ALGORITMA *IMPROVED ANT COLONY OPTIMIZATION* UNTUK PENYELESAIAN *VEHICLE ROUTING PROBLEM*

SKRIPSI

Oleh**:**
NIZARKASYI FIGHAR PRASETYA
145090407111025

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG
2018

ALGORITMA *IMPROVED ANT COLONY OPTIMIZATION* UNTUK PENYELESAIAN *VEHICLE ROUTING PROBLEM*

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana Matematika

Oleh:
NIZARKASYI FIGHAR PRASETYA
145090407111025

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG
2018

LEMBAR PENGESAHAN SKRIPSI

ALGORITMA *IMPROVED ANT COLONY OPTIMIZATION*
UNTUK PENYELESAIAN *VEHICLE ROUTING PROBLEM*

oleh:
NIZARKASYI FIGHAR PRASETYA
145090407111025

Setelah dipertahankan di depan Majelis Penguji
pada tanggal 17 Juli 2018
dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana Matematika

Pembimbing

Syaiful Anam, S.Si, M.T., Ph. D.
NIP. 197801152002121003

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Ratno Bagus Edy Wibowo, S.Si.,M.Si, Ph.D.
NIP. 197509082000031003

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama	**:**	**Nizarkasyi Fighar Prasetya**
NIM	**:**	**145090407111025**
Penulis Skripsi berjudul	**:**	**Algoritma *Improved Ant Colony Optimization* untuk penyelesaian *Vehicle Routing Problem***

dengan ini menyatakan bahwa:

1. **isi Skripsi yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termasuk di isi dan tertulis di Daftar Pustaka dalam Skripsi ini.**
2. **Apabila di kemudian hari ternyata Skripsi yang saya tulis terbukti hasil jiplakan, maka saya bersedia menanggung segala risiko yang akan saya terima. Demikian pernyataan ini dibuat dengan penuh kesadaran.**

Malang, 19 Juli 2018
yang menyatakan,

Nizarkasyi Fighar Prasetya
NIM 145090407111025

ALGORITMA *IMPROVED ANT COLONY OPTIMIZATION* UNTUK PENYELESAIAN *VEHICLE ROUTING PROBLEM*

ABSTRAK

Skripsi ini membahas perbaikan pada algoritma *Ant Colony Optimization* (ACO) dengan menambahkan metode operasi mutasi dan proses *local search* yang disebut *Improved Ant colony optimization* (IACO) untuk menyelesaikan *Vehicle Routing Problem* (VRP). VRP merupakan masalah penentuan rute dengan batasan berupa sejumlah kendaraan dengan kapasitas yang homogen. ACO dipilih sebagai metode untuk menyelesaikan VRP karena representasi solusi yang digunakan tepat untuk menyatakan rute. Umumnya, solusi yang diperoleh melalui proses konstruksi rute ACO bergantung pada nilai intensitas *pheromone* dan perbandingan jarak masing-masing titik tujuan, sehingga diperlukan metode optimasi tambahan agar solusi yang dihasilkan tidak cenderung menuju optimum lokal. Proses operasi mutasi dilakukan dengan menggunakan konsep *Reciprocal Exchange Mutation* pada Algoritma Genetika. ACO dan IACO diterapkan pada program MATLAB dengan menggunakan data uji yang ada, kemudian dilakukan perbandingan hasil baik dari segi jarak yang dihasilkan ataupun waktu komputasi. Hasil rute dengan jarak terpendek yang diperoleh menunjukkan IACO mampu bersaing dengan ACO biasa.

Kata kunci: *algoritma ant colony optimization, operasi mutasi, local search, vehicle routing problem*

IMPROVED ANT COLONY OPTIMIZATION ALGORITHM TO SOLVE VEHICLE ROUTING PROBLEM

ABSTRACT

This thesis presents an Ant Colony Optimization (ACO) algorithm with mutation and local search method called Improved Ant Colony Optimization (IACO) to solve Vehicle Routing Problem (VRP). VRP is mainly a problem to determine a route with vehicle capacity so that the total distance will be minimized. ACO is chosen to solve VRP because its capability to use route representation. Commonly, Solution of ACO's route construction is depend on pheromone intensity value and the distance's comparison of each costumer. These circumstances need an additional optimization method, so the solution will never be a local optimum. Mutation operation uses Reciprocal Exchange Mutation from Genetic Algorithm. ACO and IACO are implemented in MATLAB program using existing benchmark instances, then the minimum route results or computation times are compared. Minimum route results show that IACO gives competitive results compared to the original ACO.

Kata kunci: *ant colony optimization algorithm, mutation operation, local search, vehicle routing problem*

KATA PENGANTAR

Puji syukur ke hadirat Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan skripsi yang berjudul **ALGORITMA *IMPROVED ANT COLONY OPTIMIZATION* UNTUK PENYELESAIAN *VEHICLE ROUTING PROBLEM*** dengan baik dan lancar. Shalawat serta salam selalu tercurahkan kepada Nabi Muhammad SAW sebagai suri teladan bagi penulis.

Skripsi ini tidak dapat diselesaikan dengan baik tanpa bantuan, bimbingan, dan motivasi dari berbagai pihak. Oleh karena itu, penulis menyampaikan terimakasih kepada:

1. Syaiful Anam, S.Si., M.T., Ph. D. selaku dosen pembimbing atas segala bimbingan, motivasi, bantuan, saran, dan kesabaran yang telah diberikan selama penulisan skripsi ini.
2. Drs. Marsudi, MS, dan Nur Shofianah, S.Si.,M.Si.,Ph.D selaku dosen penguji atas segala kritik dan saran yang diberikan untuk perbaikan skripsi ini.
3. Dra. Ari Andari, M.Si. selaku dosen pembimbing yang selalu memberikan motivasi dan saran kepada penulis.
4. Dr. Isnani Darti, S.Si., M.Si selaku Ketua Program Studi Matematika yang selalu memberikan motivasi kepada penulis untuk menyelesaikan skripsi ini tepat waktu.
5. Ratno Bagus Edy Wibowo, S.Si., M.Si., Ph. D. selaku Ketua Jurusan Matematika atas segala bantuan yang diberikan.
6. Segenap dosen dan staf Jurusan Matematika atas ilmu dan pengalaman yang sangat bermanfaat bagi penulis.
7. Ayah (Heru), Ibu (Wati), Adik-adik (Yannu, Nazwa), serta seluruh keluarga besar atas segala doa, bantuan, dukungan, dan semangat yang tiada henti diberikan selama ini kepada penulis.
8. Keluarga Besar Matematika 2014 atas segala bantuan dan motivasinya.
9. Semua pihak yang tidak dapat penulis sebutkan satu persatu.

Semoga Allah SWT senantiasa memberikan anugerah dan barokah-Nya kepada semua pihak yang telah membantu menyelesaikan skripsi ini. Penulis menyadari bahwa dalam penulisan skripsi ini masih terdapat kekurangan. Oleh karena itu, penulis mengharapkan kritik dan saran yang bersifat membangun. Kritik dan saran dapat disampaikan melalui email ke alamat

pnfighar@gmail.com. Semoga skripsi ini dapat bermanfaat bagi berbagai pihak, serta menjadi inspirasi bagi penulis skripsi selanjutnya.

Malang, 19 Juli 2018

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

DAFTAR LAMPIRAN

Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang

Vehicle Routing Problem (VRP) adalah permasalahan optimasi penentuan jalur distribusi terpendek bagi sekelompok kendaraan yang ingin mengunjungi beberapa kota (Bin, dkk., 2009). Contoh dari penerapan VRP adalah pendistribusian barang yang akan dikirim ke alamat penerima oleh penyedia jasa layanan pengiriman barang seperti JNE dan J&T. Target pencarian solusi VRP yang diaplikasikan pada distribusi logistik adalah menentukan jalur distribusi dari sebuah gudang menuju lokasi pelanggan dan kembali ke gudang tanpa melebihi kapasitas daya angkut masing-masing kendaraan dengan ongkos minimum. Kombinasi lokasi pelanggan pada VRP tidak membatasi cara pemilihan jalur distribusi logistik, sehingga VRP dapat dianggap sebagai optimasi masalah kombinatorik dimana jumlah dari solusi yang mungkin untuk masalah tersebut meningkat secara eksponen dengan peningkatan jumlah pelanggan (Bell dan McMullen, 2004).

Salah satu cara memperoleh solusi VRP adalah menggunakan algoritma heuristik. Selain menggunakan algoritma heuristik, VRP juga dapat diselesaikan menggunakan metode eksak, namun untuk masalah optimasi yang kompleks akan memerlukan waktu penyelesaian yang lebih lama dibanding algoritma heuristik. Algoritma heuristik adalah algoritma yang digunakan untuk mencari solusi melalui semua kemungkinan yang ada, tetapi dalam pencariannya tidak bisa dijamin akan ditemukan solusi terbaik, sehingga algoritma ini juga disebut dengan metode perkiraan (Christofides, dkk., 1979; Foulds, 1984). Contoh algoritma heuristik yaitu *Simulated Annealing* (SA), *Genetic Algorithms* (GA), *Tabu Search* (TS), dan *Ant Colony Optimization* (ACO) (Bin, dkk., 2009). Diantara algoritma heuristik tersebut, ACO adalah algoritma optimasi solusi VRP dengan pola konstruksi yang menyerupai VRP itu sendiri (Bullnheimer, dkk., 1997).

Penerapan ACO pada VRP memisalkan gudang pusat distribusi (depot) sebagai sarang dan pelanggan sebagai makanan pada kebiasaan koloni semut dalam mencari makanan di alam. Salah satu studi penerapan ACO pada VRP adalah *Hybrid Ant System Algorithm with 2-opt exchange and saving algorithm for the VRP* (Bullnheimer,

dkk., 1997). Penelitian tersebut menghasilkan solusi yang baik namun dalam proses optimasinya tidak memiliki rentang solusi yang cukup luas sehingga solusi yang dicapai cenderung menuju solusi optimum lokal. Peneliti lain yang menerapkan ACO pada VRP adalah Bell dan Mcmullen (2004). Penelitian Bell dan Mcmullen (2004) menghasilkan solusi yang lebih baik dibanding penelitian Bullnheimer (1997) namun waktu komputasinya tidak optimal.

Bin, dkk. (2009) mengusulkan perbaikan ACO dalam penelitiannya yang berjudul "*An Improved Ant Collony Optimization for Vehicle Routing Problem*". *Improved Ant Collony Optimization* (IACO) dalam penelitian tersebut menggunakan operasi mutasi yang merujuk pada *Genetic Algorithm* (Yu, dkk., 2007) dan metode perubahan dua operator yang merujuk pada *2-opt exchange* (Bullnheimer, dkk., 1997) dengan tujuan untuk menghasilkan rentang solusi lebih lebar pada proses pencarian solusi VRP dibanding menggunakan algoritma ACO biasa. Ide pokok operasi mutasi dan perubahan dua operator adalah memutasi jalur distribusi yang sudah dikonstruksi oleh metode ACO biasa dengan tujuan memproduksi sebuah solusi baru yang berselisih kecil dari solusi aslinya. Pada skripsi ini akan dikaji ulang sebuah perbaikan ACO dengan metode baru pada aturan pembaharuan *pheromone* yang memadukan solusi optimum global dan solusi optimum lokal, yaitu sebuah operasi mutasi dan metode perubahan dua operator bagi VRP yang mengacu pada Bin, dkk. (2009).

1.2 Rumusan Masalah

Berdasarkan latar belakang, maka rumusan masalah dalam skripsi ini adalah sebagai berikut:

1. Bagaimana menyelesaikan VRP menggunakan metode IACO?
2. Bagaimana performa metode IACO dalam penyelesaian VRP?

1.3 Tujuan Penelitian

Dari rumusan masalah tersebut, tujuan dari skripsi ini adalah sebagai berikut:

1. Menyelesaikan VRP menggunakan metode IACO.
2. Mengukur performa metode IACO dalam penyelesaian VRP.

1.4 Manfaat Penelitian

Manfaat yang ingin dicapai skripsi ini adalah menguji performa metode IACO dalam mencari solusi paling optimal pada VRP dengan parameter uji jarak optimal dan waktu komputasi.

BAB II
TINJAUAN PUSTAKA

2.1 Graf

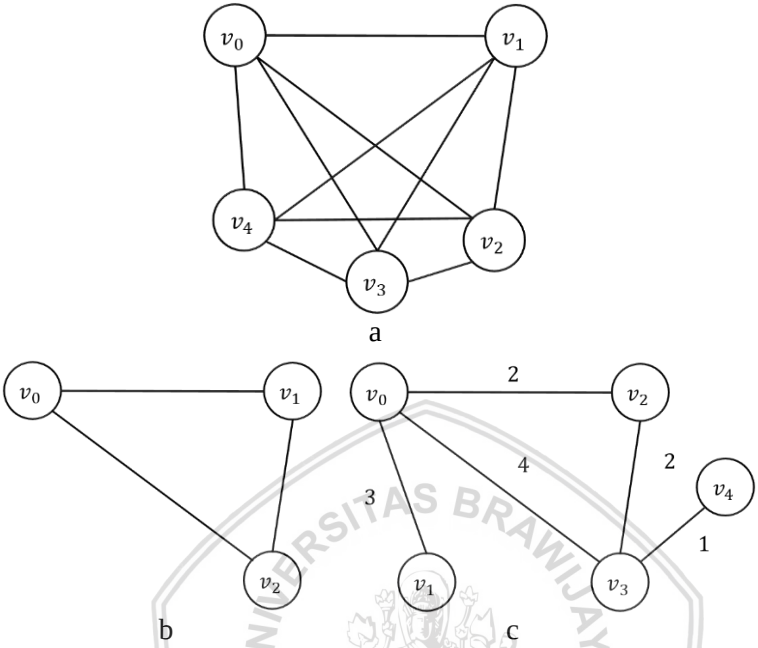
Graf G didefinisikan sebagai himpunan berhingga $V(G)$ yang tidak kosong dimana anggotanya disebut titik (*vertex*). Garis (*edge*) adalah himpunan $E(G)$ yang anggotanya terdiri dari pasangan dua elemen berbeda dari $V(G)$. Elemen dari $V(G)$ dinotasikan dengan v_i dan elemen dari $E(G)$ dinotasikan dengan e_i. Jika terdapat garis e yang menghubungkan titik v_i dan v_j, maka v_i dikatakan terhubung (*adjacent*) dengan v_j sehingga dalam hal ini v_i dan v_j dikatakan insiden dengan e. *Order* dari grapf G adalah banyaknya titik-titik pada graf G dan dinotasikan dengan $|V(G)|$. *Walk* dari graf G adalah urutan secara bergantian titik elemen $V(G)$ dan garis-garis elemen $E(G)$ yang terbentuk adalah:

$$W : v_0, e_1, v_1, e_2, v_2, \dots, v_{n-1}, e_n, v_n, (n \geq 0)$$

yang dimulai dan diakhiri dengan titik, sehingga setiap garis *incident* dengan titik sebelumnya. *Path* adalah *walk* dimana tidak ada titik yang diulang. *Cycle* adalah *walk* $v_1, v_2, \dots, v_n, n \geq 3$, dengan $v_0 = v_n$, dan titik-titik $v_1, v_2, \dots, v_n$ semuanya berbeda. *Cycle* dengan panjang n memiliki n titik. Misalkan u dan v titik-titik dalam graf G, u dikatakan terhubung (*connected*) pada v jika terdapat path u-v. Graf G dikatakan terhubung jika setiap dua titiknya dihubungkan oleh suatu *path*. Graf lengkap adalah suatu graf dengan *order p* dimana setiap dua titik yang berbeda selalu *adjacent*. Konstruksi graf yang akan dilakukan pada skripsi ini adalah konstruksi graf berbobot. Graf Berbobot adalah graf yang masing-masing sisinya diberi nilai bilangan real positif.

(Chartrand dan Oellerman, 1993)

Secara umum, teori graf yang diperlukan untuk mengonstruksi VRP adalah graf lengkap, *cycle*, dan graf berbobot. Graf lengkap adalah graf VRP secara keseluruhan tanpa besaran jarak pada sisi-sisinya, *cycle* adalah subrute yang terbentuk sebagai solusi VRP, dan graf berbobot adalah teori graf untuk menginisialisasi besaran jarak pada graf VRP. Contoh graf lengkap, *cycle*, dan graf berbobot dapat dilihat pada Gambar 2.1.



Gambar 2.1 Graf lengkap (a), *cycle* (b), dan graf berbobot (c)

2.2 Jarak *Euclid*

Jarak *Euclid* adalah metode perhitungan jarak antara dua titik dalam sistem koordinat dimensi dua. Kedua titik tersebut memiliki dua parameter yaitu koordinat X dan Y. Koordinat tersebut digunakan untuk menghitung jarak dari dua titik tersebut. Formula perhitungan yang digunakan adalah sebagai berikut:

$$d_{i,j} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$

dimana (x_i, y_i) merupakan koordinat titik dengan indeks i, (x_j, y_j) merupakan koordinat titik dengan indeks j, dan $d_{i,j}$ merupakan besar jarak dari kedua titik tersebut. Pada VRP, perhitungan nilai $d_{i,j}$ dengan metode jarak *euclid* menghasilkan matriks jarak *(D)* antara *n* buah titik dengan ukuran *n* baris dan *n* kolom.

(Danielsson, 1980)

2.3 *Vehicle Routing Problem* (VRP)

Vehicle Routing Problem (VRP) didefinisikan sebagai graf $G = (C, L)$ dimana titik-titik di dalam graf tersebut dilambangkan dalam $C = (c_0, c_1, c_2, \dots, c_n)$ dan jalur perjalanannya dilambangkan dalam $L = \{(c_i, c_j): i \neq j\}$. Dalam model graf ini, c_0 adalah gudang pusat (depot) dan titik-titik lain adalah n pelanggan yang harus dilayani. Masing-masing titik terhubung dengan jumlah permintaan logistik yang harus dikirim (q_i). Masing-masing garis penghubung (c_i, c_j) terhubung sebuah nilai $d_{i,j}$ yang melambangkan jarak antara c_i dan c_j, sehingga graf yang terkonstruksi dalam VRP adalah graf berbobot. Masing-masing perjalanan dimulai dari dan diakhiri di gudang c_0, masing-masing titik c_i harus dikunjungi tepat satu kali, dan jumlah logistik yang harus dikirim dalam satu subrute VRP tidak boleh melebihi kapasitas angkut Q.

(Bin, dkk., 2009)

Pada penerapan VRP di kehidupan sehari-hari terdapat faktor-faktor eksternal yang berpengaruh, sehingga muncul jenis-jenis variasi VRP, yaitu:

1. *Capacitated Vehicle Routing Problem* (CVRP)
   CVRP adalah VRP yang memiliki sistem sejumlah kendaraan dengan kapasitas daya angkut tertentu dan harus melayani sejumlah permintaan pelanggan untuk satu komoditas dari sebuah depot menggunakan biaya minimum. Pada dasarnya CVRP sama seperti VRP dengan faktor tambahan yaitu setiap kendaraan mempunyai kapasitas tertentu untuk satu komoditas. CVRP bertujuan meminimalisasi jumlah kendaraan dan total waktu perjalanan, dan total permintaan barang untuk tiap rute tidak melebihi kapasitas kendaraan yang melewati rute tersebut.
2. *Vehicle Routing Problem with Time Windows* (VRPTW)
   VRPTW adalah VRP dengan factor yaitu bahwa setiap pelanggan harus disuplai dalam jangka waktu tertentu.
3. *Mulitple Depot Vehicle Routing Problem* (MDVRP)
   MDVRP adalah VRP dengan faktor yaitu distributor memiliki banyak depot untuk menyuplai pelanggan. Sebuah perusahaan yang memiliki lebih dari satu depot, dan pelanggan-pelanggannya tersebar di sekitar depot-depot yang ada, maka masalah pendistribusiannya harus dimodelkan menjadi sebuah

kumpulan dari beberapa VRP yang independen. Namun, jika pelanggan dan depot-depot tidak terkumpul secara teratur maka masalahnya menjadi Multi-Depot VRP atau MDVRP. Sebuah MDVRP membutuhkan pengaturan para pelanggan ke depot-depot yang ada. Setiap kendaraan berangkat dari satu depot melayani pelanggan-pelanggan yang sudah ditentukan oleh depot tersebut, dan kembali lagi ke depot tersebut. Tujuan utama dari MDVRP adalah untuk melayani semua pelanggan, sementara jumlah kendaraan dan jarak perjalanan diminimalisasi.

4. *Vehicle Routing Problem with Pick-Up and Delivering* (VRPPD)
   VRPPD adalah VRP dengan faktor yaitu pelanggan dimungkinkan mengembalikan barang pada depot asal. Barang yang dikembalikan dapat dimasukkan ke dalam kendaraan pengantar. Perencanaan pengantaran menjadi lebih sulit dan dapat mengakibatkan penyalahgunaan kapasitas kendaraan, memperbesar jarak perjalanan atau kendaraan yang diperlukan lebih dari yang seharusnya.
5. *Split Delivery Vehicle Routing Problem* (SDVRP)
   SDVRP adalah perluasan VRP dimana setiap pelanggan dapat dilayani dengan kendaraan yang berbeda bilamana biayanya dapat dikurangi. Perluasan ini dapat dilaksanakan jika jumlah permintaan pelanggan sama dengan kapasitas kendaraan. Tujuan dari SDVRP untuk meminimalisasikan jumlah kendaraan dan total waktu perjalanan untuk pelayanan.
6. *Stochastic Vehicle Routing Problem* (SVRP)
   SVRP adalah VRP dengan faktor yaitu munculnya *random values* (seperti jumlah pelanggan, jumlah permintaan, waktu pelayanan atau waktu perjalanan). Untuk memperoleh solusi dari SVRP, maka masalah harus dibagi dalam dua tahap, solusi pada tahap pertama ditentukan sebelum *variable random* diketahui. Pada tahap kedua pengoreksian dilakukan jika nilai dari *variable random* sudah diketahui.
7. *Periodic Vehicle Routing Problem* (PVRP)
   PVRP merupakan VRP yang digeneralisasi dengan memperluas rentang perencanaan pengiriman menjadi d hari, dari semula hanya dalam rentang sehari, dengan tujuan meminimalisasi

jumlah kendaraan dan total waktu perjalanan untuk melayani tiap pelanggan.

(Tarigan, 2008)

Jika ditinjau pada artikel Bin, dkk. (2009), maka jenis VRP pada skripsi ini dapat pula dikategorikan sebagai *Capacitated Vehicle Routing Problem* (CVRP). VRP yang juga merupakan CVRP memiliki dua batasan dalam proses optimasi solusinya, yaitu:

1. Setiap titik tujuan hanya dilewati tepat satu kali pada setiap proses distribusi.
2. Total permintaan dari titik-titik tujuan pada satu subrute yang terbentuk harus kurang dari atau sama dengan batas kapasitas angkut kendaraan.

2.4 Metode Heuristik

Penyelesaian VRP pada skripsi ini menggunakan metode heuristik. Metode heuristik adalah metode yang digunakan untuk mencari solusi melalui semua kemungkinan yang ada, tetapi dalam pencariannya tidak bisa dijamin akan ditemukan solusi terbaik, sehingga algoritma ini juga disebut dengan metode perkiraan. Simulasi pencarian solusi yang diterapkan pada metode heuristik mengandalkan konsep *trial and error*.

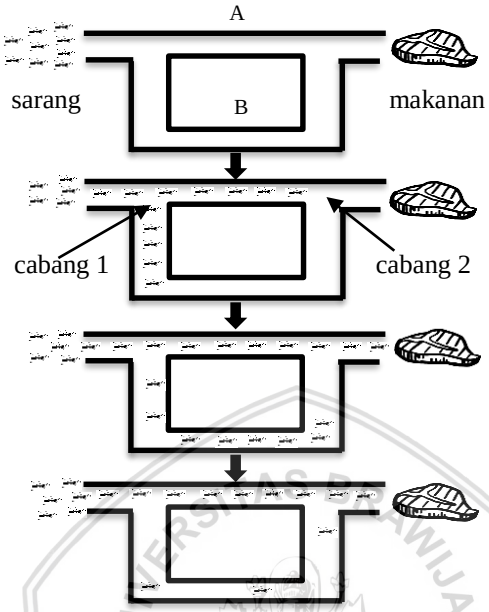
(Foulds, 1984)

Penerapan metode heuristik dalam berbagai masalah kehidupan memunculkan bermacam-macam metode optimasi. Contoh optimasi menggunakan metode heuristik adalah *Simulated Annealing, Genetics Algorithm, Tabu Search*, dan *Ant Colony Optimization*.

(Bin, dkk., 2009)

2.5 *Ant Colony Optimization* (ACO)

Pada tahun 1989, Goss dkk. melakukan penelitian terhadap perilaku semut *Iridomyrmex Humilis* (semut Argentina) yang menggunakan *pheromone* dalam menentukan jalur terpendek menuju sumber makanan. *Pheromone* adalah zat kimia yang berasal dari kelenjar endokrin dan digunakan makhluk hidup untuk mengenali sesama jenis dalam satu spesies. Percobaan tersebut seperti terlihat pada Gambar 2.2.



Gambar 2.2 Ilustrasi percobaan Goss, dkk. (1989)

Pada Gambar 2.2 terlihat semut-semut meninggalkan sarang secara bersamaan. Kemudian semut-semut tersebut sampai pada titik cabang 1 bersama-sama dan membuat keputusan dengan probabilitas yang sama untuk memilih cabang mana yang akan dilewati. Pada tahap 3 terlihat bahwa sekelompok semut memilih cabang yang lebih pendek dan mencapai sumber makanan lebih dulu (melalui jalur A), lalu mengambil makanan dan melakukan perjalanan kembali ke sarang. Ketika sampai pada titik cabang 2, semut tersebut mendeteksi *pheromone* telah terdapat pada jalur A sedemikian sehingga jalur A memiliki probabilitas lebih tinggi untuk dipilih dibanding jalur B yang belum terdapat *pheromone* pada cabang 2 karena belum ada semut yang melewatinya. Pada akhirnya semut-semut tersebut akan mengambil jalur A untuk menuju sumber makanan karena memliki probabilitas yang tinggi dan *pheromone* yang ada pada jalur B tidak mengalami penguatan lagi karena tidak ada semut yang menaruh *pheromone* pada jalur tersebut sehingga *pheromone* akan menguap dan semut akan berjalan pada jalan yang paling pendek, yaitu jalur A. (Gaertner, 2004)

Dari percobaan tersebut disusun sebuah algoritma untuk mencari solusi permasalahan optimalisasi kombinatorik yang diberi nama *Ant Colony Optimization* (ACO). Algoritma ini menyatakan permasalahan ke dalam sebuah graf, lalu menyimulasikan perjalanan semut pada setiap cabang dari titik satu ke titik yang lain sehingga membentuk suatu *path* yang merupakan representasi dari solusi permasalahan tersebut.

ACO pada skripsi ini diterapkan untuk mencari rute terpendek. Rute terpendek adalah suatu rute dengan jumlah bobot terkecil dari graf berbobot yang diperoleh dari langkah sebelumnya. Langkah-langkah untuk menentukan rute terpendek menggunakan ACO adalah:

1. Inisialisasi parameter-parameter ACO, yaitu:
   a. Intensitas *pheromone* semut antar titik (τ_{ij})
   b. Jarak antar titik (d_{ij})
   c. Penentuan titik asal dan tujuan
   d. Konstanta pengendali intensitas *pheromone* semut (α)
   e. Konstanta pengendali visibilitas (β)
   f. Visibilitas antar titik $(\eta_{ij} = \frac{1}{d_{ij}})$
   g. Jumlah semut (m)
   h. Konstanta penguapan *pheromone* semut (ρ)
   i. Jumlah iterasi maksimum (NC_{max})
   j. Konstanta (K_0)
2. Menempatkan m semut pada node awal
3. Memasukan titik pertama ke dalam *tabu list*. *Tabu list* adalah tempat yang disediakan untuk penyimpanan sementara dari solusi-solusi yang dihasilkan pada tiap iterasi. Jika s menyatakan indeks urutan kunjungan dan k menunjukan indeks semut yang sedang melakukan perjalanan, maka titik yang dikunjungi dinyatakan sebagai $tabu(k,s)$.
4. Menghitung peluang *node* yang akan dikunjungi oleh semut dengan formula sebagai berikut:

$$p_{ij}^{k} = \begin{cases} \dfrac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{k' \in [N-tabu_k]}(\tau_{ik'})^{\alpha}(\eta_{ik'})^{\beta}}, & j \in [N - tabu_k] \\ 0, & j \text{ lainnya} \end{cases}$$

dengan i indeks titik asal dan j indeks titik tujuan.

5. Menghitung panjang rute setiap semut. Perhitungan panjang rute tertutup (L_k) setiap semut dilakukan setelah semua semut menyelesaikan satu siklus perjalanan. Nilai minimum panjang rute tertutup setiap siklus (L_{min}) akan diperoleh setelah L_k setiap semut diketahui.
6. Memperbaharui intensitas *pheromone* menggunakan formula sebagai berikut:

   $$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau^k{}_{ij}$$

   dengan $\Delta\tau^k{}_{ij}$ adalah perubahan intensitas *pheromone* antar titik setiap semut yang dihitung dengan persamaan:

   $$\Delta\tau^k{}_{ij} = \frac{K}{L_k} \qquad (2.1)$$

   untuk $(i,j) \in$ titik asal dan tujuan dalam $tabu_k$ dan K adalah suatu konstanta.
7. Menghitung nilai intensitas *pheromone* semut antar titik untuk siklus selanjutnya dengan formula:

   $$\tau_{ij} = \rho\tau_{ij} + \Delta\tau_{ij}$$
8. Mengosongkan *tabu list* untuk diisi lagi dengan urutan titik baru pada siklus selanjutnya. Jika jumlah siklus maksimum belum dicapai, maka algoritma diulang lagi dari langkah tiga menggunakan parameter intensitas *pheromone* semut antar titik (τ_{ij}) yang sudah diperbaharui.

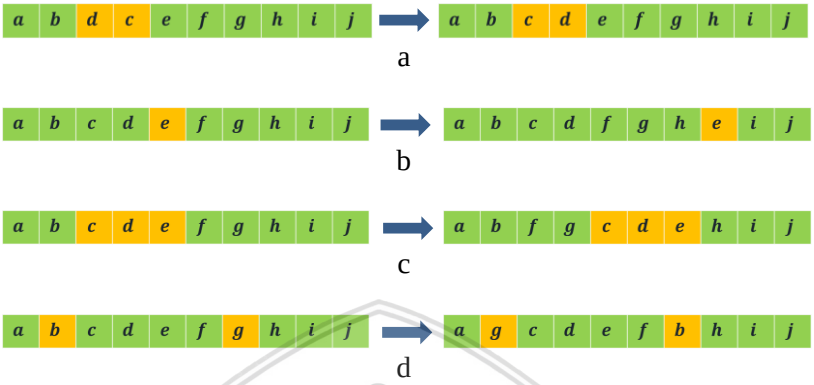
(Dorigo, dkk., 1996)

2.6 Operasi Mutasi

Mutasi merupakan proses perubahan sebagian sifat individu secara acak yang menghasilkan struktur genetik baru. Menurut Obitko (1998) terdapat empat jenis mutasi, yaitu:

a. Mutasi Inversi (*Inversion Mutation*) yaitu memilih dua posisi kromosom secara acak dan kemudian membalik untaian diantara dua posisi itu.
b. Mutasi Sisipan (*Insertion Mutation*) yaitu memilih gen secara acak dan menyisipkannya di posisi acak.
c. Mutasi Perpindahan (*Displacement Mutation*) yaitu memilih sebuah *subtour* secara acak dan memasukkannya pada posisi lain secara acak.
d. *Reciprocal Exchange Mutation* yaitu memilih dua posisi gen secara acak, kemudian menukar posisi kedua gen tersebut.

Gambar 2.3 menunjukan jenis-jenis mutasi yang sudah disebutkan dalam bentuk grafis.



Gambar 2.3 Mutasi Inversi (a), Mutasi Sisipan (b), Mutasi Perpindahan (c), dan *Reciprocal Exchange Mutation* (d)

Pada algoritma IACO yang merujuk pada artikel Bin, dkk. (2009), proses mutasi yang digunakan adalah *Reciprocal Exchange Mutation*. Berikut adalah langkah-langkah dari proses *Reciprocal Exchange Mutation* menurut Bin, dkk.:

1. Pilih dua rute secara acak dari solusi awal. Kemudian pilih satu titik dari masing-masing rute yang terpilih.
2. Tukar posisi kedua titik yang terpilih pada langkah 1.

Probabilitas mutasi yang digunakan adalah sebagai berikut:

$$P_m(t) = P_m^{min} + (P_m^{max} - P_m^{min})^{1-t/T}$$

dimana $P_m(t)$ adalah probabilitas mutasi pada iterasi ke-t, P_m^{min} adalah tingkat probabilitas mutasi terendah yang didefinisikan dengan persamaan $P_m^{min} = \frac{1}{n_c}$ dengan n_c adalah jumlah semua titik, P_m^{max} adalah tingkat probabilitas mutasi tertinggi yang didefinisikan dengan persamaan $P_m^{max} = \frac{1}{n_v}$ dengan n_v adalah jumlah rute yang terbentuk pada solusi, t adalah iterasi saat ini, dan T adalah iterasi maksimum yang diberikan.

(Bin, dkk., 2009)

2.7 *Local Search* dan Metode Perubahan Dua Operator

Menurut Stützle (1998) *Local Search* adalah metode untuk mengidentifikasi sebuah solusi dari suatu permasalahan dengan mempertimbangkan solusi-solusi potensial yang tersedia sampai ditemukan satu solusi yang memenuhi kriteria. Pada skripsi ini metode *local search* memiliki tujuan untuk mengoptimalkan waktu komputasi Algoritma IACO. Jenis metode *local search* yang digunakan adalah metode perubahan dua operator. Metode perubahan dua operator merupakan algoritma *local search* yang berfungsi mereduksi jalur yang bersilangan pada suatu rute tunggal dan membangun jalur baru sedemikian sehingga jarak rute yang dihasilkan lebih optimal.

Misalkan diberikan suatu rute $c_0, c_1, c_2, \dots, c_k, c_0$. Untuk setiap kombinasi pelanggan (c_i, c_j) dengan $i < j;\ i, j\ \{1, 2, \dots, k-1\}$ akan diperiksa apakah jalur dari c_{i-1} ke c_j dan dari c_i ke c_{j+1} lebih baik daripada jalur awal dari c_{i-1} ke c_i dan dari c_j ke c_{j+1}. Jika demikian, bentuk jalur baru dari c_i ke c_j dan dilanjutkan untuk jalur lainnya yang tersisa. Setelah semua jalur diperiksa, maka urutan kunjungan diperbaiki sesuai urutan perbaikan yang diperoleh. Jadi, jika urutan sebelum perbaikan adalah sebagai berikut:

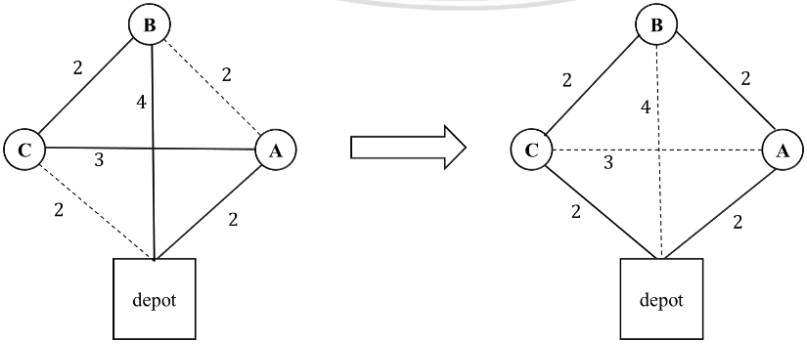
$$c_0, c_1, c_2, \dots, c_{i-1}, c_i, c_{i+1}, c_{i+2}, \dots, c_{j-1}, c_j, c_{j+1}, \dots, c_k, c_0$$

maka setelah perbaikan menjadi:

$$c_0, c_1, c_2, \dots, c_{i-1}, c_j, c_{j-1}, c_{j-2}, \dots, c_{i+1}, c_i, c_{j+1}, c_{j+2}, \dots, c_k, c_0.$$

(Konig, 2008)

Contoh metode perubahan dua operator pada graf dapat dilihat pada Gambar 2.4.



Gambar 2.4 Metode perubahan dua operator pada graf VRP

BAB III
HASIL DAN PEMBAHASAN

3.1 Data Uji

Data uji yang digunakan pada penelitian ini adalah data sekunder yang diperoleh dari penelitian sebelumnya, yaitu penelitian yang dilakukan oleh Augerat, dkk, pada tahun 1995 dan Beasley pada tahun 1990. Data uji tersebut memuat sejumlah titik (konsumen) yang harus dikunjungi (n) dan batasan kapasitas setiap kendaraan setiap kali melakukan satu perjalanan (Q). Jumlah konsumen dan kapasitas kendaraan masing-masing data disajikan pada Tabel 3.1.

Jarak antar titik pada masing-masing data dapat ditentukan dari koordinat yang telah diketahui. Koordinat serta permintaan setiap titik pada masing-masing data disertakan dalam Lampiran 1, 2, dan 3. Solusi rute dengan jarak minimum dari masing-masing permasalahan (data uji) sudah diketahui. Kriteria pemilihan tiga data uji yang digunakan adalah banyaknya jumlah konsumen, yaitu kecil (20), sedang (50), dan besar (100).

Tabel 3.1 Data uji yang digunakan

Data Uji	Jumlah Konsumen (titk)	Kapasitas Kendaraan (unit)
Augerat, dkk	20	160
Beasley	50	160
Beasley	100	200
Beasley	150	200

3.2 Tahapan Penelitian

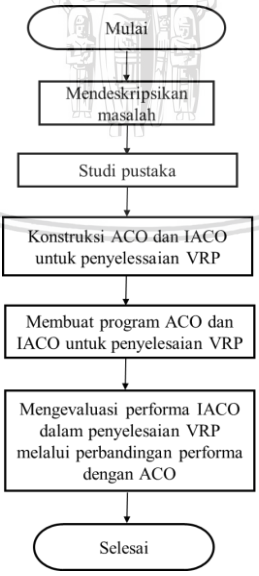
Tahapan penelitian yang digunakan dalam skripsi ini terdiri dari lima tahap, yaitu:

1. Deskripsi Masalah
   Masalah yang diselesaikan dalam skripsi ini adalah bagaimana menyelesaikan VRP menggunakan metode IACO dan bagaimana performa metode IACO untuk menyelesaikan VRP.
2. Studi Pustaka
   Tahap selanjutnya adalah melakukan studi pustaka. Pada tahap ini akan dibahas dan dikaji teori yang akan digunakan pada

pembahasan skripsi ini, yaitu Graf, Metode Heuristik, *Vehicle Routing Problem*, *Ant Colony Optimization*, Operasi Mutasi, *Local Search*, dan IACO.

3. Konstruksi IACO
   Pada tahap ini akan dibahas tentang bagaimana mengkonstruksi IACO. Proses konstruksi metode IACO terbagi menjadi tujuh tahapan, yaitu inisialisasi, konstruksi rute, cek nilai acak, mutasi, *local search*, *update pheromone*, dan cek kriteria pemberhentian.
4. Membuat Program ACO dan IACO
   Pembuatan program ACO dan IACO untuk penyelesaian VRP akan dilakukan pada aplikasi MATLAB 2017a.
5. Evaluasi Performa IACO
   Performa IACO dievaluasi dengan melihat waktu komputasi yang dibutuhkan dan rata-rata solusi terbaik yang diperoleh. Pada skripsi ini performa IACO dibandingkan dengan ACO. Solusi dari metode IACO dan ACO juga dibandingkan dengan solusi terbaik yang diketahui saat ini untuk data uji Augerat, dkk. (1995) dan Beasley (1990).

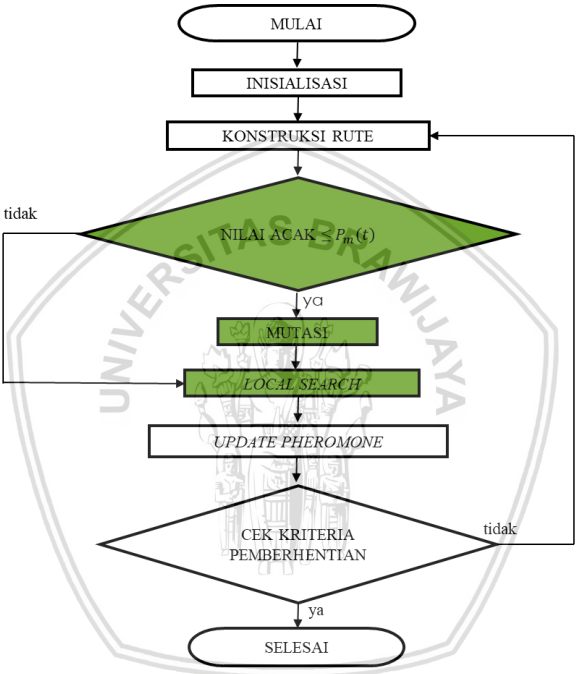
Diagram alir tahapan penelitian dapat dilihat pada Gambar 3.1.



Gambar 3.1 Diagram alir tahapan penelitian

3.3 Tahapan IACO

Pada subbab ini akan dijelaskan tahap-tahap IACO. IACO dikonstruksi oleh kombinasi algoritma ACO biasa dengan metode operasi mutasi dan metode perubahan dua operator. Diagram alir proses algoritma IACO dapat dilihat pada Gambar 3.2. Bagian diagram alir yang berwarna hijau merupakan tahapan metode IACO yang tidak terdapat pada metode ACO.



Gambar 3.2 Diagram alir algoritma IACO

3.3.1 Inisialisasi

Tahap pertama dari IACO adalah inisialisasi parameter yang akan digunakan pada tahap-tahap selanjutnya dari IACO. Parameter yang diinisialisasi pada IACO yaitu :

1. Intensitas *pheromone* semut antar titik (τ).
   Nilai τ disajikan dalam bentuk matriks $N \times N$ dengan N adalah jumlah titik pada VRP yang akan dicari solusi optimalnya. Sebagai contoh, nilai intensitas *pheromone* awal dari contoh data uji Tabel 3.2 adalah:

$$\tau = \begin{bmatrix} 0{,}00 & 0{,}01 & 0{,}01 & 0{,}01 & 0{,}01 & 0{,}01 \\ 0{,}01 & 0{,}00 & 0{,}01 & 0{,}01 & 0{,}01 & 0{,}01 \\ 0{,}01 & 0{,}01 & 0{,}00 & 0{,}01 & 0{,}01 & 0{,}01 \\ 0{,}01 & 0{,}01 & 0{,}01 & 0{,}00 & 0{,}01 & 0{,}01 \\ 0{,}01 & 0{,}01 & 0{,}01 & 0{,}01 & 0{,}00 & 0{,}01 \\ 0{,}01 & 0{,}01 & 0{,}01 & 0{,}01 & 0{,}01 & 0{,}00 \end{bmatrix}$$

Intensitas *pheromone* antar titik dapat berubah seiring berjalannya waktu, baik melalui penambahan nilai yang dilakukan jika terdapat semut yang melewati jalur tersebut saat proses mencari solusi optimal, maupun melalui pengurangan karena proses penguapan *pheromone* jika terdapat jalur yang tidak dilalui saat proses mencari solusi optimal.

2. Konstanta pengendali intensitas *pheromone* (α).
   Nilai α yang digunakan adalah 2.
3. Konstanta pengendali visibilitas (β)
   Nilai β yang digunakan adalah 1.
4. Konstanta pengali (K_0).
   Nilai K_0 yang digunakan adalah 1000.
5. Kriteria pemberhentian.
   Terdapat dua kriteria pemberhentian yang akan digunakan yaitu ketika jumlah iterasi telah mencapai nilai T (1500) atau ketika tidak terdapat perubahan nilai solusi optimal global pada 500 iterasi secara beruntun sebelum mencapai iterasi maksimum (T).
6. Konstanta penguapan *pheromone* (ρ).
   Pada skripsi ini, nilai konstanta penguapan *pheromone* (ρ) akan dibuat beragam yaitu 0,8, 0,89, dan 0,99. Tujuan dari penggunaan nilai yang beragam pada parameter tersebut adalah untuk mengetahui pengaruh dari nilai parameter tersebut pada performa IACO. Semakin besar nilai ρ, maka penguapan yang terjadi semakin sedikit.
7. Jumlah semut (m).
   Selain konstanta penguapan *pheromone* (ρ), jumlah semut yang diinisialisasi juga akan dibuat beragam yaitu 1, 5, dan 10, dengan tujuan sama dengan penggunaan parameter yang beragam pada konstanta penguapan *pheromone* (ρ).

Nilai α, β, dan K_0 yang digunakan merujuk pada penelitian yang dilakukan oleh Bin, dkk. (2009).

Tabel 3.2 Contoh data uji

Titik	Koordinat X	Koordinat Y	Permintaan
1 (Depot)	5	5	0
2	3	2	3
3	4	6	1
4	2	5	4
5	0	1	1
6	2	1	1

3.3.2 Konstruksi rute

Konstruksi rute merupakan tahapan algortima IACO dalam membentuk suatu rute solusi VRP menggunakan metode yang sama dengan ACO biasa. Sebagai contoh pengerjaan, jumlah semut yang akan diperjalankan adalah lima ekor. Langkah pertama dari proses konstruksi rute adalah menghitung jarak antar titik ($d_{i,j}$), dimana i dan j adalah indeks dari suatu titik pada data uji, sehingga $d_{i,j}$ merupakan notasi dari jarak antara titik i dan titik j. Jarak antar titik diperoleh melalui penghitungan menggunakan metode jarak *euclid* dengan koordinat sesuai data uji yang digunakan. Besar jarak dari seluruh titik disimpan sebagai sebuah matriks jarak (D) dengan ukuran $N \times N$, dimana N adalah jumlah titik yang ada dan $d_{i,j}$ merupakan anggotanya. Contoh data uji yang akan digunakan untuk simulasi metode IACO dapat dilihat pada Tabel 3.2.

Penghitungan jarak antara enam titik tersebut dilakukan dengan cara memasukan x_i dan x_j dimana x_i dan x_j adalah seluruh anggota dari koordinat X beserta y_i dan y_j dimana y_i dan y_j adalah seluruh anggota dari koordinat Y pada persamaan:

$$d_{i,j} = \sqrt{\left(x_j - x_i\right)^2 + \left(y_j - y_i\right)^2}.$$

Setelah melakukan perhitungan jarak seluruh titik pada contoh data uji Tabel 3.2 dengan metode jarak *euclid*, maka diperoleh matriks jarak (D) sebagai berikut:

$$D = \begin{bmatrix} 0{,}00 & 3{,}61 & 1{,}41 & 3{,}00 & 6{,}40 & 5{,}00 \\ 3{,}61 & 0{,}00 & 4{,}12 & 3{,}16 & 3{,}16 & 1{,}41 \\ 1{,}41 & 4{,}12 & 0{,}00 & 2{,}24 & 6{,}40 & 5{,}39 \\ 3{,}00 & 3{,}16 & 2{,}24 & 0{,}00 & 4{,}47 & 4{,}00 \\ 6{,}40 & 3{,}16 & 6{,}40 & 4{,}47 & 0{,}00 & 2{,}00 \\ 5{,}00 & 1{,}41 & 5{,}39 & 4{,}00 & 2{,}00 & 0{,}00 \end{bmatrix}.$$

Selain digunakan untuk menentukan jarak akhir dari rute solusi yang terbentuk, besaran jarak antar titik juga digunakan untuk memperoleh nilai visibilitas antar titik (η), yaitu dengan persamaan:

$$\eta_{i,j} = \frac{1}{d_{i,j}} \tag{3.1}$$

dimana i dan j adalah indeks dari suatu titik pada data uji. Nilai tersebut digunakan dalam perhitungan probabilitas sebuah titik akan dipilih sebagai kota tujuan selanjutnya. Dari nilai D dan Persamanan 3.1 pada contoh data uji Tabel 3.2 diperoleh nilai η sebagai berikut:

$$\eta = \begin{bmatrix} und & 0{,}28 & 0{,}71 & 0{,}33 & 0{,}16 & 0{,}20 \\ 0{,}28 & und & 0{,}24 & 0{,}32 & 0{,}32 & 0{,}71 \\ 0{,}71 & 0{,}24 & und & 0{,}45 & 0{,}16 & 0{,}19 \\ 0{,}33 & 0{,}32 & 0{,}45 & und & 0{,}22 & 0{,}25 \\ 0{,}16 & 0{,}32 & 0{,}16 & 0{,}22 & und & 0{,}50 \\ 0{,}20 & 0{,}71 & 0{,}19 & 0{,}25 & 0{,}50 & und \end{bmatrix}.$$

Nilai η_{ii} yang berupa suatu bilangan yang tidak terdefinisi (und) tidak menimbulkan masalah, karena tidak digunakan pada tahap selanjutnya.

Langkah selanjutnya adalah menentukan fungsi objektif dari VRP. Tujuan penentuan fungsi objektif ini adalah untuk menyesuaikan proses optimasi dengan kasus yang diajukan sehingga solusi yang diperoleh dari metode IACO ini dapat optimal sesuai dengan batasan kasus yang diberikan. VRP yang merujuk pada artikel Bin, dkk (2009) merupakan VRP dengan batasan kapasitas angkut kendaraan, sehingga fungsi yang harus dipenuhi oleh setiap subrute yang terbentuk adalah:

$$\sum_{i=1}^{c_l} q_{l,i} \leq Q,\ 1 \leq l \leq v,$$

dimana c_l adalah banyak titik pada subrute ke-l yang terbentuk sebagai solusi, v adalah jumlah subrute yang terbentuk, dan $q_{l,i}$ adalah

permintaan pada titik ke-i pada subrute ke-l. Fungsi objektif dari VRP yang akan dioptimasi adalah:

$$JT = \sum_{i=1}^{v} \left(\sum_{j=1}^{c_i - 1} d_{\vec{X}_{i,j}, \vec{X}_{i,j+1}} \right) + d_{\vec{X}_{i,c_i}, \vec{X}_{i,1}},$$

dimana JT adalah jarak total dari rute yang terbentuk, $\vec{X}_i$ adalah vektor urutan titik tujuan subrute ke-i dari rute yang terpilih sebagai solusi terbaik, $d_{\vec{X}_{i,j}, \vec{X}_{i,j+1}}$ adalah anggota matriks D dengan posisi baris sesuai dengan nilai $\vec{X}_{i,j}$, posisi kolom sesuai dengan nilai $\vec{X}_{i,j+1}$, dan $\vec{X}_{i,1}$ adalah 1 (depot). Proses konstruksi rute dengan batasan $Q = 5$ sesuai contoh data uji pada Tabel 3.2 memberi informasi bahwa sebelum semut memilih antara lima titik untuk dijadikan tujuan, semut akan terlebih dahulu mengeliminasi titik tujuan dengan dua kondisi, yaitu:

1. Titik tujuan (selain depot) tersebut sudah dikunjungi sebelumnya.
2. Permintaan pada titik tersebut tidak memenuhi batas kapasitas angkut yang ditentukan.

Misalkan ketika semut telah menempuh perjalan dari titik 1 ke titik 2, maka titik lain dengan dua kondisi tersebut akan dieliminasi dari daftar calon titik tujuan. Titik yang dieliminasi pada kondisi ini adalah titik 2 dan titik 4, karena titik 2 telah dikunjungi dan titik 4 memiliki besar permintaan 4, dimana besar permintaan tersebut melebihi kapasitas angkut yang tersisa yaitu 2. Setelah mengeliminasi titik yang memenuhi kondisi untuk dieliminasi, proses penentuan titik tujuan akan diawali dengan penghitungan nilai probabilitas masing-masing calon titik tujuan untuk dipilih menggunakan formula:

$$p_{i,j}^{k} = \begin{cases} \dfrac{(\tau_{i,j})^{\alpha} (\eta_{i,j})^{\beta}}{\sum_{k' \in [S - tabu_k]} (\tau_{i,k'})^{\alpha} (\eta_{i,k'})^{\beta}}, & j \in [S - tabu_k] \\ 0, & j \text{ lainnya} \end{cases}$$

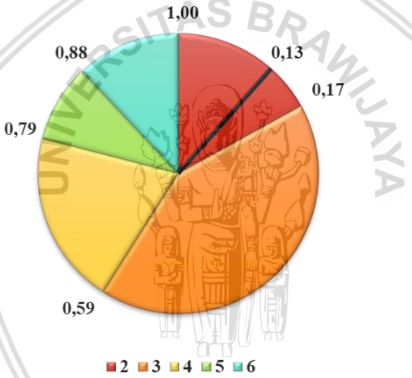
dengan i adalah indeks titik asal, j adalah indeks titik tujuan, S himpunan dengan anggota seluruh titik, dan $tabu_k$ adalah titik yang tereliminasi. Sesuai dengan artikel Bin, dkk (2009), nilai α yang digunakan adalah dua dan nilai β yang digunakan adalah satu. Sebagai contoh ketika semut memutuskan berangkat dari depot (titik 1), maka titik yang mungkin dikunjungi adalah:

$$W = [2\ 3\ 4\ 5\ 6],$$

dengan W adalah titik yang *available* untuk dipilih dan nilai probabilitas untuk dipilih serta jumlahan kumulatifnya akan disajikan pada Tabel 3.3.

Tabel 3.3 Probabilitas depot ke W subrute 1

W	Probabilitas Dipilih (P)	Kumulatif P
2	0,17	0,17
3	0,42	0,59
4	0,20	0,79
5	0,09	0,88
6	0,12	1,00



Gambar 3.3 Visualisasi nilai kumulatif P Tabel 3.3 dan penggunaan nilai r

Langkah berikutnya adalah membangkitkan nilai acak antara nol sampai satu. Misalkan nilai acak yang muncul adalah 0,13, maka indeks terkecil jumlahan kumulatif dari probabilitas titik tujuan yang lebih besar atau sama dengan 0,13 menjadi indeks (h) bagi vektor W untuk kemudian W ke-h dipilih menjadi titik tujuan. Pada contoh data uji diperoleh informasi bahwa titik yang terpilih adalah titik 2. Visualisasi nilai kumulatif P Tabel 3.3 dan penggunaan nilai r tersaji pada Gambar 3.3 untuk mempermudah pembaca dalam memahami konsep pemilihian jalur pada metode optimasi yang digunakan.

Langkah selanjutnya adalah mengeliminasi titik 2 dari vektor W, sehingga titik 2 tidak akan dipilih kembali sebagai titik tujuan. Vektor titik tujuan yang baru adalah:

$$W = [3\ 4\ 5\ 6].$$

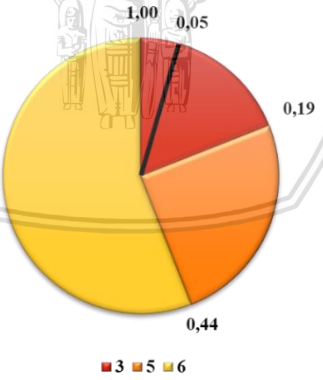
Sebelum menghitung nilai probabilitas untuk dipilih, akan dilakukan terlebih dahulu pembaharauan nilai kapasitas angkut kendaraan dengan cara mengurangkan nilai Q dengan permintaan titik 2, yaitu 3. Dengan demikian nilai $Q = 2$ dan titik yang dapat dikunjungi oleh kendaraan yaitu:

$$W = [3\ 5\ 6],$$

dengan nilai probabilitas untuk dipilih dan jumlahan kumulatifnya dapat dilihat pada Tabel 3.4.

Tabel 3.4 Probabilitas titik 2 ke W subrute 1

W	**Probabilitas Dipilih (P)**	**Kumulatif *P***
3	0,19	0,19
5	0,25	0,44
6	0,56	1,00



Gambar 3.4 Visualisasi nilai kumulatif P Tabel 3.4 dan penggunaan nilai r

Langkah selanjutnya adalah bangkitkan nilai acak antara nol sampai satu. Misalkan nilai acak yang muncul adalah 0,05 (r), maka titik tujuan selanjutnya adalah 3 dan rute yang terbentuk adalah $1 - 2 - 3$.

Visualisasi nilai kumulatif P Tabel 3.4 dan penggunaan nilai r disajikan pada Gambar 3.4.

Selanjutnya eliminasi titik 3 dari vektor W. Setelah proses eliminasi tersebut, nilai Q menjadi 1 dan vektor titik tujuan yang mungkin menjadi sebagai berikut:

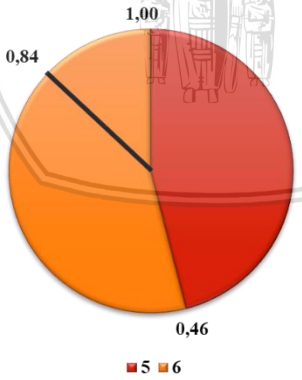
$$W = [5\ 6].$$

dengan nilai probabilitas untuk dipilih dan jumlahan kumulatifnya tersaji pada Tabel 3.5.

Tabel 3.5 Probabilitas dari titik 3 ke W subrute 1

W	**Probabilitas Dipilih (P)**	**Kumulatif P**
5	0,46	0,46
6	0,54	1,00

Langkah selanjutnya adalah bangkitkan nilai acak antara nol sampai satu. Misalkan nilai acak yang muncul adalah 0,84 (r), maka titik tujuan selanjutnya adalah 6 dan rute yang terbentuk adalah $1-2-3-6$. Visualisasi nilai kumulatif P Tabel 3.5 dan penggunaan nilai r tersaji pada Gambar 3.5.



Gambar 3.5 Visualisasi nilai kumulatif P Tabel 3.5 dan penggunaan nilai r

Dikarenakan nilai Q telah habis, maka kendaraan akan kembali menuju depot dan nilai Q menjadi 5 kembali. Pada kondisi ini rute

yang terbentuk adalah $1-2-3-6-1$ dan rute tersebut merupakan subrute pertama yang terbentuk.

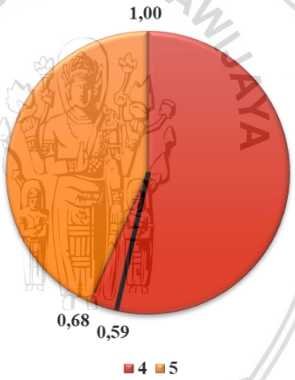
Proses selanjutnya adalah mendeteksi kembali titik-titik tujuan yang belum dikunjungi. Pada tahap ini titik tujuan yang belum dikunjungi adalah:

$$W = [4\ 5],$$

dengan nilai probabilitas untuk dipilih dan jumlahan kumulatifnya disajikan pada Tabel 3.6.

Tabel 3.6 Probabilitas dari depot ke W subrute 2

W	**Probabilitas Dipilih (P)**	**Kumulatif P**
4	0,68	0,68
5	0,32	1,00



Gambar 3.6 Visualisasi nilai kumulatif P Tabel 3.6 dan penggunaan nilai r

Tahap selanjutnya adalah mendapatkan titik tujuan yang akan dipilih dengan cara membangkitkan bilangan acak r. Misalkan nilai r yang muncul adalah 0,59, maka titik yang terpilih sebagai titik tujuan selanjutnya adalah 4. Visualisasi nilai kumulatif P Tabel 3.6 dan penggunaan nilai r dalam bentuk grafis disajikan pada Gambar 3.6. Dikarenakan titik yang mungkin hanya 5 dengan besar permintaan 1 dan nilai $Q = 1$, maka kendaraan tidak perlu kembali ke depot

sebelum mengunjungi titik 5, sehingga rute akhir yang terbentuk adalah $1-2-3-6-1-4-5-1$.

Lakukan proses yang sama sebanyak nilai m yang telah diinisialisasi, yaitu lima kali. Sebagai contoh, rute solusi VRP yang diperoleh dari simulasi lima semut dengan contoh data uji ditampilkan pada Tabel 3.7. Dari data Tabel 3.7, solusi yang dipilih dan digunakan untuk tahap optimasi selanjutnya adalah solusi dengan nilai jarak tempuh paling minimum, yaitu rute solusi semut 3, dengan informasi subrute sebagai berikut:

Subrute(1): $1-2-6-3-1$,
Subrute(2): $1-4-5-1$.

Tabel 3.7 Contoh rute solusi yang terbentuk

Semut	Rute	Jumlah Subrute	Jarak Tempuh
1	1-2-3-6-1-4-5-1	2	31,9891
2	1-3-2-5-1-4-6-1	2	27,1027
3	1-2-6-3-1-4-5-1	2	25,6944
4	1-3-5-6-1-4-1-2-1	3	28,0284
5	1-3-6-5-1-4-1-2-1	3	28,4136

3.3.3 Cek nilai acak

Tahap IACO selanjutnya adalah proses seleksi apakah rute hasil konstruksi metode ACO biasa akan dimutasi atau tidak. Proses seleksi apakah terjadi mutasi atau tidak dijelaskan oleh Bin, dkk (2009) sebagai berikut:

Langkah 1: hitung nilai probabilitas terjadi mutasi $(P_m(t))$ dengan formula:

$$P_m(t) = P_m^{min} + (P_m^{max} - P_m^{min})^{1-t/T},$$

dimana P_m^{min} adalah tingkat probabilitas mutasi terendah yang didefinisikan dengan persamaan $P_m^{min} = \frac{1}{N}$ dengan N adalah jumlah semua titik tujuan (selain depot), P_m^{max} adalah tingkat probabilitas mutasi tertinggi yang didefinisikan dengan persamaan $P_m^{max} = \frac{1}{v}$ dengan v adalah jumlah rute yang terbentuk pada solusi, t adalah

iterasi saat ini, dan T adalah iterasi maksimum yang diberikan, yaitu 1500. Sesuai dengan nilai variabel pada Tabel 3.7, diperoleh nilai N adalah lima dan v adalah dua, sehingga diperoleh nilai probabilitas terjadi mutasi pada iterasi satu ($P_m(1)$), yaitu 0,50.

Langkah 2: pilih secara acak dua subrute yang akan dimutasi. Karena banyak subrute dari rute yang akan dimutasi adalah dua, maka kedua subrute pada rute solusi dari tahap sebelumnya yang dipilih.

Langkah 3: Bangkitkan nilai acak antara nol dan satu pada masing-masing titik tujuan (selain depot) untuk dijadikan parameter apakah terjadi mutasi dengan membandingkan nilai-nilai tersebut dengan $P_m(1)$. Pembangkitan bilangan acak untuk masing-masing titik tujuan disajikan pada Tabel 3.8.

Langkah 4: Jika terdapat lebih banyak atau sama dengan satu titik pada masing-masing subrute yang memiliki nilai acak kurang dari atau sama dengan nilai $P_m(1)$ maka operasi mutasi akan dilakukan. Apabila tidak terdapat titik dengan nilai acak kurang dari atau sama dengan nilai $P_m(1)$ pada masing-masing subrute yang dipilih, maka operasi mutasi tidak dilakukan. Jika pada sebuah subrute terdapat lebih dari satu titik yang memiliki nilai acak kurang dari atau sama dengan nilai $P_m(1)$, maka titik yang dipilih untuk mutasi dari subrute tersebut adalah titik yang memiliki nilai acak paling kecil. Sesuai dengan data Tabel 3.8 dan proses seleksi, operasi mutasi akan dilakukan antara titik 3 dan titik 5.

Tabel 3.8 Nilai acak mutasi

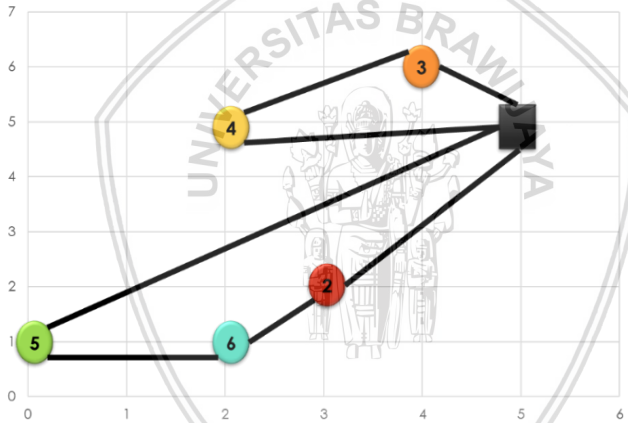
Titik	Nilai acak	Subrute	Mutasi
2	0,40	1	Ya
6	0,58	1	Tidak
3	**0,33**	**1**	**Ya**
4	0,87	2	Tidak
5	**0,02**	**2**	**Ya**

3.3.4 Operasi mutasi

Merujuk pada artikel Bin, dkk (2009), jenis operasi mutasi yang digunakan adalah *Reciprocal Exchange Mutation*, yaitu menukar posisi antara dua titik yang telah ditentukan pada proses seleksi sebelumnya. Tujuan dari improvisasi metode ACO ini adalah untuk

menghasilkan rentang solusi lebih lebar pada proses pencarian solusi VRP dibanding menggunakan algoritma ACO biasa, namun pada eksekusinya tidak menutup kemungkinan solusi yang dihasilkan *infeasible*. Solusi hasil mutasi mengalami kondisi *infeasible* yaitu ketika terdapat subrute dengan jumlah permintaan seluruh titiknya melebihi batas kapasitas angkut yang telah ditentukan. Berdasarkan tahap sebelumnya, operasi mutasi akan dilakukan pada titik 3 dari subrute pertama dan titik 5 dari subrute ke-2. Operasi mutasi kedua titik tersebut dapat dilihat pada Gambar 3.7 dan plot rutenya tersaji pada Gambar 3.8.

Gambar 3.7 *Reciprocal Exchange Mutation* pada rute solusi



Gambar 3.8 Plot rute hasil mutasi

Sebelum menghitung besar jarak tempuh dari rute hasil operasi mutasi dan membandingkannya dengan rute hasil konstruksi metode ACO, perlu dicek kembali apakah rute solusi yang terbentuk memiliki subrute dengan jumlah permintaan seluruh titiknya lebih dari batas kapasitas angkut (Q), yaitu 5. Data permintaan seluruh titik pada masing-masing subrute hasil operasi mutasi dapat dilihat pada Tabel 3.9. Tabel tersebut menunjukkan bahwa seluruh subrute yang terbentuk *feasible*, sehingga rute solusi hasil mutasi dapat diuji pada tahap selanjutnya yaitu proses *local search*.

Tabel 3.9 Jumlah permintaan setiap subrute

Subrute	Titik	Jumlah Permintaan	*Feasible*/*Infeasible*
1	1-2-6-5-1	5	***Feasible***
2	1-4-3-1	5	***Feasible***

3.3.5 Local search

Metode *local search* adalah metode untuk mengidentifikasi sebuah solusi dari suatu permasalahan dengan mempertimbangkan solusi-solusi potensial yang tersedia hingga ditemukan satu solusi yang memenuhi kriteria. Merujuk pada artikel Bin, dkk (2009), jenis metode *local search* yang akan digunakan adalah metode perubahan 2 operator. Metode perubahan 2 operator adalah metode *local search* dengan tahap mereduksi jalur yang bersilangan pada suatu subrute dan membangun jalur penghubung titik-titik tujuan pada subrute tersebut sedemikian sehingga jarak tempuh rute yang dihasilkan lebih optimal.

Pemilihan rute solusi yang akan dioptimasi menggunakan metode *local search* akan dilakukan sesuai dengan kondisi yang tercipta pada tahap sebelumnya, yaitu jika tejadi proses mutasi dan rute yang dihasilkan *feasible*, maka rute yang akan dioptimasi dengan metode *local search* adalah rute yang mengalami proses mutasi. Apabila pada tahap optimasi sebelumnya tidak mengalami proses mutasi atau rute hasil operasi mutasinya *infeasible*, maka rute hasil proses ACO yang akan dioptimasi dengan metode *local search*. Berdasarkan proses simulasi data uji contoh, proses *local search* akan dilakukan pada rute hasil operasi mutasi. Gambar 3.5 memberikan informasi bahwa tidak terdapat jalur yang bersilangan pada rute hasil mutasi, sehingga proses *local search* pada iterasi ini difungsikan untuk menunjukkan proses konstruksi rute menggunakan metode *local search*.

Konstruksi subrute pertama akan kembali diulang menggunakan titik-titik tujuan yang tidak berubah, yaitu titik 2, 5 dan 6. Dalam pemilihan titik tujuan pada tahap optimasi kali ini mengambil jarak terpendek antar titik sebagai acuan utamanya. Pada kasus ini, penentuan titik kedua pada subrute pertama akan dipilih dengan cara membandingkan besar jarak antara depot ke titik 2, depot ke titik 5, dan depot ke titik 6. Data jarak masing-masing perjalanan

disajikan pada Tabel 3.10. Berdasarkan data pada Tabel 3.10 dan metode konstruksi rute menggunakan metode perubahan 2 operator, titik tujuan setelah depot di subrute pertama adalah titik 2.

Tahap selanjutnya adalah melakukan proses yang sama dengan titik asal 2 dan titik tujuan 5 dan 6. Besar jarak tempuh dari titik-titik tersebut tersaji pada Tabel 3.11. sesuai data dan syarat konstruksi rute metode perubahan 2 operator, titik ketiga bagi subrute pertama adalah 6, dan titik tujuan terakhir sebelum kembali ke depot adalah 5. Langkah optimasi tersebut juga akan diterapkan pada subrute selanjutnya.

Tabel 3.10 Jarak tempuh pemilihan jalur pertama

Titik Asal	Titik Tujuan	Jarak Tempuh
1 (Depot)	2	3,6056
1 (Depot)	5	6,4031
1 (Depot)	6	5,0000

Tabel 3.11 Jarak tempuh pemilihan jalur kedua

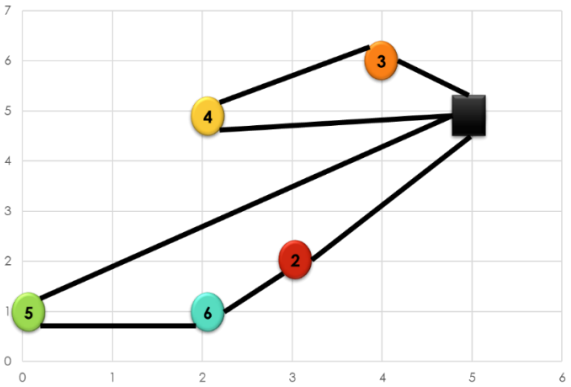
Titik Asal	Titik Tujuan	Jarak Tempuh
2	5	3,1623
2	6	1,4142

Setelah semua tahapan proses optimasi menggunakan metode perubahan 2 operator terlaksana pada semua subrute, maka proses *local search* dinyatakan selesai. Berdasarkan simulasi data uji contoh, subrute yang terbentuk adalah sebagai berikut:

Subrute(1): $1-2-6-5-1$,

Subrute(2): $1-3-4-1$.

Besar jarak tempuh rute yang dihasilkan oleh proses *local search* adalah 20,0732 dan plot rutenya tersaji pada Gambar 3.9. Perbedaan rute hasil mutasi dan hasil *local search* terletak pada urutan titik tujuan yang terdapat pada subrute 2, yaitu 1-4-3-1 untuk mutasi dan 1-3-4-1 untuk *local search*.



Gambar 3.9 Plot rute hasil *local search*

3.3.6 Update pheromone

Karena pada data uji contoh besar jarak tempuh rute hasil operasi mutasi dan *local search* lebih optimal dibanding besar jarak tempuh rute hasil konstruksi menggunakan metode ACO biasa, maka jalur yang akan mengalami *update* nilai *pheromone* adalah jalur pada rute solusi hasil operasi mutasi dan *local search*. Merujuk pada artikel Bin, dkk (2009), formula yang digunakan untuk menghitung penambahan nilai *pheromone* berbeda dengan formula yang digunakan pada ACO biasa seperti tertera pada Persamaan 2.1. Formula yang digunakan adalah sebagai berikut:

$$\Delta\tau^{k}{}_{i,j} = \begin{cases} \left(\frac{K_0}{K \times L}\right)\left(\frac{D^k - d_{i,j}}{m^k \times D^k}\right), & \text{jika jalur } (i,j) \text{ ada pada subrute } k \\ 0, & \text{selainnya,} \end{cases}$$

dengan K_0 adalah konstanta pengali bernilai 1000, K adalah banyak subrute yang terbentuk pada rute solusi yang terpilih sebagai solusi optimal iterasi saat ini, L adalah besar jarak rute yang terpilih sebagai solusi optimal iterasi saat ini, D^k adalah besar jarak subrute ke-k pada solusi, $d_{i,j}$ adalah besar jarak antara titik i dan j, dan m^k adalah banyak titik tujuan (tanpa depot) pada subrute ke-k. Setelah memperoleh nilai $\Delta\tau^{k}{}_{i,j}$, langkah selanjutnya adalah menghitung intensitas *pheromone* semut antar titik untuk siklus selanjutnya dengan formula:

$$\tau_{i,j} = \rho \times \tau_{i,j} + \Delta\tau^{k}{}_{i,j}.$$

Formula tersebut juga digunakan pada metode ACO biasa untuk proses update *pheromone*, menggantikan persamaan 2.1.

Sebagai contoh perhitungan, nilai $\Delta\tau^1{}_{1,2}$ dihitung dengan persamaan:

$$\Delta\tau^1{}_{1,2} = \left(\frac{K_0}{K \times L}\right)\left(\frac{D^1 - d_{1,2}}{m^1 \times D^1}\right)$$

$$\Delta\tau^1{}_{1,2} = \left(\frac{1000}{2 \times 20{,}0732}\right)\left(\frac{13{,}4229 - 3{,}6056}{3 \times 13{,}4229}\right).$$

Dari perhitungan tersebut, diperoleh nilai $\Delta\tau^1{}_{1,2} = 5{,}7120$. Selanjutnya adalah menghitung $\tau_{1,2}$ baru dengan persamaan:

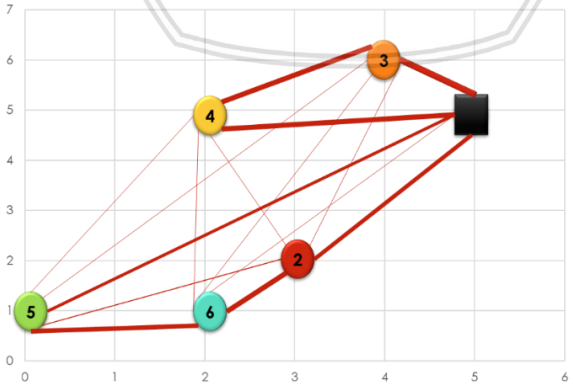
$$\tau_{1,2} = \rho \times \tau_{1,2} + \Delta\tau^1{}_{1,2}$$

$$\tau_{1,2} = 0{,}8 \times 0{,}01 + 5{,}7120,$$

sehingga diperoleh nilai $\tau_{1,2} = 5{,}7200$. Jika proses update pheromone terhadap data uji contoh menggunakan formula tersebut dilakukan sebanyak 100 iterasi, maka matriks nilai intensitas *pheromone* yang terbentuk adalah sebagai berikut:

$$\tau = \begin{bmatrix} 0{,}00 & 30{,}06 & 52{,}37 & 39{,}18 & 23{,}48 & 2.04 \times 10^{-10} \\ 30{,}06 & 0{,}00 & 2.04 \times 10^{-10} & 2.04 \times 10^{-10} & 2.04 \times 10^{-10} & 41{,}85 \\ 52{,}37 & 2.04 \times 10^{-10} & 0{,}00 & 45{,}67 & 2.04 \times 10^{-10} & 2.04 \times 10^{-10} \\ 39{,}18 & 2.04 \times 10^{-10} & 45{,}67 & 0{,}00 & 2.04 \times 10^{-10} & 2.04 \times 10^{-10} \\ 23{,}48 & 2.04 \times 10^{-10} & 2.04 \times 10^{-10} & 2.04 \times 10^{-10} & 0{,}00 & 38{,}10 \\ 2.04 \times 10^{-10} & 41{,}85 & 2.04 \times 10^{-10} & 2.04 \times 10^{-10} & 38{,}10 & 0{,}00 \end{bmatrix}$$

Graf setelah 100 iterasi proses *update pheromone* tersaji pada Gambar 3.10.



Gambar 3.10 Plot intensitas *pheromone* setelah 100 iterasi

3.3.7 Cek kriteria pemberhentian

Langkah selanjutnya adalah cek kriteria pemberhentian. Metode IACO pada skripsi ini menggunakan dua kriteria pemberhentian, yaitu:

1. Ketika jumlah iterasi telah mencapai batas iterasi maksimum (T).
2. Ketika tidak terdapat perubahan nilai solusi optimum global pada 500 iterasi secara beruntun sebelum mencapai iterasi maksimum (T).

Penggunaan keriteria pemberhentian tersebut bertujuan untuk membatasi waktu komputasi yang dibutuhkan.

3.4 Hasil Percobaan dan Analisis

ACO dan IACO diimplementasikan ke dalam bahasa pemrograman MATLAB, lalu dijalankan dengan menggunakan perangkat lunak MATLAB 2017a, pada komputer dengan prosesor AMD A10 3,50 GHz, 8,0 GB RAM. Data yang dijadikan sebagai data uji adalah data uji yang berasal dari penelitian Augerat, dkk (1995) dan Beasley (1990). Kombinasi nilai akan dilakukan pada dua parameter yang digunakan, yaitu:

a. Jumlah semut (m).
b. Konstanta penguapan *pheromone* (ρ).

Tabel 3.12 Jarak Terpendek diketahui

Data Uji	Jumlah Titik Tujuan	Jarak Terpendek Diketahui
Augerat, dkk.	20	211,000
Beasley	50	524,610
Beasley	100	819,560
Beasley	150	1028,420

Empat data yang akan diuji telah memiliki nilai jarak terpendek yang diketahui, sehingga nilai tersebut menjadi acuan dalam pengujian performa metode IACO. Jarak terpendek yang diketahui untuk masing-masing data uji dapat dilihat pada Tabel 3.12.

Percobaan dilakukan menggunakan dua parameter yang dibuat beragam. Parameter-parameter tersebut adalah konstanta penguapan *pheromone* (ρ) dan jumlah semut (m). Pemberian nilai beragam pada dua parameter tersebut dilakukan untuk membandingkan metode ACO biasa dengan IACO dari segi jarak solusi yang diperoleh. Setiap satu kombinasi parameter dilakukan lima kali perulangan.

Setelah melakukan lima kali perulangan pada sembilan kombinasi nilai parameter yang dibuat beragam, proses selanjutnya adalah menghitung Standar Deviasi (SD) dari jarak yang dihasilkan oleh percobaan dalam satu kombinasi nilai parameter. Tujuan perhitungan nilai SD adalah untuk membandingkan tingkat kestabilan nilai solusi yang dihasilkan ACO biasa dan IACO dalam proses memperoleh rata-rata jarak. SD dihitung menggunakan persamaan:

$$SD = \sqrt{\frac{\sum_{i=1}^{n}\left(sol_i - \overline{sol}\right)^2}{(n-1)}},$$

dengan n adalah banyaknya perulangan, $\overline{sol}$ adalah rata-rata jarak, dan sol_i adalah data perulangan ke-i (waktu komputasi atau jarak yang diperoleh). Semakin kecil nilai SD, maka tingkat kestabilan metode optimasi yang digunakan semakin baik. Namun perlu diperhatikan bahwa tingkat kestabilan yang lebih baik belum tentu menghasilkan solusi yang lebih baik pula, karena metode optimasi yang digunakan pada skripsi ini merupakan metode heuristik.

Percobaan pertama dilakukan menggunakan nilai konstanta penguapan *pheromone* (ρ) sebesar 0,8 dan jumlah semut (m) sebanyak 1, 5, dan 10. Percobaan selanjutnya dilakukan menggunakan nilai konstanta penguapan *pheromone* (ρ) yang lebih besar (0,89 dan 0,99) dengan kombinasi jumlah semut (m) yang sama. Percobaan ini dilakukan dengan data uji sekunder yang berasal dari penelitian Augerat, dkk (1995) dengan banyak titik 20, lalu Beasley (1990) dengan banyak titik 50, 100, dan 150, dengan tujuan menguji kemampuan IACO pada ukuran data yang berbeda-beda serta mengetahui dampak dari kombinasi parameter yang dilakukan.

3.4.1 Hasil percobaan dan analisisnya pada data uji 20 titik

Data uji pertama yang digunakan adalah data sekunder yang berasal dari penelitian Augerat, dkk. (1995). Pada data ini, terdapat 20 titik tujuan yang harus dikunjungi, dengan masing-masing titik tujuan memiliki permintaan berkisar [6,30] dan kapasitas angkut kendaraan Q sebesar 160. Detail permintaan dan koordinat setiap titik tersaji pada Lampiran 1.

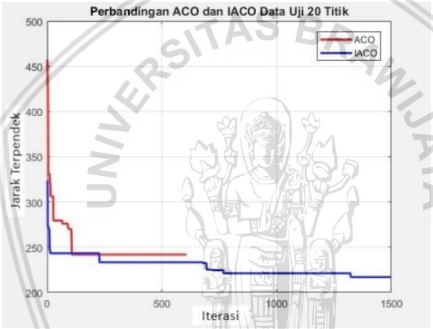
Percobaan pertama untuk data tersebut menggunakan nilai konstanta penguapan *pheromone* (ρ) sebesar 0,8 dan jumlah semut (m) sebesar 1. Intensitas *pheromone* semut antar titik (τ) diberikan nilai awal masing-masing jalur antar titik sebesar 0,01, $\alpha = 2$, $\beta = 1$, dan $K_0 = 1000$. Proses komputasi akan dihentikan apabila jumlah iterasi telah mencapai 1500 atau terdapat 500 iterasi dengan solusi terbaik global bernilai sama secara beruntun. Data hasil seluruh percobaan untuk semua data uji akan disajikan dalam bentuk tabel. Nilai yang diberi warna merah pada masing-masing tabel adalah nilai yang paling optimal diantara metode ACO dan IACO pada sebuah kombinasi parameter yang sama.

Tabel 3.13 Hasil optimasi data uji Augerat, dkk. (20 titik)

No	Kombinasi parameter		Jarak			
			Rata-rata		SD	
	m	ρ	ACO	IACO	ACO	IACO
1	1	0,8	277,011	225,812	**10,888**	5,290
2	1	0,89	282.022	**221,256**	17,738	4,812
3	1	0,99	259,186	228,738	17,371	5,190
4	5	0.8	282,877	227,624	17,054	6,141
5	5	0.89	273,980	229,809	14,647	6,125
6	5	0.99	**257,527**	230,786	14,648	6,853
7	10	0.8	294,194	227,992	32,157	**2,260**
8	10	0.89	291,532	229,376	32,853	5,918
9	10	0.99	258,528	231,202	11,216	5,881

Terlihat pada Tabel 3.13, rata-rata jarak yang bernilai paling kecil diperoleh menggunakan metode IACO, yaitu 221,256. Kombinasi parameter yang digunakan ketika metode IACO memperoleh nilai rata-rata jarak tersebut adalah 1 untuk m dan 0,89

untuk ρ. Jarak terpendek yang diperoleh pada percobaan tersebut adalah 216,882, yaitu oleh metode IACO. Dari seluruh percobaan yang dilakukan, metode IACO berhasil memperoleh rata-rata jarak yang lebih kecil dibandingkan dengan metode ACO. Rata-rata jarak paling kecil yang dihasilkan metode ACO adalah 257,527, dengan jarak terpendek 241,814. Grafik perbandingan jarak terpendek yang dihasilkan metode ACO dan IACO dapat dilihat pada Gambar 3.11. Plot rute terpendek percobaan ini tersaji pada Lampiran 5 dan untuk rute terpendek yang telah diketahui tersaji pada Lampiran 6. Pada percobaan menggunakan data uji 20 titik kali ini, metode IACO menghasilkan nilai solusi yang bernilai lebih stabil dibandingkan dengan ACO, karena menghasilkan nilai SD yang lebih kecil untuk semua percobaan yang dilakukan.



Gambar 3.11 Perbandingan jarak terpendek data uji 20 titik

Perbandingan waktu komputasi antara metode ACO dan IACO untuk data uji 20 titik dapat dilihat pada Tabel 3.14. Pada tabel tersebut terlihat bahwa metode ACO berhasil memperoleh waktu komputasi lebih singkat dibandingkan dengan metode IACO untuk semua percobaan yang dilakukan. Waktu komputasi IACO lebih lama dibanding ACO karena IACO memerlukan waktu komputasi tambahan untuk melakukan operasi mutasi dan proses *local search*, sehingga hipotesa yang dapat diambil melalui percobaan pada data uji 20 titik ini adalah perbaikan yang dilakukan pada ACO biasa dengan menambahkan dua metode tersebut membuat waktu komputasi lebih lama sebagai konsekuensi untuk mendapatkan rute solusi dengan jarak yang lebih pendek. Perbandingan performa ACO dan IACO secara ringkas dapat dilihat pada Tabel 3.15. Selain hal tersebut, rata-rata

iterasi yang dilakukan ACO juga bernilai lebih kecil dibanding IACO, yaitu 711 berbanding 823.

Tabel 3.14 Rata-rata waktu komputasi data uji Augerat, dkk. (20 titik)

No	Kombinasi parameter		Waktu komputasi (s)			
			Rata-rata		SD	
	m	ρ	ACO	IACO	ACO	IACO
1	1	0,8	1,295	2,379	0,296	0,476
2	1	0,89	**1,042**	2,373	**0,067**	0,565
3	1	0,99	1,256	**1,665**	0,108	**0,100**
4	5	0.8	4,032	5,004	0,669	0,687
5	5	0.89	3,405	5,081	0,608	1,098
6	5	0.99	4,133	4,441	1,349	1,167
7	10	0.8	8,178	9,693	4,065	2,651
8	10	0.89	5,520	8,478	0,905	1,734
9	10	0.99	6,622	7,484	1,778	1,351

Tabel 3.15 Perbandingan solusi diperoleh dengan solusi diketahui data Augerat, dkk. (20 titik)

Metode	Rata-Rata Jarak Terbaik	Keunggulan Waktu Komputasi	Jarak Terpendek Percobaan	Jarak Terpendek Diketahui
ACO	257,527	9 percobaan	241,814	211,000
IACO	221.256	0 percobaan	216,882	

Jarak terpendek dari percobaan yang dilakukan pada data uji Augerat, dkk. (20 titik) ini belum mampu mencapai jarak terpendek yang diketahui. Hal tersebut disebabkan karena masih terdapat jalur yang bersilangan pada rute solusi yang terbentuk. Plot rute solusi dengan jarak terpendek ACO dan IACO pada percobaan kali ini disajikan pada Lampiran 5 dan Lampiran 6, serta rute solusi terpendek yang diketahui disajikan pada Lampiran 7. Terlihat pada Lampiran 5 dan Lampiran 6 bahwa jalur yang bersilangan pada rute solusi ACO berjumlah lebih banyak dibanding rute solusi IACO, yaitu tiga

berbanding satu. Hal tersebut adalah penyebab jarak rute solusi IACO lebih pendek dibanding milik ACO sesuai dengan fungsi metode *local search* pada IACO.

3.4.2 Hasil percobaan dan analisisnya pada data uji 50 titik

Data uji kedua yang digunakan adalah data sekunder yang berasal dari penelitian Beasley (1990), yaitu data yang memiliki 50 titik tujuan yang harus dikunjungi. Masing-masing titik tujuan tersebut memiliki permintaan tiap titik yang beragam, yaitu berkisar [3,41]. Besar kapasitas kendaraan (Q) yang digunakan bernilai 160. Koordinat dan permintaan tiap titik disajikan pada Lampiran 2.

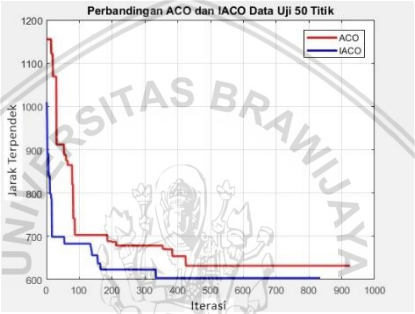
Kombinasi parameter yang digunakan untuk percobaan kali ini bernilai sama dengan kombinasi parameter pada percobaan dengan data uji sebelumnya (20 titik). Kriteria pemberhentian yang digunakan juga serupa, yaitu apabila jumlah iterasi telah mencapai 1500 atau terdapat 500 iterasi dengan solusi terbaik global bernilai sama secara beruntun sebelum mencapai 1500 iterasi.

Tabel 3.16 Hasil optimasi data uji Beasley (50 titik)

No	Kombinasi parameter		Jarak			
			Rata-rata		SD	
	m	ρ	ACO	IACO	ACO	IACO
1	1	0,8	792,489	642,491	21,339	16,735
2	1	0,89	754,220	632,797	44,778	20,717
3	1	0,99	660,690	635,039	32,459	11,014
4	5	0.8	821,898	643,189	30,059	**4,946**
5	5	0.89	761,198	644,338	45,443	26,934
6	5	0.99	656,558	638,448	22,723	10,652
7	10	0.8	798,284	**627,517**	20,096	17,782
8	10	0.89	747,242	679,323	44,138	19,220
9	10	0.99	**652,528**	642,625	**5,999**	19,747

Rata-rata jarak yang dihasilkan melalui penggunaan metode ACO biasa dan IACO dapat dilihat pada Tabel 3.16. Tabel tersebut menunjukkan bahwa metode IACO selalu memperoleh rata-rata jarak lebih kecil dibanding ACO. Rata-rata jarak yang memiliki nilai paling kecil diperoleh oleh percobaan menggunakan metode IACO dengan

nilai 627,517 dan jarak terpendek bernilai 603,345. Nilai tersebut lebih kecil dibanding rata-rata jarak yang bernilai paling kecil dari solusi milik ACO, yaitu 652,528 dengan jarak terpendeknya bernilai 631,617. Perbandingan proses ACO dan IACO dalam mencari jarak terpendek dapat dilihat pada Gambar 3.12. Plot rute solusi dengan jarak terpendek ACO dan IACO pada percobaan kali ini disajikan pada Lampiran 8 dan Lampiran 9, sedangkan rute solusi terpendek yang diketahui akan disajikan pada Lampiran 10. Lampiran 8 dan Lampiran 9 memberi informasi bahwa jalur yang bersilangan pada rute solusi ACO berjumlah lebih banyak dibanding milik rute solusi IACO, yaitu sembilan berbanding empat.



Gambar 3.12 Perbandingan jarak terpendek data uji 50 titik

Tabel 3.17 Rata-rata waktu komputasi data uji Beasley (50 titik)

No	Kombinasi parameter		Waktu komputasi (s)			
			Rata-rata		SD	
	m	ρ	ACO	IACO	ACO	IACO
1	1	0,8	**5,419**	8,857	1,552	2,715
2	1	0,89	5,552	8,510	2,112	3,363
3	1	0,99	5,765	**6,988**	**0,949**	**1,211**
4	5	0.8	28,627	29,294	8,205	8,806
5	5	0.89	20,633	24,184	3,845	8,969
6	5	0.99	24,793	23,673	2,065	6,953
7	10	0.8	48,691	53,919	23,881	16,021
8	10	0.89	33,450	45,177	3,564	8,139
9	10	0.99	48,226	46,326	7,487	9,144

Percobaan pada data uji 50 titik ini mencapai rata-rata jarak paling kecil menggunakan kombinasi parameter $m = 10$ dan $\rho = 0{,}8$. Kondisi tersebut memberi informasi bahwa kombinasi nilai parameter yang digunakan pada percobaan dengan data uji 50 titik berbeda dengan kombinasi parameter yang digunakan saat percobaan dengan data uji 20 titik, yaitu $m = 1$ dan $\rho = 0{,}89$. Keterangan tersebut menunjukkan bahwa performa terbaik metode optimasi yang digunakan pada skripsi ini tidak bergantung pada satu jenis kombinasi nilai parameter saja. Dapat dilihat pula nilai SD rata-rata jarak metode IACO mengalami peningkatan dibanding percobaan pada data uji sebelumnya. Hal tersebut mengindikasikan bahwa rentang solusi yang mampu dicapai oleh metode IACO pada data uji 50 titik bernilai lebih besar dibanding ketika menyelesaikan data uji 20 titik.

Tabel 3.18 Perbandingan solusi diperoleh dengan solusi diketahui data Beasley (50 titik)

Metode	Rata-Rata Jarak Terbaik	Keunggulan Waktu Komputasi	Jarak Terpendek Percobaan	Jarak Terpendek Diketahui
ACO	652,528	6 percobaan	631,617	524.610
IACO	627,517	3 percobaan	603,345	

Ditinjau dari Tabel 3.17, terlihat bahwa metode IACO menghasilkan tiga rata-rata waktu komputasi yang bernilai lebih kecil dibanding ACO pada 9 percobaan yang dilakukan. Penyebab hal tersebut dapat terjadi adalah karena selisih rata-rata iterasi yang diperlukan bernilai lebih kecil dibandingkan selisih rata-rata iterasi pada percobaan yang dilakukan dengan data uji sebelumnya. Selisih antara rata-rata iterasi yang diperlukan ACO dan IACO pada percobaan kali ini bernilai 51 (823 berbanding 874), sedangkan pada data uji 20 titik bernilai 112. Kondisi tersebut tidak lepas dari peran operasi mutasi dan proses *local search* yang berfungsi membangun rute solusi baru dengan jangkaun nilai solusi yang lebih luas dengan diiringi usaha menjaga rute solusi tersebut tetap optimal pada masing-masing subrutenya. Hal ini membuat IACO cenderung tidak memerlukan banyak iterasi untuk mencapai nilai optimal globalnya. Perbandingan secara ringkas hasil percobaan yang diperoleh metode

ACO dan IACO pada percoban ini dengan jarak terpendek yang saat ini diketahui tersaji pada Tabel 3.18.

Serupa dengan percobaan sebelumnya, nilai jarak terpendek percobaan yang dilakukan pada data uji Beasley (50 titik) ini belum mampu menjangkau nilai jarak terpendek yang diketahui. Hal tersebut disebabkan karena hal yang sama pula, yaitu masih terdapat jalur yang bersilangan pada rute solusi yang terbentuk seperti terlihat pada Lampiran 9.

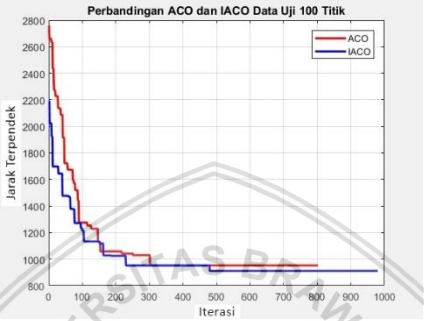
3.4.3 Hasil percobaan dan analisisnya pada data uji 100 titik

Data uji ketiga yang digunakan adalah data sekunder yang berasal dari penelitian Beasley (1990) dengan titik tujuan berjumlah 100. Masing-masing titik tujuan tersebut memiliki permintaan tiap titik yang beragam. Ragam permintaan masing-masing titik tujuan berkisar $[10,50]$. Sesuai data uji, kapasitas kendaraan (Q) yang digunakan adalah sebesar 200. Koordinat titik dan permintaan tiap titik disajikan pada lampiran 3.

Kombinasi parameter yang digunakan pada percobaan ini bernilai sama dengan kombinasi parameter pada dua percobaan sebelumnya. Kriteria pemberhentian yang diinisialisasi juga serupa, yaitu proses komputasi akan berhenti jika telah mencapai 1500 iterasi atau terdapat 500 iterasi dengan solusi terbaik global bernilai sama secara beruntun sebelum mencapai 1500 iterasi.

Ditinjau dari hasil percobaan yang tertera pada Tabel 3.19, metode IACO kembali berhasil memperoleh rata-rata jarak lebih baik dari metode ACO untuk semua kombinasi parameter. IACO berhasil memperoleh rata-rata jarak sebesar 964,560 dengan jarak terpendek 911,389, sedangkan ACO memperoleh rata-rata jarak sebesar 987,009 dengan jarak terpendek 952,832. Hal tersebut semakin menegaskan bahwa metode IACO memiliki performa yang lebih baik dibanding metode ACO dalam memperoleh jarak terpendek dari suatu VRP, termasuk jika ditinjau dari performa pada dua percobaan data uji sebelumnya. Perbandingan proses kedua metode tersebut dalam memperoleh jarak terpendek disajikan pada Gambar 3.13. Untuk plot rute solusi terpendek pada pepercobaan ini dan rute solusi terpendek ACO, IACO, dan yang diketahui disajikan pada Lampiran 11, Lampiran 12, dan Lampiran 13. Terlihat pula pada Lampiran 11 dan Lampiran 12 bahwa jalur yang bersilangan pada rute solusi ACO

berjumlah lebih banyak dibanding rute solusi IACO, yaitu delapan berbanding lima. Selain rata-rata jarak, terlihat juga pada Tabel 3.19 bahwa nilai SD metode IACO kembali mengalami peningkatan nilai. Kondisi tersebut mengindikasikan bahwa IACO semakin efektif dalam hal memperluas rentang solusi pada saat VRP yang diselesaikan memiliki jumlah titik tujuan yang semakin besar.



Gambar 3.13 Perbandingan jarak terpendek data uji 100 titik

Tabel 3.19 Hasil optimasi data uji Beasley (100 titik)

No	Kombinasi parameter		Jarak			
			Rata-rata		SD	
	m	ρ	ACO	IACO	ACO	IACO
1	1	0,8	1.161,183	1.036,562	49,332	22,538
2	1	0,89	1.078,017	1.011,650	24,826	**8,411**
3	1	0,99	988,037	987,879	10,325	28,247
4	5	0.8	1.231,261	1.027,381	45,131	20,413
5	5	0.89	1.096,005	1.007,465	33,056	9,935
6	5	0.99	1.009,150	**964,560**	**8,734**	37,928
7	10	0.8	1.204,379	1.016,742	43,033	21,053
8	10	0.89	1.086,975	1.013,212	26,302	9,971
9	10	0.99	**987,009**	981,133	21,846	23,941

Selanjutnya adalah membandingkan rata-rata waktu komputasi. Data waktu komputasi hasil percobaan dapat dilihat pada Tabel 3.20. Percobaan pada data uji sebelumnya telah memperoleh informasi yaitu terjadi peningkatan jumlah keunggulan metode IACO dari segi waktu

komputasi (0 menjadi 3). Pada percobaan kali ini kembali terjadi peningkatan jumlah keunggulan rata-rata waktu komputasi IACO dibanding ACO, yaitu menjadi 4 percobaan. Kondisi tersebut disebabkan oleh variabel yang sama, yaitu perbandingan rata-rata iterasi yang terjadi. Pada percobaan ini, rata-rata iterasi metode IACO bernilai lebih kecil dibandingkan dengan rata-rata iterasi metode ACO, yaitu 960 (ACO) dan 893 (IACO) dengan selisih sebesar 67 iterasi.

Tabel 3.20 Rata-rata waktu komputasi data uji Beasley (100 titik)

No	Kombinasi parameter		Waktu komputasi (s)			
			Rata-rata		SD	
	m	ρ	ACO	IACO	ACO	IACO
1	1	0,8	**28,615**	34,473	**4,487**	8,253
2	1	0,89	29,118	**25,712**	9,564	8,548
3	1	0,99	31,499	27,276	5,130	**4,153**
4	5	0.8	102,472	119,867	36,080	35,119
5	5	0.89	109,394	102,250	26,431	38,213
6	5	0.99	134,903	142,897	31,357	38,191
7	10	0.8	194,039	199,063	59,065	60,684
8	10	0.89	252,734	214,959	76,500	78,878
9	10	0.99	216,043	216,590	17,325	27,346

Tabel 3.21 Perbandingan solusi diperoleh dengan solusi diketahui data Beasley (100 titik)

Metode	Rata-Rata Jarak Terbaik	Keunggulan Waktu Komputasi	Jarak Terpendek Percobaan	Jarak Terpendek Diketahui
ACO	987,009	5 percobaan	952,832	819,560
IACO	964,560	4 percobaan	911,389	

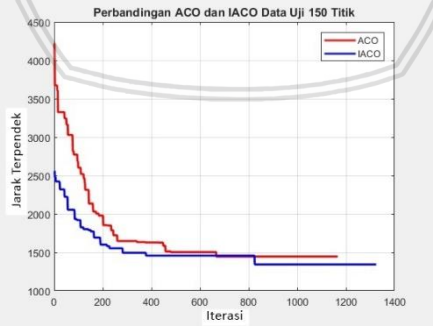
Sebagai tambahan informasi, Tabel 3.21 menyajikan perbandingan performa ACO dan IACO pada percobaan ini secara ringkas. Hal yang sama dalam perbandingan jarak terpendek yang

diperoleh dan jarak terpendek yang diketahui juga terjadi pada percobaan kali ini, yaitu nilai hasil optimasi yang diperoleh untuk data uji Beasley (100 titik) ini belum mampu mencapai nilai jarak terpendek yang diketahui. Hal tersebut disebabkan karena masih terdapat jalur yang bersilangan pada rute solusi yang terbentuk seperti terlihat pada Lampiran 12.

3.4.4 Hasil percobaan dan analisisnya pada data uji 150 titik

Data uji terakhir pada skripsi ini adalah data uji dengan 150 titik tujuan yang berasal dari penelitian Beasley (1990). Setiap titik tujuan telah memiliki koordinat dan jumlah permintaan dengan nilai berkisar [3,41]. Koordinat dan jumlah permintaan tersebut terlampir pada Lampiran 4. Perlu diketahui pula bahwa IACO mengalami peningkatan jumlah keunggulan rata-rata waktu komputasi seiring dengan bertambahnya jumlah titik tujuan pada data uji 20, 50, dan 100 titik. Hal tersebut merupakan hipotesis dari percobaan yang dilakukan sebelumnya, sehingga penggunaan data uji 150 titik ini digunakan dengan tujuan untuk menguatkan hipotesis tersebut.

Kombinasi parameter yang digunakan pada percobaan ini sama dengan kombinasi parameter pada percobaan tiga data uji sebelumnya. Kriteria pemberhentian yang diinisialisasi juga serupa, yaitu proses komputasi akan berhenti jika telah mencapai 1500 iterasi atau terdapat 500 iterasi dengan solusi terbaik global bernilai sama secara beruntun sebelum mencapai 1500 iterasi.



Gambar 3.14 Perbandingan jarak terpendek data uji 150 titik

Rata-rata jarak yang diperoleh dari metode IACO berhasil mengungguli rata-rata jarak yang diperoleh metode ACO pada semua

percobaan. Informasi tersebut terdapat pada Tabel 3.22.. Jarak terpendek yang diperoleh metode IACO adalah 1.342,901 dengan rata-rata jarak paling kecil sebesar 1.408,403. Nilai tersebut lebih kecil dibandingkan dengan jarak terpendek yang diperoleh metode ACO, 1.443,774 dengan rata-rata jarak paling kecil sebesar 1.494,974. Gambar 3.14 menyajikan perbandingan proses ACO dan IACO dalam memperoleh jarak terpendek. Hasil tersebut semakin menegaskan bahwa metode IACO memiliki performa yang lebih baik dalam pencarian rute solusi VRP dengan jarak terpendek dibanding metode ACO. Plot rute solusi dengan jarak terpendek ACO, IACO, dan yang telah diketahui tersaji pada Lampiran 14, Lampiran 15, dan Lampiran 16. Jika diitinjau dari rute solusi ACO dan IACO yang tersaji pada Lampiran 14 dan Lampiran 15, maka akan diperoleh informasi bahwa jumlah jalur yang bersilangan pada rute solusi ACO adalah dua belas, sedangkan pada rute solusi IACO berjumlah sembilan. Berdasarkan informasi tersebut dan informasi serupa dari percobaan pada data uji sebelumnya, maka dapat disimpulkan bahwa metode *local search* berhasil mereduksi jumlah jalur yang bersilangan pada seluruh data uji VRP yang digunakan pada skripsi ini.

Tabel 3.22 Hasil optimasi data uji Beasley (150 titik)

No	Kombinasi parameter		Jarak			
			Rata-rata		SD	
	m	ρ	ACO	IACO	ACO	IACO
1	1	0,8	2.096,365	1.491,835	65,822	26,764
2	1	0,89	1.716,986	1.446,901	71,845	22,068
3	1	0,99	**1.494,974**	**1.408,403**	34,656	41,564
4	5	0.8	1.992,750	1.484,979	80,874	28,279
5	5	0.89	1.736,150	1.441,696	32,531	34,903
6	5	0.99	1.497,460	1430,736	23,143	**16,567**
7	10	0.8	2.061,006	1.480,432	87,017	37,459
8	10	0.89	1.789,816	1.441,664	52,469	30,149
9	10	0.99	1.504,608	1.418,159	**13,217**	29,780

Jarak terpendek pada percobaan ini dibandingkan dengan jarak terpendek yang telah diketahui. Dari perlakuan tersebut diperoleh informasi bahwa jarak terpendek percobaan ini belum mencapai jarak

terpendek yang telah diketahui. Penyebab hal tersebut dapat terjadi adalah serupa dengan percobaan pada data uji yang lain, yaitu masih terdapat jalur yang bersilangan pada satu subrute.

Analisis selanjutnya adalah membandingkan rata-rata waktu komputasi ACO dan IACO. Data waktu komputasi terdapat pada Tabel 3.23. Dari data pada tabel tersebut dapat disimpulkan bahwa IACO cenderung mampu melakukan proses optimasi lebih cepat dibandingkan dengan ACO pada data uji yang memiliki titik tujuan berjumlah banyak. Semakin banyak titik tujuan pada suatu VRP, maka metode IACO akan semakin unggul baik dalam hal memperoleh jarak terpendek, maupun waktu komputasi yang dibutuhkan.

Tabel 3.23 Rata-rata waktu komputasi data uji Beasley (150 titik)

No	Kombinasi parameter		Waktu komputasi (s)			
			Rata-rata		SD	
	m	ρ	ACO	IACO	ACO	IACO
1	1	0,8	**78,140**	74,620	18,110	**16,354**
2	1	0,89	86,551	**73,479**	18,019	26,966
3	1	0,99	88,159	79,235	**15,554**	18,517
4	5	0.8	365,947	299,176	101,054	83,665
5	5	0.89	417,879	334,014	47,141	126,435
6	5	0.99	452,227	313,126	61,542	87,076
7	10	0.8	570,271	719,028	195,449	235,807
8	10	0.89	698,615	641,242	182,157	221,682
9	10	0.99	689,246	761,168	140,560	197,506

Tabel 3.24 menyajikan data rata-rata iterasi yang dilakukan ACO dan IACO. Terlihat pada tabel tersebut, metode IACO mengalami peningkatan rata-rata iterasi lebih sedikit dibanding ACO seiring bertambahnya jumlah titik tujuan dari data uji yang diselesaikan. Hal ini mengindikasikan bahwa jika jumlah titik tujuan dari VRP semakin banyak, maka IACO mampu mencapai konvergensi lebih cepat dalam proses optimasinya, namun tetap menuju nilai optimal global. Hal tersebut menyebabkan rute solusi yang dihasilkan IACO tetap memiliki jarak lebih pendek dibandingkan dengan rute solusi ACO meskipun IACO melakukan iterasi dengan jumlah yang

lebih sedikit dibanding ACO biasa. Hasil percobaan disajikan secara ringkas pada Tabel 3.25.

Tabel 3.24 Rata-rata iterasi yang terjadi

Metode	Data Uji			
	20 titik	50 titik	100 titik	150 titik
ACO	711	823	960	1154
IACO	823	874	893	1065

Tabel 3.25 Perbandingan solusi diperoleh dengan solusi diketahui data Beasley (150 titik)

Metode	Rata-Rata Jarak Terbaik	Keunggulan Waktu Komputasi	Jarak Terpendek Percobaan	Jarak Terpendek Diketahui
ACO	1.494,974	2 percobaan	1.408,403	1028,420
IACO	1.408,403	7 percobaan	1.342,901	

3.4.5 Analisis dampak kombinasi parameter

Tabel 3.26 memperlihatkan kombinasi parameter yang menghasilkan rata-rata jarak paling kecil pada masing-masing metode optimasi. Metode ACO memperoleh rata-rata jarak paling kecil ketika menggunakan kombinasi parameter m dengan nilai 1, 5, dan 10. Untuk nilai ρ yang digunakan konsisten bernilai 0,99.

Tabel 3.26 Kombinasi parameter yang memperoleh rata-rata jarak terbaik

Metode	Data Uji			
	20 titik	50 titik	100 titik	150 titik
ACO	$m = 5$ $\rho = 0{,}99$	$m = 10$ $\rho = 0{,}99$	$m = 5$ $\rho = 0{,}99$	$m = 1$ $\rho = 0{,}99$
IACO	$m = 1$ $\rho = 0{,}89$	$m = 10$ $\rho = 0{,}8$	$m = 10$ $\rho = 0{,}89$	$m = 1$ $\rho = 0{,}99$

Berbeda dengan kondisi yang dialami metode ACO, metode IACO memperoleh rata-rata jarak paling kecil ketika ρ bernilai 0,8, 0,89, dan 0,99. Metode IACO memperoleh rata-rata terbaiknya melalui penggunaan m dengan nilai 1 dan 10. Hal ini menunjukkan bahwa percobaan menggunakan metode ACO dan IACO untuk penyelesaian VRP pada skripsi ini cenderung memerlukan nilai parameter yang beragam supaya kedua metode tersebut mampu mengeluarkan performa yang dinamis.

BAB IV
PENUTUP

4.1 Kesimpulan

Berdasarkan percobaan serta pembahasan yang telah dilakukan, dapat diambil kesimpulan sebagai berikut:

1. IACO dilakukan dengan menambahkan metode operasi mutasi dan proses *local search* kepada ACO untuk membuat rentang solusi yang dapat dijangkau semakin lebar dan membuat jarak pada setiap subrute yang terbentuk menjadi lebih optimal.
2. Perbaikan ACO dengan menambahkan metode operasi mutasi dan proses *local search* (IACO) terbukti mampu menghasilkan rute solusi dengan jarak yang lebih pendek dibanding ACO. Dalam hal waktu komputasi, IACO cenderung lebih cepat dibanding ACO seiring dengan bertambahnya jumlah titik tujuan pada VRP yang akan diselesaikan.
3. Dampak kombinasi parameter yang diterapkan, yaitu nilai konstanta penguapan *pheromone* (ρ) dan jumlah semut (m), memperlihatkan bahwa ACO dan IACO tidak bergantung hanya pada satu kombinasi parameter untuk mencapai solusi terbaik masing-masing metode.

4.2 Saran

Percobaan yang dilakukan pada skripsi ini belum mampu memperoleh nilai solusi terbaik yang diketahui. Salah satu penyebabnya terlihat pada plot rute solusi, yaitu masih terdapat jalur yang bersilangan dalam satu subrute, sehingga untuk penelitian selanjutnya disarankan menggunakan metode *local search* yang lebih efektif. Selain itu juga disarankan melakukan kombinasi pada parameter lain seperti konstanta pengendali intensitas *pheromone* semut (α) dan konstanta pengendali visibilitas (β).

DAFTAR PUSTAKA

Augerat, P., J. Belenguer, E. Benavent, A. Corberán, D. Naddef, dan G. Rinaldi. 1995. *Computational Results with a Branch and Cut Code for the Capacitated Vehicle Routing Problem*. France: Universit´e Joseph Fourier.

Beasley, J.E. 1990. *OR-Library: distributing test problems by electronic mail*. *Journal of the Operational Research Society*. 41: 1069–1072.

Bin, Y., Y. Z. Zhen, dan Y. Baozhen. 2009. *An Improved Ant Colony Optimization for Vehicle Routing Problem*. *European Journal of Operational Research*, 196: 171-176.

Bullnheimer, B., R. F. Hartl, dan C. Strauss. 1997. *Applying The Ant System to The Vehicle Routing Problem*. *Second Metaheuristics International Conference*. pp: 109-120.

Chartrand, G. dan Oellerman, O.R. 1993. *Applied and Algorithmic Graph Theory*. *McGraw-Hill*. New York.

Danielsson, P. 1980. *Computer Graphics and Image Processing*. Linkoping: Linkoping University.

Dorigo, M., V. Maniezzo, dan A. Colorni. 1996. *The Ant System: Optimization by a Colony of Cooperating Agents*. *IEEE Transactions on System, Man, Cybernetics Part B*. 26: 29-41.

Foulds, L.R. 1984. *Combinatorial Optimization for Undergraduates*. *Springer-Verlag*. New York.

Gaertner, D. 2004. *Natural Algorithms in Timetabling and Scheduling*. Edinburgh: University of Edinburgh.

Gen, M. dan Cheng, R. 1997. *Genetic Algorithms and Engineering Design*. New York: John Wiley & Sons.

Goss, S., S. Aron, J. L. Deneubourg, dan J. M. Pasteels. 1989. *Self-organised shortcuts in the Argentine Ant*. Bruxelles: Université Libre de Bruxelles.

Konig, F. 2008. *Scheduling in Water Business*. Thesis. Karlsruhe: Universitat Karlsruhe.

Obitko, M. 1998. *Introduction to Genetic Algorithms*. Prague: Czech Technical University.

Stützle, T.G. 1998. *Local Search Algorithms for Combinatorial Problems.* Dissertation. Darmstadt: Technische Universität Darmstadt.

Toth, P. dan Vigo D. 2002. *The Vehicle Routing Problem*. *Siam Publisher*. Philadelphia.

Tarigan, D. 2008. *Pemodelan Vehicle Routing Problem Terbuka dengan Keterbatasan Waktu*. Thesis. Medan: Sekolah Pascasarjana Universitas Sumatera Utara.

Yu, B., Yang, Z.Z., dan Cheng, C.T. 2007. *Optimizing the distribution of shopping centers with parallel genetic algorithm*. *Engineering Applications of Artificial Intelligence*. 20(2): 215-223.