalah sebagai berikut:

ans{2,1} <1x13 double>

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	6	4	8	9	11	12	10	7	5	3	2	1

Gambar 4.21 Hasil rute vehicle 1

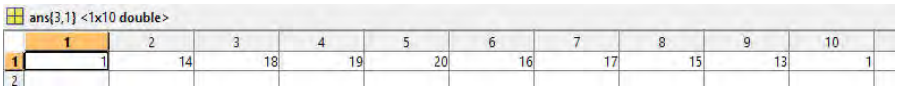
Pada Gambar 4.21 ans{1,1} merupakan rute yang dilalu oleh kendaraan 1 yaitu **1-6-4-8-9-11-12-10-7-5-3-2-1**. Dengan total beban yang diangkut oleh kendaraan 2 adalah **160** satuan berat. Serta jarak tempuh **59.49** satuan jarak dengan total waktu tempuh adalah **1049.50** satuan waktu.

ans{1,1} <1x8 double>

	1	2	3	4	5	6	7	8
1	1	21	25	26	24	23	22	1
2								

Gambar 4.22 Hasil rute vehicle 2

Pada Gambar 4.22 ans{2,1} waktumerupakan rute yang dilalu oleh kendaraan 1 yaitu **1-21-25-26-24-23-22-1**. Dengan total beban yang diangkut oleh kendaraan 1 adalah **100** satuan berat. Serta jarak tempuh **36.44** satuan jarak dengan total waktu tempuh adalah **1017.20** serta memiliki waktu tunggu **440.76** satuan waktu.



Gambar 4.23 Hasil rute vehicle 3

Pada Gambar 4.23 ans{3,1} merupakan rute yang dilalu oleh kendaraan 1 yaitu **1-14-18-19-20-16-17-15-13-1**. Dengan total beban yang diangkut oleh kendaraan 3 adalah **190** satuan berat. Serta jarak tempuh **95.89** satuan jarak dengan total waktu tempuh adalah **815.89** satuan waktu.

Dari keseluruhan kendaraan memiliki total jarak tempuh **191.82** satuan jarak. Jika dibandingkan dengan solusi eksisting pada data set dengan total jarak **191.30** maka tingkat % error pada *sotfware* adalah **0.27%**.

BAB V
HASIL DAN PEMBAHASAN

5.1. Analisis Hasil Validasi Algoritma

Tujuan validasi adalah untuk memastikan bahwa program dapat berjalan sesuai dengan tujuan semula sehingga dapat memberikan *output* yang tepat. Salah satu cara validasi adalah membandingkan hasil perhitungan program dengan perhitungan manual pada data set skala kecil. Dari hasil uji validasi tersebut, dapat diketahui bahwa hasil perhitungan manual memberikan solusi yang sama optimal dengan hasil yang diberikan oleh algoritma ACOTS pada program. Solusi yang dihasilkan baik secara manual maupun model *software* yaitu dengan rute **1-6-4-5-3-2-1** serta dengan jarak **42,52** jika dibulatkan menjadi **43** satuan jarak. Dari uji validitas tersebut dapat disimpulkan bahwa model *software* yang dibuat sudah valid dan dapat digunakan untuk melakukan pengujian data set dalam skala yang lebih besar.

5.2. Analisis Hasil Implementasi Program

Sebelumnya telah ditentukan parameter input yang digunakan untuk melakukan pengolahan data yaitu α=1, β=3, q0=0.96, dan e=0.5. Parameter inilah yang dijadikan bahan untuk menunjang data lain guna mendapatkan rute terbaik. Data lain yang digunakan yaitu koordinat masing-masing titik yang diolah menjadi satu titik tertentu sehingga dapat diketahui jarak antar titik baik depot-kustomer maupun kustomer-kustomer. Jumlah *vehicle* dan jumlah maksimal kapasitas menjadi salah satu konstrain penentuan rute. Selain itu data *demand, time windows,* dan *service time* sesuai dengan data set yang ada. Pada data set diketahui bahwa perbandingan jarak dan waktu adalah 1:1 sehingga kecepatan kendaraan saat melakukan perjalanan adalah 1 pula. Jumlah data secara keseluruhan adalah 1 depot dengan 100 kustomer yang harus dikunjugi semua. Output yang dihasilkan adalah total jarak serta rute maupun subrute yang terbaik.

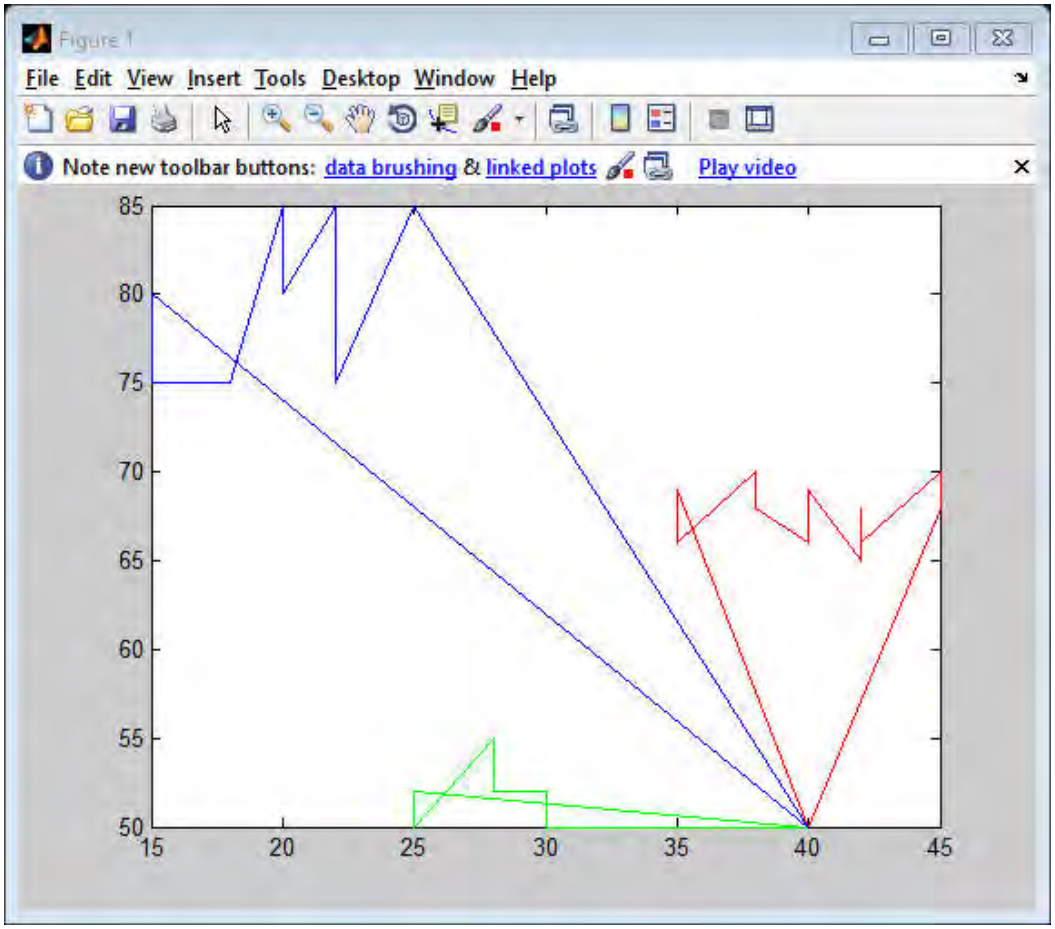
5.2.1. Hasil implementasi pada C101.25

Sebelum telah dilakukan implementasi program terhadap data set C101.25 yang berjalan dengan baik. Dari hasil yang diperolah keseluruhan kendaraan memiliki total jarak tempuh **191.82** satuan jarak serta menggunakan 3 *vehicle* untuk menempuh rute tersebut.

Tabel 5.1 Detail solusi rute C101.25

NV	Rute	Jarak	Waktu Tempuh	Waktu Tunggu
1	1-6-4-8-9-11-12-10-7-5-3-2-1	59.49	1049.50	0
2	1-21-25-26-24-23-22-1	36.44	1017.20	440.76
3	1-14-18-19-20-16-17-15-13-1	95.89	815.89	0
Total		191.82	2882.59	440.76

Pada Tabel 5.1 dapat diketahui bahwa untuk menempuh semua rute yang ada dibutuhkan 3 *vehicles*. Selain itu total jarak tempuh minimal yang didapat yaitu **191.82** dengan total waktu tempuh **2882.59** serta memiliki waktu tunggu **440.76** satuan waktu. Jika dibandingkan dengan solusi eksisting pada data set dengan total jarak **191.30** maka tingkat % error pada *sotfware* adalah **0.27%**.



Gambar 5.1 Plot rute terpilih pada C101.25

Pada Gambar 5.1 merupakan visualisasi rute yang dipilih oleh program ketika dijalankan pada data set C101.25. Menurut gambar tersebut, terdapat tiga jenis garis yang disimbolkan dengan berbagai warna yang mewakili setiap *vehicle* serta rutenya. Mulai dari *vehicle* pertama dengan warna merah, kedua dengan warna hijau, dan ketiga dengan warna biru.

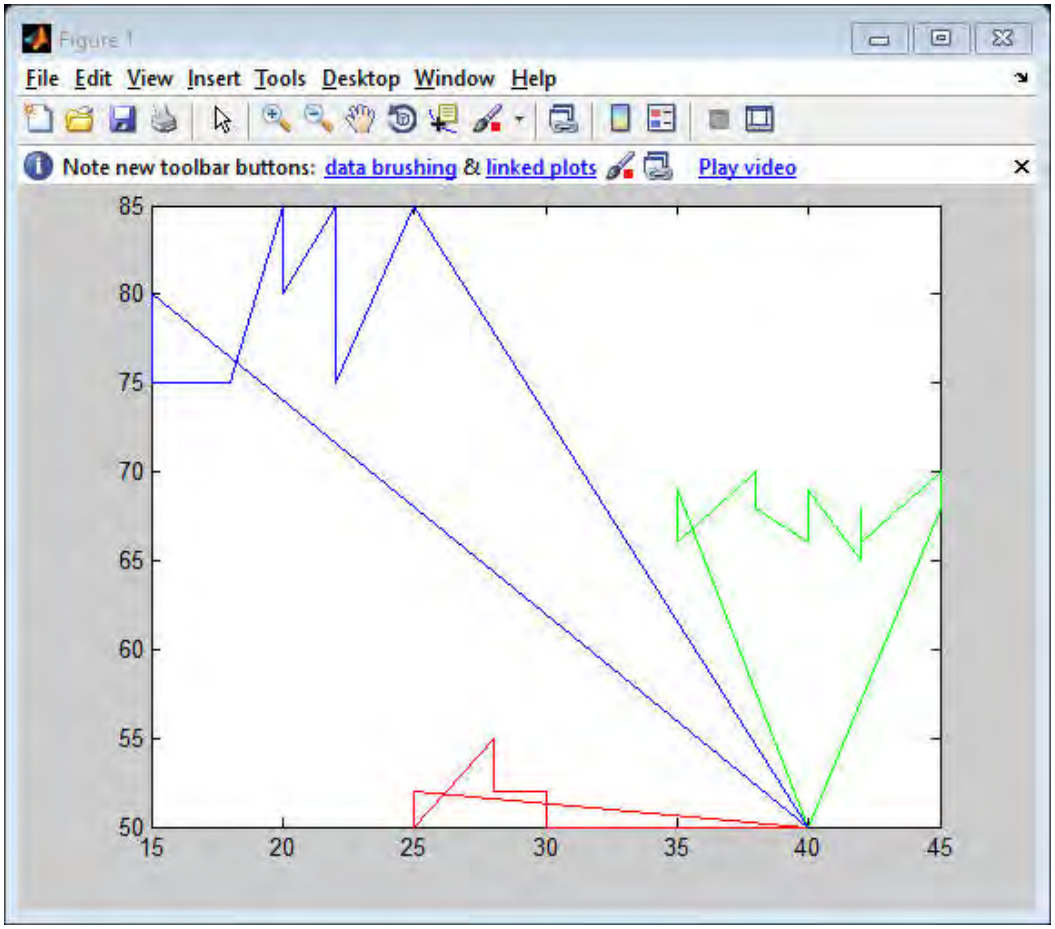
5.2.2. Hasil implementasi pada C105.25

Setelah melakukan implementasi pada data set C101.25, maka dilanjutkan mengimplementasikan program pada data set C105.25 dengan langkah serta parameter yang sama dengan implementasi C101.25. Dari hasil yang diperolah keseluruhan kendaraan memiliki total jarak tempuh **191.82** satuan jarak serta menggunakan 3 *vehicle* untuk menempuh rute tersebut.

Tabel 5.2 Detail solusi rute C105.25

NV	Rute	Jarak	Waktu Tempuh	Waktu Tunggu
1	1-21-25-26-24-23-22-1	36.44	994.20	417.76
2	1-6-4-8-9-11-12-10-7-5-3-2-1	59.49	1049.50	0
3	1-14-18-19-20-16-17-15-13-1	95.89	815.89	0
Total		191.82	2859.59	417.76

Pada Tabel 5.2 dapat diketahui bahwa untuk menempuh semua rute yang ada dibutuhkan 3 *vehicles*. Selain itu total jarak tempuh minimal yang didapat yaitu **191.82** dengan total waktu tempuh **2859.59** serta memiliki waktu tunggu **417.76** satuan waktu. Jika dibandingkan dengan solusi eksisting pada data set dengan total jarak **191.30** maka tingkat % error pada *sotfware* adalah **0.27%**. Terdapat persamaan total jarak temmpuh yang dilalui program, hal tersebut disebabkan karena titik koordinat kustomer pada C105.25 sama dengan C101.25. Hal yang membedakan dari kedua data set tersebut adalah *time windows* yaitu waktu minimal dan maksimal kendaraan tiba pada titi tersebut. Sehingga terdapat perbedaan total waktu tempuh serta waktu tunggu antara data set C101.25 dan C105.25.



Gambar 5.2 Plot rute terpilih pada C105.25

Pada Gambar 5.2 merupakan visualisasi rute yang dipilih oleh program ketika dijalankan pada data set C105.25. Menurut gambar tersebut, terdapat tiga jenis garis yang disimbolkan dengan berbagai warna yang mewakili setiap *vehicle* serta rutenya. Mulai dari *vehicle* pertama dengan warna merah, kedua dengan warna hijau, dan ketiga dengan warna biru.

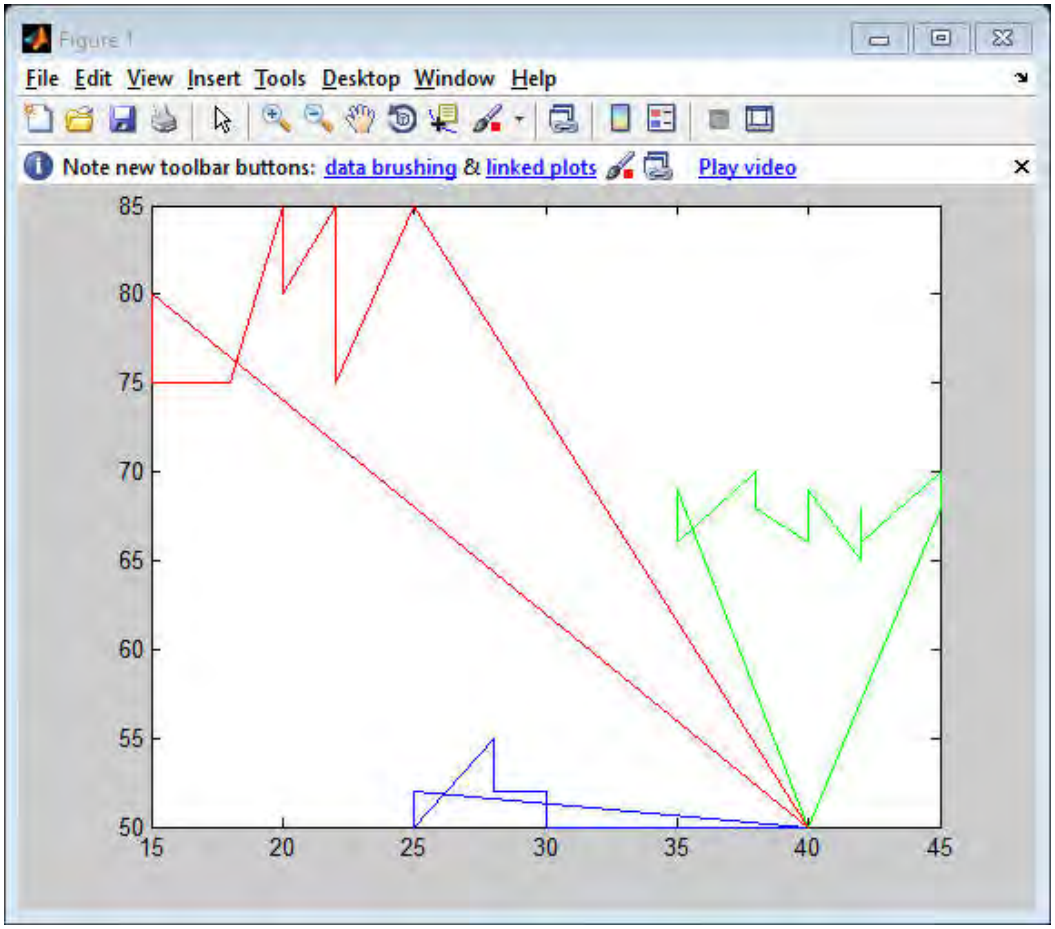
5.2.3. Hasil implementasi pada C106.25

Melakukan implementasi program pada data lain yang memiliki sifat serupa dengan data C105.25, yaitu C106.26. Pada data set C106.25 tersebut juga memiliki persamaan koordinat dengan data set C101.25 maupun C105.25 namun memiliki perbedaan *time windows* pada setiap titiknya. Dari hasil yang diperolah keseluruhan kendaraan memiliki total jarak tempuh **191.82** satuan jarak serta menggunakan 3 *vehicle* untuk menempuh rute tersebut.

Tabel 5.3 Detail solusi rute C106.25

NV	Rute	Jarak	Waktu Tempuh	Waktu Tunggu
1	1-14-18-19-20-16-17-15-13-1	95.89	815.89	0
2	1-6-4-8-9-11-12-10-7-5-3-2-1	59.49	1049.50	0
3	1-21-25-26-24-23-22-1	36.44	1024.20	447.76
Total		191.82	2889.59	447.76

Pada Tabel 5.3 dapat diketahui bahwa untuk menempuh semua rute yang ada dibutuhkan 3 *vehicles*. Selain itu total jarak tempuh minimal yang didapat yaitu **191.82** dengan total waktu tempuh **2889.59** serta memiliki waktu tunggu **447.76** satuan waktu. Jika dibandingkan dengan solusi eksisting pada data set dengan total jarak **191.30** maka tingkat % error pada *sotfware* adalah **0.27%**.



Gambar 5.3 Plot rute terpilih pada C106.25

Pada Gambar 5.3 merupakan visualisasi rute yang dipilih oleh program ketika dijalankan pada data set C106.25. Menurut gambar tersebut, terdapat tiga jenis garis yang disimbolkan dengan berbagai warna yang mewakili setiap *vehicle* serta rutenya. Mulai dari *vehicle* pertama dengan warna merah, kedua dengan warna hijau, dan ketiga dengan warna biru.

LAMPIRAN 1

C101.25

VEHICLE NUMBER	CAPACITY
25	200

CUSTOMER

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

NO.	X	Y	D	READY TIME	DUE DATE	SERVICE TIME
19	15	75	20	179	254	90
20	15	80	10	278	345	90
21	30	50	10	10	73	90
22	30	52	20	914	965	90
23	28	52	20	812	883	90
24	28	55	10	732	777	90
25	25	50	10	65	144	90
26	25	52	40	169	224	90

LAMPIRAN 2

C105.25

VEHICLE NUMBER	CAPACITY
25	200

CUSTOMER

NO.	X	Y	D	READY TIME	DUE DATE	SERVICE TIME
1	40	50	0	0	1236	0
2	45	68	10	885	994	90
3	45	70	30	802	893	90
4	42	66	10	25	186	90
5	42	68	10	699	810	90
6	42	65	10	15	120	90
7	40	69	20	580	743	90
8	40	66	20	142	253	90
9	38	68	20	220	359	90
10	38	70	10	499	640	90
11	35	66	10	331	436	90
12	35	69	10	420	533	90
13	25	85	20	617	756	90
14	22	75	30	30	155	90
15	22	85	10	541	646	90
16	20	80	40	362	451	90
17	20	85	40	448	555	90
18	18	75	20	75	172	90

NO.	X	Y	D	READY TIME	DUE DATE	SERVICE TIME
19	15	75	20	142	291	90
20	15	80	10	244	379	90
21	30	50	10	10	137	90
22	30	52	20	888	991	90
23	28	52	20	776	919	90
24	28	55	10	709	800	90
25	25	50	10	25	184	90
26	25	52	40	142	251	90

LAMPIRAN 3

C106.25

VEHICLE NUMBER	CAPACITY
25	200

CUSTOMER

NO.	X	Y	D	READY TIME	DUE DATE	SERVICE TIME
1	40	50	0	0	1236	0
2	45	68	10	890	989	90
3	45	70	30	816	879	90
4	42	66	10	55	156	90
5	42	68	10	703	806	90
6	42	65	10	15	60	90
7	40	69	20	559	764	90
8	40	66	20	172	223	90
9	38	68	20	250	329	90
10	38	70	10	489	650	90
11	35	66	10	361	406	90
12	35	69	10	450	503	90
13	25	85	20	647	726	90
14	22	75	30	30	95	90
15	22	85	10	571	616	90
16	20	80	40	392	421	90
17	20	85	40	478	525	90
18	18	75	20	105	142	90

NO.	X	Y	D	READY TIME	DUE DATE	SERVICE TIME
19	15	75	20	172	261	90
20	15	80	10	274	349	90
21	30	50	10	10	77	90
22	30	52	20	918	961	90
23	28	52	20	806	889	90
24	28	55	10	739	770	90
25	25	50	10	55	154	90
26	25	52	40	172	221	90

LAMPIRAN 4

```
function [globsol]=
newacotstime(n_ants,iter,Capacity)

filename='C105_25.xlsx';
distance=xlsread(filename,'jarak');
d=xlsread(filename,'demand');
si=xlsread(filename,'loading');
li=xlsread(filename,'tutup');
ei=xlsread(filename,'buka');

%distance=round(distance);
newjarak=distance;
tprog=cputime;

m = n_ants; %jumlah semut
n = length(distance); %jumlah tujuan
e = 0.5; %evaporation coefficient
alpha=1; %pangkat untuk ants' sight
beta=3; %pangkat untuk trace's effect
qo = 0.96; %koefisien cost elimination

%menghitung visibility
for i=1:n
    for j=1:n
        if distance(i,j)==0
            h(i,j)=0;
        else
            h(i,j) = 1/distance(i,j); %inverse
distance
        end
    end
end


T=distance;
si=transpose(si); %waktu loading unloading
```

```
t=0.01*ones(n); %pheromone awal
bestsol=999999;
globsol={};
for i=1:iter
    for i=1:m
        app(i,1)=1; %semua semut mulai dari
kota 1
    end
    %rute semut
    for i=1:m %untuk semua semut

        cap=0;
        maxcap=Capacity;
        time=0;
        mh = h; % matriks invers jarak
        indpilih=(1:n);
        indpilih=transpose(indpilih);
        indpilih(:,2)=zeros(n,1);

        for j=1:n-1 %simpul berikutnya

            c=app(i,j); %memilih satu kota
            mh(:,c)=0; %jk sdh dipilih makan
inv distance=0

temp=(t(c,:).^beta).*(mh(c,:).^alpha); %hitung
jml pheromone
            s=(sum(temp)); %jml tho
            p=(1/s).*temp; %probabilitas

            indpilih(c,2)=1;

            r=rand;
            s=0;
            captemp=cap;
            timetemp=time;

            for k=1:n
                s=s+p(k);
                jalan=newjarak(c,k);
```

```
                    time=timetemp+jalan;
                    muat=d(k,1);
                    cap=captemp+muat;

                    if r<=s && indpilih(k,2)==0
                        if time>=ei(k,1) &&
time<=li(k,1)
                            %if cap<=maxcap
                                time=time+si(1,k);
                                app(i,j+1)=k;
%penempatan semut i di simpul berikutnya
                                break
                            %else
                                %captemp=0;
                                %timetemp=0;
                                %s=0;
                                %k=1;
                            %end
                        end
                    end
                    if k==n || cap>=maxcap
                        cap=0;
                        buka=9999;
                        for v=1:n
                            ind=indpilih(v,2);
                            titik=indpilih(v,1);
                            if ind==0 &&
ei(v,1)<=buka
                               buka=ei(v,1);
                               pilih=titik;
                            end
                        end
                        app(i,j+1)=pilih; %pilih
semut dengan waktu buka terkecil
                        cap=d(pilih,1);
                        time=buka+si(1,pilih);
                        break
                    end
                end
            end
            app(i,n+1)=1;
```

```
    end


    %KONSTRAIN TIME WINDOWS DAN KAPASITAS
    subrute=[];
    at=app; %rute yang sudah ada
    [a,b]=size(at);
    look = [];
    for hh=1:a
        lop=1;
        travel=0;
        carry(hh,lop)=0; %jumla barang yang
dibawa
        next=2;
        zz2(hh,lop)=0; %jarak yang ditempuh
selama perjananan dalam 1 subrute
        travel=ei(1,1); %waktu pergi dari node
sebelumnya
        penalti(hh,lop)=0; %penalti yang akan
didapat
        for cc=1:b-1 %jumlah node
            look(1,:)=at(hh,:);
            if sum(size(look))~=0
                nc=length(look);
                subrute{hh,lop}(1)=1; %subrute
di awali dengan node 1
                Q=Capacity; %kapasitas
kendaraan
                Ti{hh,lop}(next)=travel +
T(look(1,cc),look(1,cc+1)); %waktu datang ke
node selanjutnya

                %KONSTRAIN KAPASITAS & WAKTU
                if
Ti{hh,lop}(next)<li(look(1,cc+1)) &&
carry(hh,lop)+d(look(1,cc+1))<=Q %d = demand
                    carry(hh,lop)=
carry(hh,lop)+d(look(1,cc+1));

subrute{hh,lop}(next)=look(1,cc+1);
```

```
zz2(hh,lop)=zz2(hh,lop)+distance(subrute{hh,lo
p}(next-1),subrute{hh,lop}(next));
                else

                    subrute{hh,lop}(next)=1;

zz2(hh,lop)=zz2(hh,lop)+distance(subrute{hh,lo
p}(next-1),subrute{hh,lop}(next));

Ti{hh,lop}(next)=travel+T(look(1,cc),look(1,cc
+1));
                    lop=lop+1;
                    subrute{hh,lop}(1)=1;
                    travel=0;
                    zz2(hh,lop)=0;

carry(hh,lop)=d(look(1,cc));
                    next=2;
                    travel=ei(1,1);
                    Ti{hh,lop}(next)=travel+
T(look(1,1),look(1,cc+1));
                    subrute{hh,lop}(next) =
look(1,cc+1);

zz2(hh,lop)=zz2(hh,lop)+distance(subrute{hh,lo
p}(next-1),subrute{hh,lop}(next));

carry(hh,lop)=d(look(1,cc+1));
                end

                %KONSTRAIN TIME WINDOWS
                if Ti{hh,lop}(next)<
ei(look(1,cc+1)) %ei(waktu buka kustomer)
                    start=ei(look(1,cc+1));
                else
                    start=Ti{hh,lop}(next);
                end
                if
Ti{hh,lop}(next)>li(look(1,cc+1)) %li adalah
waktu tutup kustomer
```

```
                    start= Ti{hh,lop}(next);
                    penalti(hh,lop)=
(abs(Ti{hh,lop}(next)-li(look(1,cc+1))));
                end
                if cc==b-1
                    subrute{hh,lop}(next)=1;

Ti{hh,lop}(next)=travel+T(look(1,cc),look(1,cc
+1));
                    lop=lop+1;
                    travel=0;
                    zz2(hh,lop)=0;
                end

travel=start+si(1,look(1,cc+1));
                next=next+1;
            end
        end

    end

    %menghitung total jarak yang ditempuh
    yy2=transpose(zz2);
    TotalJarak=sum(yy2);

    %menghitung total pinalti
    aa=transpose(penalti);
    totalpinalti=sum(aa);

    %menentukan rute terbaik
    Rute= TotalJarak;
    [rute_minimal,minIndex]=min(Rute);
    bestrute = subrute(minIndex,:);
    %peny=totalpinalti(minIndex);

    %rute_minimal
    %bestrute

    for i=1:m
        s=0;
```

```
            for j=1:n
                s=s+distance(at(i,j),at(i,j+1));
            end
            f(i)=s;
        end
        cost=f;

        f=f-qo*min(f);

        for i=1:m
            for j=1:n
                dt=1/f(i);
                t(at(i,j),at(i,j+1))=(1-
e)*t(at(i,j),at(i,j+1))+dt;
            end
        end

        %[mincost(i),number]=min(cost);
        %bestrute(i,:)=at(number,:);
        %iteration(i)=i;

        sizeRoute = length(bestrute);
        for z=1:sizeRoute
            sbr=bestrute{1,z};
            lsbr=length(sbr);
            tabuant=[];
            for jr=1:lsbr-1
                lanjut1=sbr(1,jr);
                lanjut2=sbr(1,jr+1);

tabuant(1,jr)=newjarak(lanjut1,lanjut2);
            end
            tabuiter(1,z)=sum(tabuant);
        end
        tabuit=sum(tabuiter);

        if tabuit<bestsol
            bestsol=tabuit;
            globsol=bestrute;
        end
        %end memori
```

```
%TABU
jml=length(globsol);
indmax=0;
for a=1:jml
    tabglob=globsol{1,a};
    ltg=length(tabglob);
    if indmax<=ltg
        indmax=ltg;
    end
end

%TABU2
for a=1:jml
    tabglob=globsol{1,a};
    ltg=length(tabglob);
    beban=0;
    for b=1:ltg
        beban=beban+d(b,1);
    end
    %beban
    if ltg ~= 0
        waktu(a,1)=ei(tabglob(1,ltg-
1))+si(1,ltg-1);%newjarak(tabglob(1,ltg-
1),tabglob(1,ltg));
        waktu(a,2)=li(tabglob(1,2));
        if waktu(a,1)>=(li(1,1)-si(1,2))
%ltg>(indmax*2/3)
            waktu(a,3)=1;
        else
            waktu(a,3)=0;
        end
    end

end
pan=length(waktu(:,1)); %banyaknya semut
sementara
gs=1;

%NextTabu
for a=1:pan
```

```
    combine=[a];
    if waktu(a,3) == 1
        globcom0=globsol{1,a};
        newglobsol{1,gs}=globcom0;
        gs=gs+1;
    else
    if waktu(a,3) == 0
        for b=1:pan
            if a~=b && waktu(b,3)==0
                if waktu(b,2)>=waktu(a,1)
                    combine=[combine,b];
%menambah kombinasi semut
                    waktu(a,1)=waktu(b,1);
                    waktu(b,3)=2;
                else
                    if waktu(b,1)<=waktu(a,2)
                        combine=[b,combine];
%menambah kombinasi semut
                        waktu(a,2)=waktu(b,2);
                        waktu(b,3)=2;
                    end
                end
            end
            waktu(a,3)=1;
        end
        comtb=length(combine);
        globcom0=globsol{1,combine(1,1)};
        if comtb~=1
            lgc0=length(globcom0);
            globcom0=globcom0(1:lgc0-1);
            for com=2:comtb

globcom1=globsol{1,combine(1,comtb)};
                lgc1=length(globcom1);

globcom0=[globcom0,globcom1(2:lgc1-1)];
                nlgc0=length(globcom0);
            end
            globcom0(1,nlgc0+1)=1;
        end
        newglobsol{1,gs}=globcom0;
        gs=gs+1;
```

```
    end

    end
end

sizenew = length(newglobsol);
for z=1:sizenew
    sbr=newglobsol{1,z};
    lsbr=length(sbr);
    tabuant=[];
    for jr=1:lsbr-1
        lanjut1=sbr(1,jr);
        lanjut2=sbr(1,jr+1);

tabuant(1,jr)=newjarak(lanjut1,lanjut2);
    end
    tabuiter(1,z)=sum(tabuant);
end
tabuit=sum(tabuiter);

if tabuit<bestsol
    bestsol=tabuit;
    globsol=newglobsol;
end
%end Tabu

%tabuit
%newglobsol

%end iterasi i
end
tprog
bestsol
globsol=globsol(end,:)';
```

PENGEMBANGAN METODE *ANT COLONY OPTIMIZATION* DAN *TABU SEARCH* UNTUK MENYELESAIKAN *VEHICLE ROUTING PROBLEM WITH TIME WINDOWS*

Wildan Habiby Fanani, Edwin Riksakomara S.Kom., M.T., dan Amalia Utamima S.Kom., M.B.A.
Jurusan Sistem Informasi, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS)
Jl. Arief Rahman Hakim, Surabaya 60111 Indonesia
e-mail: wildan.byfan@outlook.com, utamima@gmail.com, erk@is.its.ac.id

Vehicle Routing Problem with Time Windows (VRPTW) merupakan salah satu dari beberapa permasalahan yang sering terjadi pada suatu sistem dalam penyaluran logistik. Sistem pengiriman barang kepada pelanggan yang terdapat pada e-commerece merupakan bagian dari permasalahan tersebut. Dimana setiap depot yang memiliki sejumlah kendaraan dengan kapasitas tertentu dapat melayani semua customer pada lokasi tertentu. Dengan ketentuan setiap customer memiliki jumlah permintaan serta batasan waktu yang berbeda-beda. Serta dalam setiap pengiriman memiliki tujuan untuk meminimalkan biaya distribusi tanpa mengabaikan batasan yang ada.

VRPTW dapat diselesaikan menggunakan metode eksak, heuristik, maupun meta-heuristik. Dalam tugas akhir ini VRPTW diselesaikan menggunakan Ant Colony Optimization (ACO) yang berdasarkan pada observasi perilaku koloni semut dalam menentukan jalur untuk mencari lokasi makanan yang kemudian disempurnakan menggunakan Tabu Search (TS) dalam pengambilan keputusan.

Penyelesaian VRPTW menggunakan algoritma ACOTS ini diujicobakan pada data set Solomon Problem, yang merupakan standar permasalahan internasional VRPTW. Hasil dari uji coba tersebut menunjukkan bahwa algoritma ACOTS memberikan hasil solusi yang mendekati optimal atau mendekati solusi terbaik pada Solomon Data Set.

***Kata Kunci*— Ant Colony Optimization, Tabu Search, Vehicle Routing Problem with Time Windows.**

1. PENDAHULUAN

Saat ini *e-commerce* di Indonesia mengalami pertumbuhan yang sangat pesat. Mulai dari usaha rumahan hingga toko besar telah membuka lapak online mereka. Bagi konsumen, kehadiran toko online tentu sangat memudahkan mereka agar tidak pelru keluar rumah untuk mendapatkan barang. Selain itu e-commerce erat kaitannya dengan jasa ekspedisi yang akan mengirim barang menuju konsumen. Dengan demikian semakin banyaknya transaksi yang ada, maka semakin banyak pula konsumen yang akan dituju.

Dari sinilah terdapat permasalahan yang mengakibatkan banyaknya keterlambatan pengiriman. Salah satu penyebabnya adalah jalur distribusi barang yang kurang tepat dalam menjalankan proses pengiriman. Disisi lain terdapat beberapa pelanggan yang memiliki batasan waktu penerimaan, sehingga ketika terdapat keterlambatan tersebut akan menimbulkan sebuah kerugian. Hal tersebut dapat digolongkan sebagai Vehicle Routing Problem with Time Windows (VRPTW). Permasalahan tersebut muncul dimana suatu pengiriman harus menentukan siapa pengirimnya dan rute yang dituju dengan melihat batasan kapasitas setiap pengirim serta waktu yang merukapan batasan dalam menentukan solusi rute yang optimal.

Menurut Toth, P. dan Vigo, D [8]., komponen-komponen yang terdapat dalam *Vehicle Routing Problem* adalah:

a. Jaringan jalan, biasanya direpresentasikan dalam sebuag diagram yang terdiri dari *arc* (lengkung/garis atau bagian-bagian jalan) dan *vertex* (titik lokasi *customer* dan depot). Setiap *arc* diasosiasikan dengan jarak, waktu, dan biaya untuk menuju *vertex* tertentu berdasarkan kondisi yang ada (jenis kendaraan, kondisi/karakteristik jalan, dan periode perlintasan).
b. *Customer*, ditandai dengan *vertex* (titik) dan biasanya memiliki hal-hal seperti berikut:
    1. Jumlah permintaan barang
    2. Periode pelayanan
    3. Waktu yang dibutuhkan untuk menurunkan atau memuat barang
    4. Pengelompokan titik
    5. Prioritas
c. Depot, juga ditandai dengan suatu titik yang merupakan awal dan akhir dari sebuah rute.
d. Kendaraan, merupakan alat yang digunakan untuk mendistribusikan barang dan memiliki hal sebagai berikut:
    1. Depot awal/akhir
    2. Kapasitas
    3. Pengelompokan rute
e. Batasan waktu

Glover menyatakan bahwa *Tabu Search* (TS) adalah salah satu prosedur metaheuristik tingkat tinggi untuk penyelesaian permasalahan optimisasi kombinatorial. TS ini dirancang untuk mengarahkan metode-metode lain (atau komponen proses TS itu sendiri) untuk keluar atau menghindari dari masuk dalam solusi optimal yang bersifat lokal. Secara garis besar kemampuan TS dalam menghasilkan solusi yang dapat dikatakan mendekati optimal dalam studi kasus terkentu [13]. Didalam menerapkan Tabu search, ada beberapa bagian yang perlu diperhatikan [14], yaitu:

1. *Tabu list* : tujuan dari penggunaan Tabu list (memori jangka pendek) adalah untuk menghindari mengevaluasi kandidat solusi yang pernah dikunjungi sebelumnya. Akan tetapi, menampung semua kandidat solusi yang pernah dievaluasi kedalam Tabu list akan membuat Tabu search menjadi tidak efektif (memerlukan ukuran memori yang besar dan wakt yang lama untuk mengecek apakah suatu candidate solusi pernah

dievaluasi atau belum).
2. *Aspiration criteria*: umumnya diterapkan jika pergerakan tabu tersebut menghasilkan kandidat solusi yang memiliki nilai yang lebih baik daripada solusi terbaik yang telah dihasilkan.
3. *Intensifikasi* (memori jangka menengah): memori jangka menengah menyimpan sejumlah solusi yang berkualitas (elite solution) yang dihasilkan selama proses pencarian. Memori jangka menengah ini bertujuan untuk memberikan prioritas kepada atribut dari solusi berkualitas tersebut.
4. *Diversifikasi* (memori jangka panjang): memori jangka panjang menyimpan infromasi tentang kandidat solusi yang pernah dikunjungi. Berdasarkan infromasi tersebut, memori ini dapat mengeksplorasi area dalam ruang pencarian yang belum dikunjungi.

2. URAIAN PENELITIAN

A. Identifikasi dan Perumusan Masalah

Pada tahapan pertama ini dilakukan identifikasi serta perumusan masalah yang terkait dengan latar belakang, tujuan serta manfaat yang akan didapat dalam penelitian. Pada tahapan ini diharapkan mampu menjadi awalan dalam memulai penelitian serta secara tidak langsung dapat menjadi masukan bagi tahapan selanjutnya.

B. Tinjauan Pustaka

Pada tahapan ini dilakukan studi literatur terkait penelitian yang telah dilakukan oleh peneliti lain. Studi yang dilakukan terkait permasalahan yang diambil, metode yang digunakan, serta hasil yang didapat. Sehingga diharapkan dapat digunakan sebagai bahan acuan serta perbandingan agar penelitian lebih baik.

C. Pengolahan Data

Data set yang telah diambil akan diproses sesuai dengan kebutuhan penelitian. Sehingga dapat mencakup fungsi tujuan serta batasan yang terdapat pada studi kasus.

D. Pemodelan Algoritma ACOTS untuk VRPTW

Pada tahapan ini adalah memodelkan algoritma gabungan antara Ant Colony Optimizazion dan Tabu Search. Model yang diterapkan adalah flow diagram dari algoritma gabungan yang akan diterapkan oleh peneliti untuk mendapatkan solusi.

E. Pengembangan ACOTS pada Software

Model algoritma yang sudah dibuat sebelumnya, dijadikan acuan untuk pengembangan model pada software. Dimana software yang digunakan adalah MATLAB. Model software ini nantinya yang akan digunakan untuk memproses data yang telah didapat sebelumnya. Input yang dibutuhkan adalah jarak customer, time windows, kapasitas kendaraan, demand, loading/unloading.

F. Pengujian dan Evaluasi

Setelah model software selesai dibuat maka akan dilakukan pengujian. Tujuan pengujian adalah untuk memastikan bahwa program tersebut dapat berjalan serta menghasilkan output yang tepat. Dimana output yang dibutuhkan adalah jumlah pengirim, jarak rute, dan waktu. Kemudian dilakukan evaluasi ketika terdapat permasalahan dalam pelaksanaan pengujian.

G. Pembahasan dan Penarikan Kesimpulan

Pada tahapan ini dilakukan analisis dari hasil running model terkait tingkat optimasi yang didapatkan serta kelebihan dan kekurangan algoritma yang dijalankan. Hasil yang didapat akan dibandingkan dengan solusi yang telah ada pada Solomon Dataset. Penarikan kesimpulan terdiri dari poin-poin yang menjawab tujuan dari penelitian. Selain penarikan kesimpulan diberikan juga saran terkait penelitian yang dilakukan.

3. HASIL DAN DISKUSI

A. Pengolahan Data

Pada subbab ini akan dilakukan pengumpulan informasi terkait data-data yang dibutuhkan. Data-data tersebut diantaranya adalah lokasi titik depot dan *customer*, jumlah maksimal kendaraan, *time windows* depot dan *customer*, serta rute eksisting.

CUST NO.	XCOORD.	YCOORD.	DEMAND	READY TIME	DUE DATE	SERVICE TIME
1	40.00	50.00	0.00	0.00	1236.00	0.00
2	45.00	68.00	10.00	912.00	967.00	90.00
3	45.00	70.00	30.00	825.00	870.00	90.00
4	42.00	66.00	10.00	65.00	146.00	90.00
5	42.00	68.00	10.00	727.00	782.00	90.00
6	42.00	65.00	10.00	15.00	67.00	90.00

Gambar 3.1 Bagian data set

Pada Gambar 3.1 merupakan sebagian data VRPTW yang digunakan dalam pengujian program ACOTS. Terdapat beberapa keterangan yang menunjukkan bagian yang berfungsi sebagai koordinat maupun batasan, yaitu:

CUST NO = Nomor index setiap titik, dimana index 1 merupakan depot dan selain itu merupakan kustomer
XCOORD. = Titik koordinat pada sumbu x
YCOORD. = Titik koordinat pada sumbu y
DEMAND = Jumlah permintaan untuk masing-masing titik
READY TIME = Batas minimal kendaraan sampai pada titik
DUE DATE = Batas maksimal kendaraan sampai pada titik
SERVICE TIME = Waktu yang digunakan untuk bongkar muat pada suatu titik

B. Implementasi Pada C101_25

Pada bagian ini akan dilakukan implementasi menggunakan 25 titik dan 1 depot dari data set yang ada. Dimana hasil dari pengujian tersbeut akan dibandingkan dengan solusi eksisting pada data set yang digunakan. Salah satu langkah implementasi yaitu menentukan parameter yang akan digunakan sesuai dengan data set. Parameter yang digunakan untuk implementasi program pada data set tertera pada Tabel 3.1 berikut:

Tabel 3.1 Parameter input pada model 25 titik

Parameter Input	Nilai
Jumlah Semut (m)	25
Jumlah titik (n)	26
Kapasitas kendaraan (Capacity)	200
Pheromone awal (t)	0.01
Evaporasi (e)	0.5
Qo	0.96
α (alpha)	1
β (beta)	3
Jumlah iterasi	100
Demand	Sesuai Data Set
Time Windows	Sesuai Data Set
Service Time	Sesuai Data Set
Jarak	Sesuai Data Set

Pada pengujian untuk 25 titik dan 1 depot ini menggunakan 25 semut yang akan melakukan pencarian rute. Dengan masing-masing semut memiliki kapasitas angkut 200 satuan berat. Pada pengujuan kali ini menggunakan 100 kali iterasi untuk mengoptimalkan hasil pencarian rute. Serta parameter lainnya yang telah ditentukan dalam data set sebelumnya.

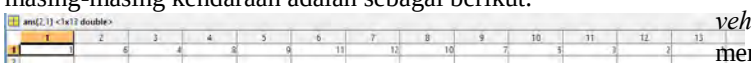
C. *Hasil Implementasi C101_25*

Model *software* yang telah dibuat akan diimplementasikan menggunakan data set yang telah diproses sebelumnya. *Output* dari model *software* adalah jumlah kendaraan yang digunakan, total jarak keseluruhan, dan rute setiap kendaraan.

```
bestsol =

  191.8136


ans =

    [1x13 double]
    [1x8  double]
    [1x10 double]
```

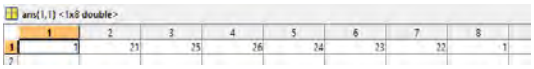
Gambar 3.2 Hasil pengujian dengan 25 titik

Dari Gambar 3.2 di atas terdapat informasi bahwa solusi optimal adalah **191.82** satuan jarak serta menggunakan 3 *vehicles* untuk menempuh jarak tersebut. Detail tujuan untuk masing-masing kendaraan adalah sebagai berikut:



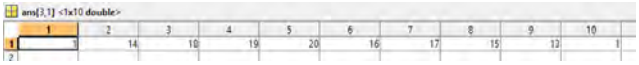
Gambar 3.3 Hasil rute vehicle 1

Pada Gambar 3.3 ans{1,1} merupakan rute yang dilalu oleh kendaraan 1 yaitu **1-6-4-8-9-11-12-10-7-5-3-2-1**. Dengan total beban yang diangkut oleh kendaraan 2 adalah **160** satuan berat. Serta jarak tempuh **59.49** satuan jarak dengan total waktu tempuh adalah **1049.50** satuan waktu.



Gambar 3.4 Hasil rute vehicle 2

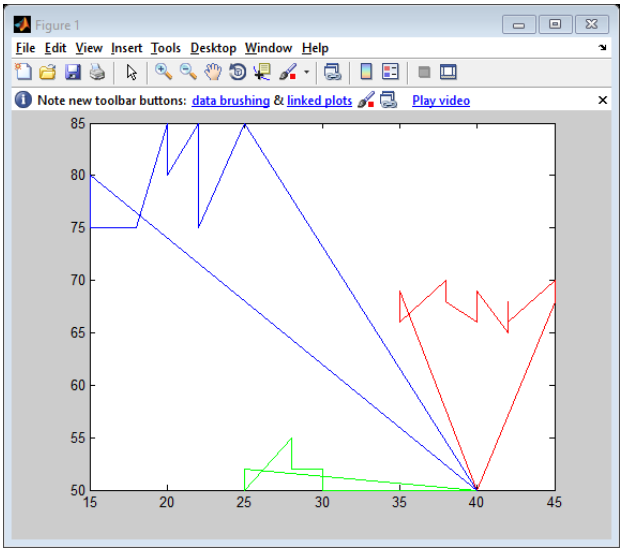
Pada Gambar 3.4 ans{2,1} waktumerupakan rute yang dilalu oleh kendaraan 1 yaitu **1-21-25-26-24-23-22-1**. Dengan total beban yang diangkut oleh kendaraan 1 adalah **100** satuan berat. Serta jarak tempuh **36.44** satuan jarak dengan total waktu tempuh adalah **1017.20** serta memiliki waktu tunggu **440.76** satuan waktu.



Gambar 3.5 Hasil rute vehicle 3

Pada Gambar 3.5 ans{3,1} merupakan rute yang dilalu oleh kendaraan 1 yaitu **1-14-18-19-20-16-17-15-13-1**. Dengan total beban yang diangkut oleh kendaraan 3 adalah **190** satuan berat. Serta jarak tempuh **95.89** satuan jarak dengan total waktu tempuh adalah **815.89** satuan waktu.

Dari keseluruhan kendaraan memiliki total jarak tempuh **191.82** satuan jarak. Jika dibandingkan dengan solusi eksisting pada data set dengan total jarak **191.30** maka tingkat % error pada *sotfware* adalah **0.27%**.



Gambar 3.6 Plot rute terpilih pada C101.25

Pada Gambar 3.6 merupakan visualisasi rute yang dipilih oleh program ketika dijalankan pada data set C101.25. Menurut gambar tersebut, terdapat tiga jenis garis yang disimbolkan dengan berbagai warna yang mewakili setiap *vehicle* serta rutenya. Mulai dari *vehicle* pertama dengan warna merah, kedua dengan warna hijau, dan ketiga dengan warna biru.

4. KESIMPULAN/RINGKASAN

Hasil uji coba dan analisis yang telah dilakukan dapat diambil beberapa kesimpulan sebagai berikut:

1. *Algoritma gabungan Ant Colony Optimization* dan *Tabu Search* (ACOTS) dapat berjalan pada Vehicle Routing Problem with Time Windows (VRPTW). Dibuktikan dengan hasil validasi algoritma yang berjalan dengan lancar serta pengujian menggunakan 25 titik yang mendapatkan tingkat error 0.27% terhadap solusi eksisting
2. Algoritma ACOTS dapat memberikan solusi yang optimal pada permasalahan VRPTW pada data set solomon C101, C105, dan C106 dengan 25 titik.
3. Pembatasan bilangan random yang digunakan pada semut untuk menentukan tujuan selanjutnya mempengaruhi tingkat optimasi algoritma. Karena semakin terbatas semut tersebut memilih tujuan, semakin sedikit pula variasi rute yang diberikan oleh algorimta sehingga kemungkinan solusi tidak optimal semakin besar.
4. Subrute yang dihasilkan dapat mewakili sebagai jumlah kendaraan yang digunakan untuk melakukan perjalanan. Sehingga dapat menjadi acuan ketika ingin melakukan perhitungan lebih lanjut terkait biaya yang akan dikeluarkan.

DAFTAR PUSTAKA

[1] M. M. Abdulkader, Y. Gajpal dan T. Y. ElMekkawy, "Hybridized ant colony algorithm for the Multi Compartment Vehicle Routing Problem," ScienceDirect, 2015.

[2] Solomon, "VRPTW Solomon Benchmark," Sintef, 18 April 2008. [Online]. Available: https://www.sintef.no/projectweb/top/vrptw/solomon-benchmark/100-customers/. [Diakses 26 Februari 2016].

[3] D. D. Haynes dan S. M. Corns, "Agorithm for Tabu - Ant Colony Optimizer," IEEE, 2015.

[4] H. Katagiri, T. Hayashida, I. Nishizaki dan Q. Guo, "A hybrid algorithm based on tabu search and ant colony optimization fo k-minimum spanning tree problem," ScienceDirect, 2012.

[5] P. Agustina, Implementation of Algoritm Ant Colony System (ACS) for Open Vehicle Routing Problem (OVRP) to Determine Distribution Route of Newspaper (Case Study: Harian Surya), Surabaya: Institut Teknologi Sepuluh Nopember, 2008.

[6] O. Brasysy dan M. Grendeau, Genetic Algoritm for the Vehicle Routing Problem with Time Windows, Arpakannus, 2001.

[7] K. Iswardani, Penerapan Ant Colony Optimization Pada Vehicle Routing Problem Time Windows (Case Study: CV. Yufa Barokah), Surabaya: Institut Teknologi Sepuluh Nopember, 2015.

[8] P. Toth dan D. Vigo, The Vehicle Routing Problem, Philadelpia: SIAM, 2002.

[9] Liu, Qi dan Chen, "Optimization of Vehicle Routing Problem Based on Ant Colony System," dalam D. S. Huang, K. Li, and G. W. Irwin (Eds.): Computational Intelligence, 2006.

[10] B. Kallehauge, J. Larsen, O. B. G. Madsen dan M. M. Solomon, "Vehicle Routing with Time Windows," dalam Desaulniers G. et al., editor. Column Generation, New York, Springer, 2002.

[11] L. Joe, "Slideshare," 30 April 2014. [Online]. Available: http://www.slideshare.net/LindaJoe/energyaware-task-scheduling-using-antcolony-optimization-in-cloud. [Diakses 24 February 2016].

[12] J. E. Bell dan P. R. McMullen, "Ant colony optimization techniques for the vehicle routing problem," Advanced Engineering Informatics, pp. 41-48, 2004.

[13] Priyandari, "Tabu Search – Introduction," Universitas Sebelas Maret, 9 September 2009. [Online]. Available: http://priyandari.staff.uns.ac.id/200909/tabu-search-introduction/. [Diakses 18 Februari 2016].

[14] Hindriyanto, "Metaheuristics : Tabu Search," duniaku, 3 September 2012. [Online]. Available: https://hindriyanto.wordpress.com/2012/09/03/metaheuristics-tabu-search/. [Diakses 18 Februari 2016].

PENGEMBANGAN METODE *ANT COLONY OPTIMIZATION* DAN *TABU SEARCH* UNTUK MENYELESAIKAN *VEHICLE ROUTING PROBLEM WITH TIME WINDOWS*

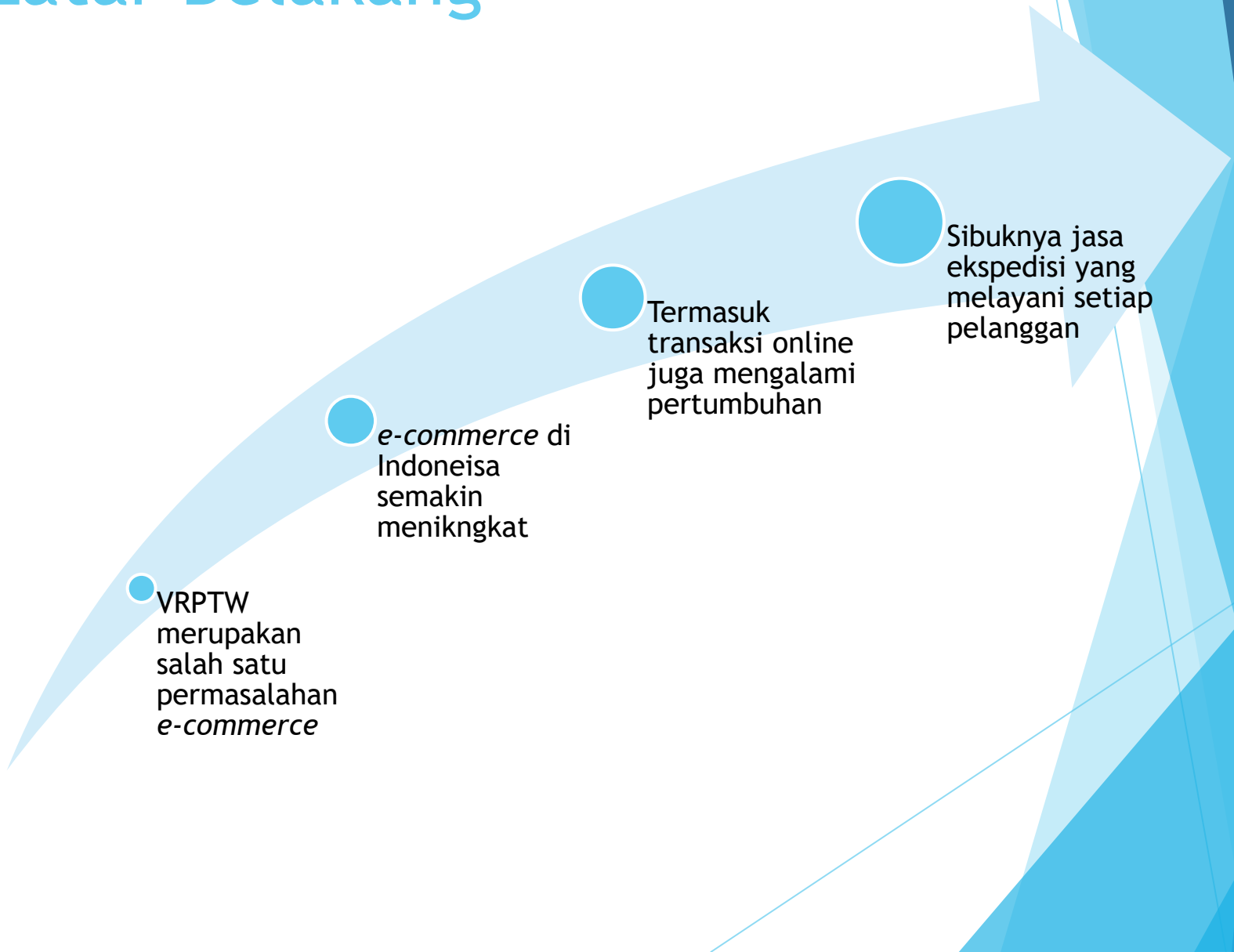
Wildan Habiby Fanani
5212100056

Dosen Pembimbing:
Edwin Riksakomara, S.Kom., M.T.
Amalia Utamima, S.Kom., M.B.A.

Rekayasa Data dan Intelegensi Bisnis
JURUSAN SISTEM INFORMASI
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

Pendahuluan

Latar Belakang



Rumusan Masalah

- Bagaimana mengaplikasikan algoritma gabungan *Ant Colony Optimization* dan *Tabu Search* (ACOTS) pada *Vehicle Routing Problem with Time Windows* (VRPTW)?
- Bagaimana mendapatkan solusi yang optimal pada VRPTW menggunakan ACOTS?

Batasan Masalah

- Penelitian ini menggunakan data set *solomon problem* dari website Sintef : *Research, Technology and Innovation*
- Jumlah kendaraan, kapasitas kendaraan, serta waktu *loading* dan *unloading* barang disesuaikan dengan data set yang ada.

Tujuan

- Mengembangkan algoritma gabungan *Ant Colony Optimization* dan *Tabu Search* (ACOTS) pada *Vehicle Routing Problem with Time Windows* (VRPTW).
- Mendapatkan solusi yang optimal pada VRPTW menggunakan ACOTS.

Manfaat

- Mendapatkan rute tercepat sehingga dapat menimimalkan biaya yang dikeluarkan.
- Mampu mengetahui jumlah kendaraan serta waktu yang dibutuhkan.

Tinjauan Pustaka

Studi Sebelumnya

No	Judul	Peneliti	Hasil
1	Algorithm for a Tabu – Ant Colony Optimizer	David D. Haynes and Steven M. Corns	Algoritma Tabu-ACO dapat berjalan dengan baik pada kasus Steiner Tree Problem
2	A hybrid algorithm based on tabu search and ant colony optimization for k-minimum spanning tree problems	Hideki Katagiri, Tomohiro Hayashida, Ichiro Nishizaki, Qingqiang Guo	Metode gabungan Tabu-ACO memiliki nilai yang lebih optimal dari pada ACO itu sendiri pada kasus k-Minimum Spanning Tree
3	Hybridized ant colony algorithm for the Multi Compartment Vehicle Routing Problem	Mohamed M.S. Abdulkader, Yuvraj Gajpal, Tarek Y. ElMekkawy	Ant Colony dikombinasi dengan Local Search memiliki hasil yang baik serta waktu yang cepat dalam kasus Multi Compartment Vehicle Routing Problem

Perbandingan Penelitian

No	Peneliti	Metode		Studi Kasus	
		ACO	TS	VRPTW	Lainnya
1	David D. Haynes and Steven M. Corns	v	v		STP
2	Hideki Katagiri et. al.	v	v		k-MST
3	Mohamed M.S. et. al.	v	v		MCVRP
[redacted]	Wildan H. Fanani	v	v	v	

Vehicle Routing Problem with Time Windows (VRPTW)

VRPTW adalah permasalahan pencarian rute dalam menentukan sistem distribusi

Rute yang optimal adalah rute yang memiliki total jarak dan waktu perjalanan terpendek dalam memenuhi *demand customer*

Rumus Matematis VRPTW

Fungsi Tujuan, meminimalkan:

$$z = \sum_{k \in V} \sum_{i \in N} \sum_{k \in N} c_{ij} x_{ijk}$$

$$x_{ijk} = \begin{Bmatrix} 1, \text{jika kendaraan } k \text{ melalui } i \text{ lalu ke } j \\ 0, \text{selainnya} \end{Bmatrix}$$

Dimana:

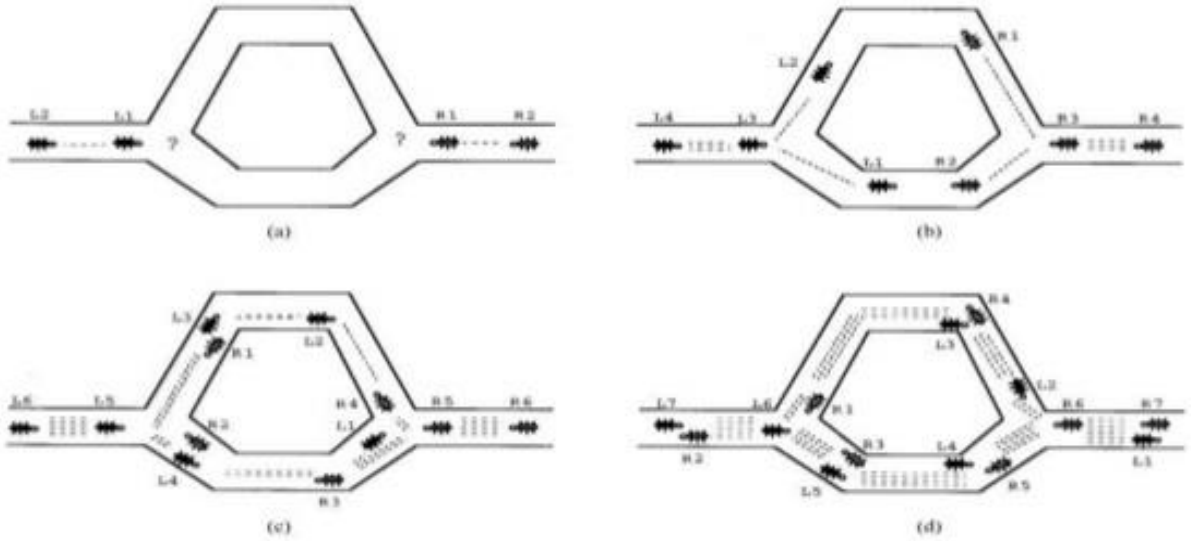
- V = himpunan kendaraan berkapasitas sama
- C = himpunan pelanggan = {1,…,n}
- A = himpunan sisi berarah = $\{(i,j)|i,j \in N, i \neq j\}$
- c_{ij} = jarak/biaya dari simpul i ke simpul j.
- t_{ij} = waktu tempuh dari simpul i ke simpul j.
- d_i = jumlah permintaan pelanggan i.
- q = kapasitas kendaraan
- $[a_i, b_i]$ = time windows dari simpul i.

Rumus Matematis VRPTW (2)

Dengan Batasan:

- Setiap pelanggan dikunjungi tepat satu kali oleh suatu kendaraan
- Total permintaan semua pelanggan dalam satu rute tidak melebihi kapasitas kendaraan
- Setiap rute awal dari depot 0
- Setiap kendaraan yang mengunjungi suatu pelanggan pasti akan meninggalkan pelanggan tersebut
- Setiap rute berakhir di depot n+1
- Waktu pelayanan disetiap pelanggan memenuhi *time windows*

Ant Colony Optimization



- Ant Colony Optimization adalah meniru perilaku sekelompok semut untuk menemukan solusi terbaik dari permasalahan optimasi kombinatorial ketika sekelompok semut mencari makanan.

Ant Colony Optimization (2)

- **Aturan transisi status**: Aturan yang digunakan oleh semut untuk memutuskan kemana dia akan pergi
- **Aturan *local updating***: Aturan yang digunakan untuk menandai lintasan yang baru dilalui dengan meng-*update* jumlah *pheromone* yang ada pada sebuah lintasan
- **Aturan *global updating***: Aturan yang digunakan setelah semua semu membentuk jalur perjalanan

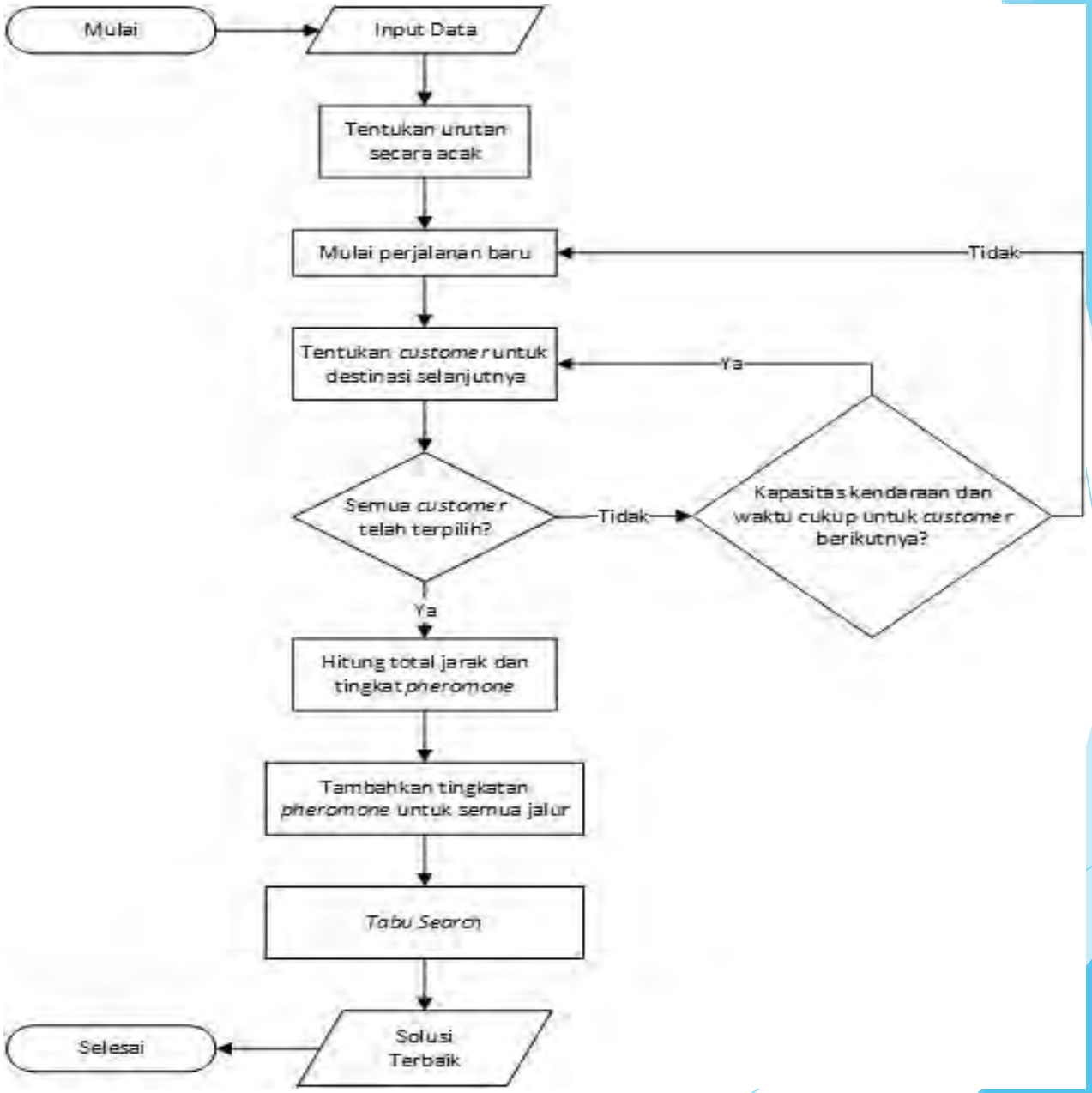
Tabu Search

Tabu Search merupakan salah satu prosedur metaheuristik untuk menyelesaikan permasalahan optimisasi kombinatorial

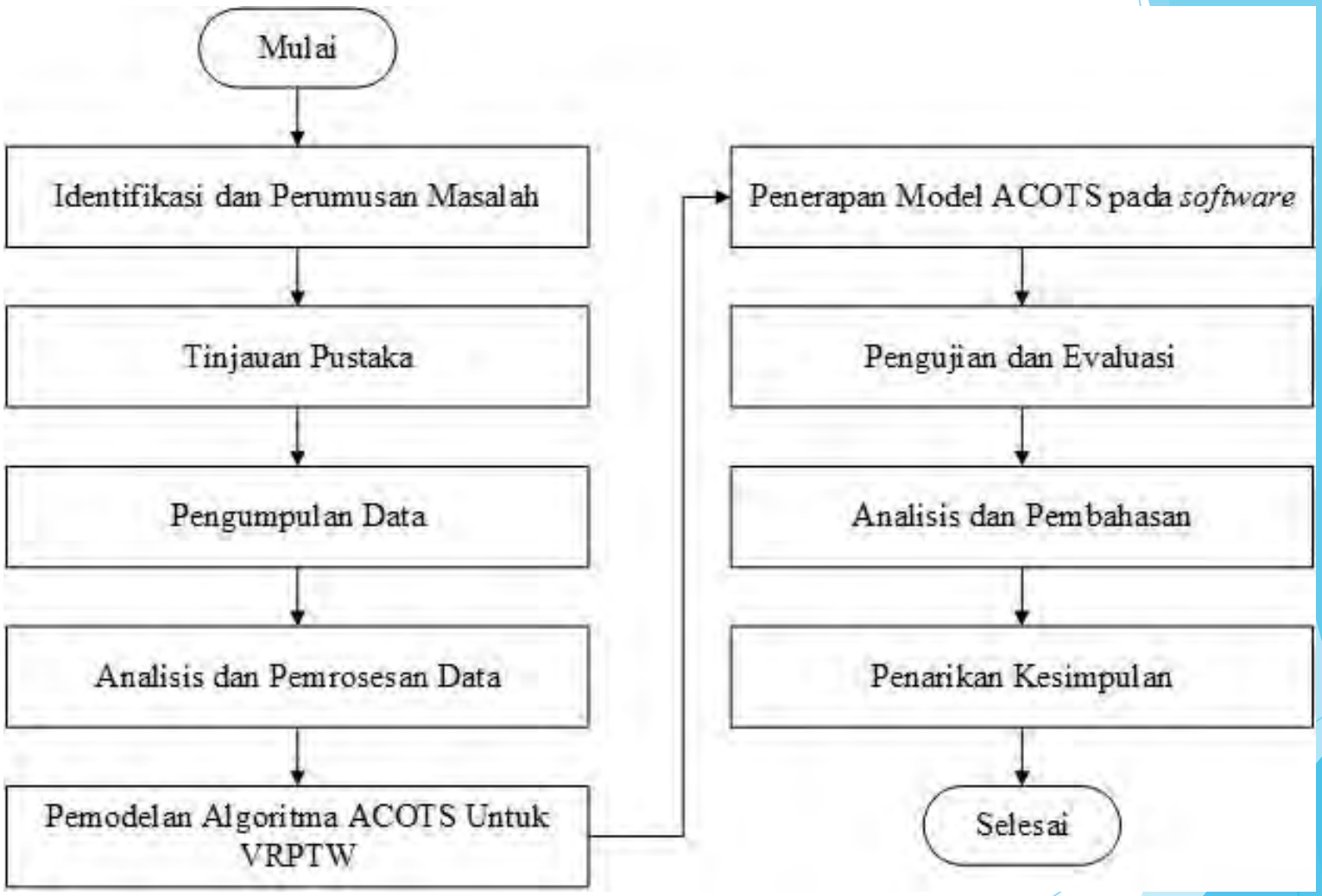
Tabu Search berprinsip pada penggunaan memori sebagai elemen esensial dalam pencariannya

Tabu Search (2)

- ***Tabu list*:** untuk menghindari mengevaluasi kandidat solusi yang pernah dikunjungi sebelumnya
- ***Aspiration criteria*:** jika pergerakan tabu tersebut menghasilkan kandidat solusi yang memiliki nilai yang lebih baik daripada solusi terbaik yang telah dihasilkan
- **Intensifikasi:** untuk memberikan prioritas kepada atribut dari solusi berkualitas tersebut
- **Diversifikasi:** dapat mengeksplorasi area dalam ruang pencarian yang belum dikunjungi


Mulai
Input Data
Tentukan urutan secara acak
Mulai perjalanan baru
Tidak
Tentukan *customer* untuk destinasi selanjutnya
Ya
Semua *customer* telah terpilih?
Tidak
Kapasitas kendaraan dan waktu cukup untuk *customer* berikutnya?
Ya
Hitung total jarak dan tingkat *pheromone*
Tambahkan tingkatan *pheromone* untuk semua jalur
Tabu Search
Selesai
Solusi Terbaik

Metodologi


Mulai
Identifikasi dan Perumusan Masalah
Tinjauan Pustaka
Pengumpulan Data
Analisis dan Pemrosesan Data
Pemodelan Algoritma ACOTS Untuk VRPTW
Penerapan Model ACOTS pada *software*
Pengujian dan Evaluasi
Analisis dan Pembahasan
Penarikan Kesimpulan
Selesai

Implementasi

Pengolahan Data

- Data Set diambil oleh peneliti dari website SINTEF yang merupakan salah satu organisasi riset terbesar di Scanidavia.

CUST NO.	XCOORD.	YCOORD.	DEMAND	READY TIME	DUE DATE	SERVICE TIME
1	40.00	50.00	0.00	0.00	1236.00	0.00
2	45.00	68.00	10.00	912.00	967.00	90.00
3	45.00	70.00	30.00	825.00	870.00	90.00
4	42.00	66.00	10.00	65.00	146.00	90.00
5	42.00	68.00	10.00	727.00	782.00	90.00
6	42.00	65.00	10.00	15.00	67.00	90.00

Jarak Antar Titik

- $\sqrt{(X_a - X_b)^2 + (Y_a - Y_b)^2}$

	A	B	C	D	E	F	G	H	I	J
1	0,00	18,68	20,62	16,12	18,11	15,13	19,00	16,00	18,11	20,10
2	18,68	0,00	2,00	3,61	3,00	4,24	5,10	5,39	7,00	7,28
3	20,62	2,00	0,00	5,00	3,61	5,83	5,10	6,40	7,28	7,00
4	16,12	3,61	5,00	0,00	2,00	1,00	3,61	2,00	4,47	5,66
5	18,11	3,00	3,61	2,00	0,00	3,00	2,24	2,83	4,00	4,47
6	15,13	4,24	5,83	1,00	3,00	0,00	4,47	2,24	5,00	6,40
7	19,00	5,10	5,10	3,61	2,24	4,47	0,00	3,00	2,24	2,24
8	16,00	5,39	6,40	2,00	2,83	2,24	3,00	0,00	2,83	4,47
9	18,11	7,00	7,28	4,47	4,00	5,00	2,24	2,83	0,00	2,00
10	20,10	7,28	7,00	5,66	4,47	6,40	2,24	4,47	2,00	0,00

Uji Validasi Program dengan Perhitungan Manual

- Data yang digunakan untuk pengujian program dengan perhitungan manual adalah 5 titik + 1 depot.

Jarak antar titik

	1	2	3	4	5	6
1	0,00	18,68	20,62	16,12	18,11	15,13
2	18,68	0,00	2,00	3,61	3,00	4,24
3	20,62	2,00	0,00	5,00	3,61	5,83
4	16,12	3,61	5,00	0,00	2,00	1,00
5	18,11	3,00	3,61	2,00	0,00	3,00
6	15,13	4,24	5,83	1,00	3,00	0,00

Batasan setiap titik

Cust. No	Demand	Ready Time	Due Date	Service Time
1	0	0	1236	0
2	10	912	967	90
3	30	825	870	90
4	10	65	146	90
5	10	727	782	90
6	10	15	67	90

Penyelesaian Perhitungan Manual

- ***Langkah 1:***
- Rute diawali dari depot (Cust. 1), kemudian dihitung waktu sampai ke masing-masing konsumen.

Pos	Time Start	Next Cust.	Demand	Waktu Sampai	Waktu Pergi	Time Win.	Cap.
Depot	0	2	10	18,68	108,68	tidak	ya
		3	30	20,62	110,62	tidak	ya
		4	10	16,12	106,12	tidak	ya
		5	10	18,11	108,11	tidak	ya
		6	10	15,13	105,13	ya	ya

Pada langkah 1, konsumen yang dipilih adalah konsumen 6, sehingga rute sementara adalah 1-6. Dari rute sementara tersebut didapatkan *demand* sementara yang diangkut adalah **10** dengan total jarak **15,13** serta waktu **15,13**.

- *Langkah 2:*
- Pada langkah sebelumnya telah terpilih konsumen 6.

Pos	Time Start	Next Cust.	Demand	Waktu Sampai	Waktu Pergi	Time Win.	Cap.
6	105,1	2	20	109,37	199,37	tidak	ya
		3	40	110,96	200,96	tidak	ya
		4	20	106,13	196,13	ya	ya
		5	20	108,13	198,13	tidak	ya

Pada tahapan ini yang terpilih adalah konsumen 4, sehingga rute sementara kali ini adalah 1-6-4 dengan *demand* yang diangkut adalah **20** dengan total jarak **16,13** serta waktu yang dibutuhkan adalah **106,13**.

- ***Langkah 3:***
- Melakukan pengecekan terhadap kapasitas serta waktu yang dibutuhkan untuk melakukan tahap selanjutnya.

Pos	Time Start	Next Cust.	Demand	Waktu Sampai	Waktu Pergi	Time Win.	Cap.
4	196,1	2	30	199,74	289,74	tidak	ya
		3	50	201,13	291,13	tidak	ya
		5	30	198,13	288,13	tidak	ya

- ***Langkah 3(2):***
- Pada tabel di atas konstrain waktu tidak dapat terpenuhi sehingga perlu dilakukan proses lebih lanjut untuk mencari jarak minimal serta selisih *time windows*.

Cust.	Jarak	Waktu Sampai	Ready	End	Selisih
2	3,61	199,74	912	967	712,26
3	5,00	201,13	825	870	623,87
5	2,00	198,13	727	782	528,87

- Melakukan tahapan tersebut hingga didapatkan rute optimal adalah 1-6-4-5-3-2-1 dengan jarak **42,52** satuan jarak serta waktu tempuh **1019,29.**

Perhitungan Program

- ***Langkah 1:***
- Berikut adalah parameter yang digunakan untuk inisialisasi awal algoritma ACOTS:
- α = 1 (parameter pengendali intensitas *pheromone*)
- β = 3 (parameter pengendali visibilitas)
- e = 0.5 (parameter penguapan *pheromone*)
- m = 25 (jumlah maksimal semut)
- iter = 100 (jumlah maksimal iterasi)
- Cap = 200 (jumlah kapasitas maksimal *m*)
- q0 = 0.96 (koefisien cost elemination)

Tingkat Pheromone Awal

```
t =

    0.0100    0.0100    0.0100    0.0100    0.0100    0.0100
    0.0100    0.0100    0.0100    0.0100    0.0100    0.0100
    0.0100    0.0100    0.0100    0.0100    0.0100    0.0100
    0.0100    0.0100    0.0100    0.0100    0.0100    0.0100
    0.0100    0.0100    0.0100    0.0100    0.0100    0.0100
    0.0100    0.0100    0.0100    0.0100    0.0100    0.0100
```

Visibilitas jarak ($\eta_{ij} = 1/d_{ij}$)

```
h =

         0    0.0526    0.0476    0.0625    0.0556    0.0667
    0.0526         0    0.5000    0.2500    0.3333    0.2500
    0.0476    0.5000         0    0.2000    0.2500    0.1667
    0.0625    0.2500    0.2000         0    0.5000    1.0000
    0.0556    0.3333    0.2500    0.5000         0    0.3333
    0.0667    0.2500    0.1667    1.0000    0.3333         0
```

- ***Langkah 2:***
- Jalankan fungsi *program* pada *command window* Matlab dengan memberikan parameter jumlah semut, iterasi, dan kapasitas.

```
>> acotsvrptw(25,100,200)

bestsol =

    43


ans =

    [1x7 double]
```

Variabel **bestsol** yang menunjukkan jarak minimal yang ditempuh oleh semut yaitu 43 satuan jarak.
Sedangkan variabel **ans** menunjukkan bahwa hanya membutuhkan 1 semut.

ans{1,1} <1x7 double>

	1	2	3	4	5	6	7
1	1	6	4	5	3	2	1

Pengujian 25 Titik Data Set

- Parameter Input Model

Parameter Input	Nilai
Jumlah Semut (m)	25
Jumlah titik (n)	26
Kapasitas kendaraan (Capacity)	200
Pheromone awal (t)	0.01
Evaporasi (e)	0.5
Qo	0.96
α (alpha)	1
ß (beta)	3
Jumlah iterasi	100
Demand	Terdapat di Lampiran 1
Time Windows	Terdapat di Lampiran 1
Service Time	Terdapat di Lampiran 1
Jarak	Terdapat di Lampiran 1

Proses

- Penentuan terbaik setiap iterasi

```
rute_minimal =

  297.9848


bestrute =

  Columns 1 through 4

    [1x5 double]    [1x10 double]    [1x13 double]    [1x5 double]

  Columns 5 through 10

     []      []      []      []      []      []


tabuit =

  329.8363


newglobsol =

    [1x8 double]    [1x10 double]    [1x13 double]
```

Rute minimal
Pada iterasi 1

Kombinasi rute
Dengan jarak minimal
Pada iterasi 1

Jarak yang terbentuk
Setelah proses tabu

Hasil tabu search
Pada iterasi 1

- Penemuan rute terbaik

```
rute_minimal =

  252.8649


bestrute =

    [1x13 double]    [1x10 double]    [1x8 double]      []      []


tabuit =

  191.8136


newglobsol =

    [1x13 double]    [1x8 double]    [1x10 double]
```

Rute minimal
Pada iterasi x

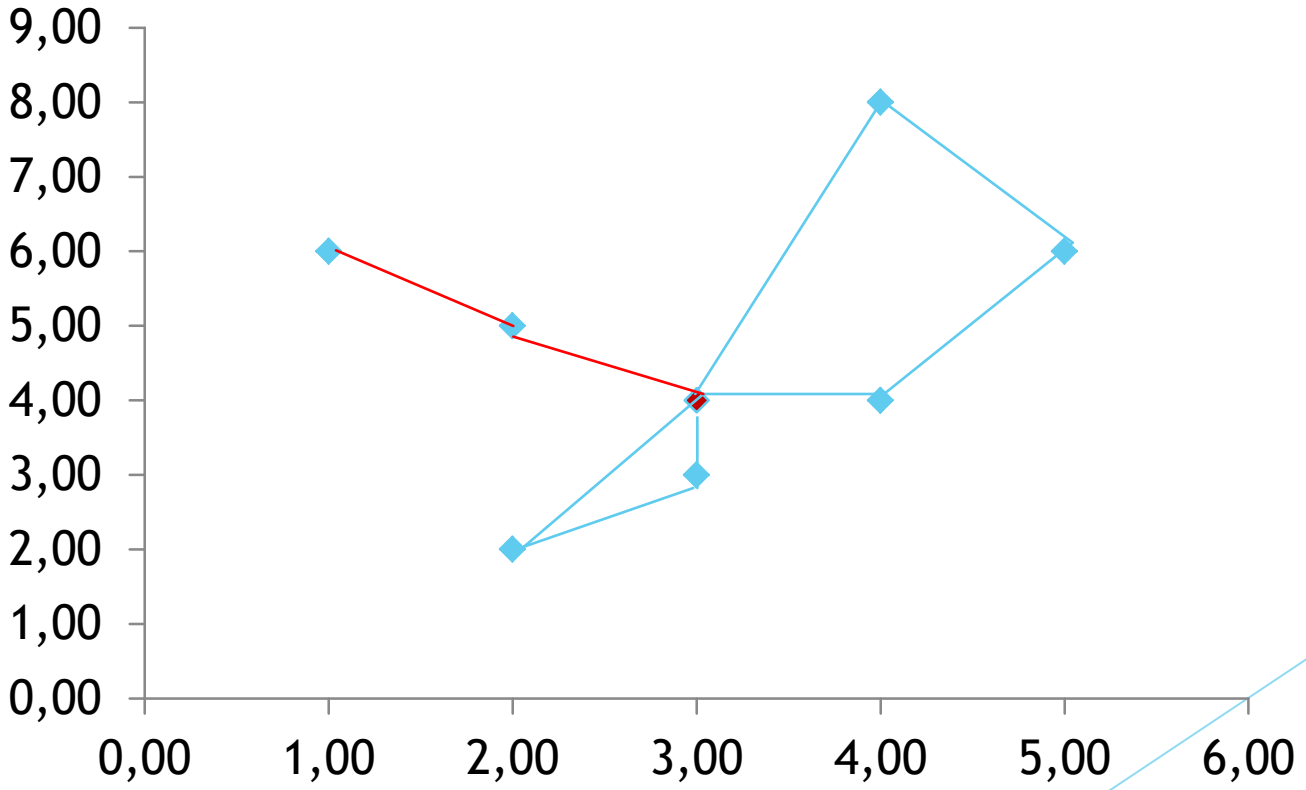
Kombinasi rute
Dengan jarak minimal
Pada iterasi x

Jarak yang terbentuk
Setelah proses tabu

Hasil tabu search
Pada iterasi 1

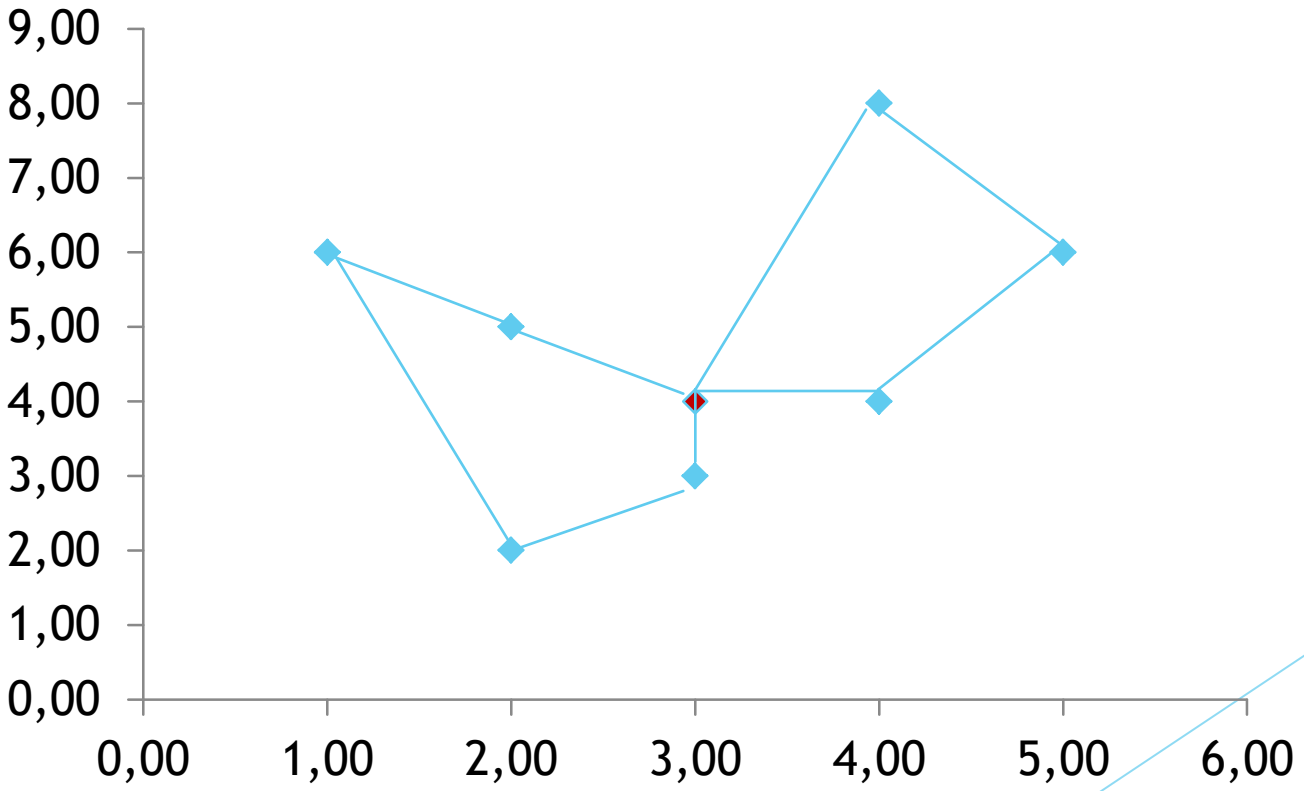
Ilustrasi

Ant Colony Optimization yang berjalan dengan batasan waktu serta kapasitas. Terdapat pembatasan pola yang dijalankan oleh semut sehingga terdapat 3 subrute yang dibuat oleh ACO.



Ilustrasi(2)

Tabu search digunakan untuk menkombinasikan rute yang memungkinkan untuk digabung sehingga dapat meminimalkan jumlah subrute. Kemudian dibandingkan apakah dengan kombinasi tersebut memiliki jarak tempuh yang lebih baik atau tidak.



- Hasil Pengujian 25 Titik

```
bestsol =

   191.8136


ans =

    [1x13 double]
    [1x8  double]
    [1x10 double]
```

Problem	NV	Distance	Authors
C101.25	3	191.3	KDMSS

Dari keseluruhan kendaraan memiliki total jarak tempuh **191.81**. Jika dibandingkan dengan solusi eksisting pada data set dengan total jarak **191.30** maka tingkat error pada *sotfware* adalah **0.36%**.

Hasil dan Pembahasan

Analisis Hasil Validasi Algoritma

- Solusi yang dihasilkan baik secara manual maupun model *software* yaitu dengan rute **1-6-4-5-3-2-1** serta dengan jarak **42,52** jika dibulatkan menjadi **43** satuan jarak.
- Dari uji validitas tersebut dapat disimpulkan bahwa model *software* yang dibuat sudah valid dan dapat digunakan untuk melakukan pengujian data set dalam skala yang lebih besar.

Analisis Hasil Pengolahan Data Set

- Parameter input yang digunakan untuk melakukan pengolahan data yaitu α=1, ß=3, q0=0.96, dan e=0.5.
- Data lain yang digunakan yaitu koordinat masing-masing titik yang diolah menjadi satu titik tertentu sehingga dapat diketahui jarak antar titik baik depot-konsumen maupun konsumen-konsumen.
- Selain itu data *demand*, *time windows*, dan *service time* sesuai dengan data set yang ada.
- Output yang dihasilkan adalah total jarak serta rute maupun subrute yang terbaik.

Pengujian 25 Titik

- Pengujian yang dilakukan pada data set C101 mendapatkan hasil: total jarak tempuh **191.82** satuan jarak, serta menggunakan **3** *vehicles* untuk menempuh jarak tersebut.
- Jika dibandingkan dengan solusi eksisting pada data set dengan total jarak **191.30** maka tingkat error pada *sotfware* adalah **0.27%**.

Kesimpulan dan Saran

Kesimpulan

- Algoritma gabungan *Ant Colony Optimization* dan *Tabu Search* (ACOTS) dapat berjalan pada *Vehicle Routing Problem with Time Windows* (VRPTW). Dibuktikan dengan hasil validasi algoritma yang berjalan dengan lancar serta pengujian menggunakan 25 titik yang mendapatkan tingkat error **0.27%** terhadap solusi eksisting.
- Algoritma ACOTS dapat memberikan solusi yang optimal pada permasalahan VRPTW pada data set solomon C101, C105, dan C106 dengan 25 titik.

Percobaan Lain

Problem	NV	Distance	NV Prog.	Dist. Prog.	Error
C101.25	3	191,3	3	191,82	0,272
C105.25	3	191,3	3	191,82	0,272
C106.25	3	191,3	3	191,82	0,272

Saran

- Pengolahan data set untuk perhitungan jarak antar dua titik masih menggunakan Ms. Excel dengan perhitungan manual. Sehingga dikhawatirkan terdapat kesalahan perhitungan yang menyebabkan kesalahan pula pada model *software*.
- Dalam penelitian ini menggunakan *software* MATLAB dengan semua script ada pada satu file serta belum menerapkan OOP. Sehingga ketika terdapat kesalahan terdapat pada salah satu script akan mengalami kesulitan saat melakukan perbaikan.
- *Ant Colony Optimization* dan *Tabu Search* dapat digabungkan dengan algoritma lain yang dapat meningkatkan kualitas hasil solusi serta mampu menampung lebih dari 25 titik.

Daftar Pustaka

[1]	**M. M. Abdulkader, Y. Gajpal dan T. Y. ElMekkawy, "Hybridized ant colony algorithm for the Multi Compartment Vehicle Routing Problem," ScienceDirect, 2015.**
[2]	Solomon, "VRPTW Solomon Benchmark," Sintef, 18 April 2008. [Online]. Available: https://www.sintef.no/projectweb/top/vrptw/solomon-benchmark/100-customers/. [Diakses 26 Februari 2016].
[3]	D. D. Haynes dan S. M. Corns, "Agorithm for Tabu - Ant Colony Optimizer," IEEE, 2015.
[4]	H. Katagiri, T. Hayashida, I. Nishizaki dan Q. Guo, "A hybrid algorithm based on tabu search and ant colony optimization fo k-minimum spanning tree problem," ScienceDirect, 2012.
[5]	P. Agustina, Implementation of Algoritm Ant Colony System (ACS) for Open Vehicle Routing Problem (OVRP) to Determine Distribution Route of Newspaper (Case Study: Harian Surya), Surabaya: Institut Teknologi Sepuluh Nopember, 2008.
[6]	O. Brasysy dan M. Grendeau, Genetic Algoritm for the Vehicle Routing Problem with Time Windows, Arpakannus, 2001.
[7]	K. Iswardani, Penerapan Ant Colony Optimization Pada Vehicle Routing Problem Time Windows (Case Study: CV. Yufa Barokah), Surabaya: Institut Teknologi Sepuluh Nopember, 2015.
[8]	P. Toth dan D. Vigo, The Vehicle Routing Problem, Philadelpia: SIAM, 2002.
[9]	Liu, Qi dan Chen, "Optimization of Vehicle Routing Problem Based on Ant Colony System," dalam D. S. Huang, K. Li, and G. W. Irwin (Eds.): Computational Intelligence, 2006.
[10]	B. Kallehauge, J. Larsen, O. B. G. Madsen dan M. M. Solomon, "Vehicle Routing with Time Windows," dalam Desaulniers G. et al., editor. Column Generation, New York, Springer, 2002.
[11]	L. Joe, "Slideshare," 30 April 2014. [Online]. Available: http://www.slideshare.net/LindaJoe/energyaware-task-scheduling-using-antcolony-optimization-in-cloud. [Diakses 24 February 2016].
[12]	J. E. Bell dan P. R. McMullen, "Ant colony optimization techniques for the vehicle routing problem," Advanced Engineering Informatics, pp. 41-48, 2004.
[13]	Priyandari, "Tabu Search – Introduction," Universitas Sebelas Maret, 9 September 2009. [Online]. Available: http://priyandari.staff.uns.ac.id/200909/tabu-search-introduction/. [Diakses 18 Februari 2016].
[14]	Hindriyanto, "Metaheuristics : Tabu Search," duniaku, 3 September 2012. [Online]. Available: https://hindriyanto.wordpress.com/2012/09/03/metaheuristics-tabu-search/. [Diakses 18 Februari 2016].

Terima Kasih