TESIS – SM 142501

PENYELESAIAN *MULTI-DEPOT MULTIPLE TRAVELING SALESMAN PROBLEM* MENGGUNAKAN *HYBRID FIREFLY ALGORITHM—ANT COLONY OPTIMIZATION*

OLIEF ILMANDIRA RATU FARISI
NRP 1213 201 010

DOSEN PEMBIMBING
Dr. Budi Setiyono, S.Si., M.T.
Dr. Ir. R. Imbang Danandjojo, M.T.

PROGRAM MAGISTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2015

THESIS – SM 142501

SOLVING MULTI-DEPOT MULTIPLE TRAVELING SALESMAN PROBLEM USING HYBRID FIREFLY ALGORITHM—ANT COLONY OPTIMIZATION

OLIEF ILMANDIRA RATU FARISI
NRP 1213 201 010

SUPERVISORS
Dr. Budi Setiyono, S.Si., M.T.
Dr. Ir. R. Imbang Danandjojo, M.T.

MAGISTER PROGRAM
DEPARTMENT OF MATHEMATICS
FACULTY OF MATHEMATICS AND SCIENCES
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2015

PENYELESAIAN *MULTI-DEPOT MULTIPLE TRAVELING SALESMAN PROBLEM* MENGGUNAKAN *HYBRID FIREFLY ALGORITHM–ANT COLONY OPTIMIZATION*

Tesis ini disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Sains (M.Si)
di
Institut Teknologi Sepuluh Nopember

Oleh:

OLIEF ILMANDIRA RATU FARISI
NRP. 1213 201 010

Tanggal Ujian : 3 Juli 2015
Periode Wisuda : September 2015

Disetujui oleh:

Dr. Budi Setiyono, S.Si., M.T. (Pembimbing I)
NIP 19720202 199702 1 001

Dr. Ir. R. Imbang Danoedjoe, M.T. (Pembimbing II)
NIP 19630625 199303 1 001

Dr. Imam Mukhlash, S.Si., M.T. (Penguji)
NIP 19700831 199403 1 003

Dr. Dam[illegible], S.Si., M.T. (Penguji)
NIP 19691013 199412 1 001

Prof. Dr. Mohammad Isa Irawan, M.T. (Penguji)
NIP 19631225 198903 1 001

Direktur Program Pascasarjana

Prof. Dr. Ir. Adi Soeprijanto, M.T.
NIP 19640405 199002 1 001

PENYELESAIAN *MULTI-DEPOT MULTIPLE TRAVELING SALESMAN PROBLEM* MENGGUNAKAN *HYBRID FIREFLY ALGORITHM – ANT COLONY OPTIMIZATION*

Nama Mahasiswa	: Olief Ilmandira Ratu Farisi
NRP	: 1213 201 010
Pembimbing	: Dr. Budi Setiyono, S.Si., M.T.
Ko-pembimbing	: Dr. Ir. R. Imbang Danandjojo, M.T.

ABSTRAK

Secara umum, *Multi-Depot Multiple Traveling Salesman Problem* (MmTSP) digambarkan sebagai masalah pencarian rute terpendek dari beberapa orang *salesman* yang berangkat dari beberapa kota berbeda, yang disebut dengan depot, dan harus kembali ke depot masing-masing sedemikian hingga setiap kota dikunjungi oleh tepat satu *salesman*. Nilai optimum didapat dengan mencari semua kemungkinan rute. Sehingga, semakin banyak kota, semakin lama waktu yang diperlukan untuk menyelesaikan permasalahan tersebut. Penelitian ini mengusulkan metode pendekatan *hybrid Firefly Agorithm* (FA) – *Ant Colony Optimization* (ACO) untuk menyelesaikan MmTSP. Sebagai studi kasus, digunakan rute angkutan laut.

Pada penelitian ini, algoritma FA dijalankan terlebih dahulu untuk mencari solusi lokal karena kemampuan kekonvergenan FA yang cepat. Solusi lokal dari FA kemudian dijadikan inisialisasi untuk pencarian solusi global menggunakan ACO. Kemudian, dilakukan uji coba parameter agar metode *hybrid* FA-ACO dapat menghasilkan solusi yang optimal dengan banyak kunang-kunang dan semut seminimum mungkin. Selanjutnya, metode *hybrid* FA-ACO dibandingkan dengan metode ACO dalam menyelesaikan studi kasus.

Hasil uji coba menunjukkan parameter terbaik untuk menyelesaikan studi kasus dengan metode *hybrid* FA-ACO adalah $\gamma = 0.01$, kunang-kunang sebanyak 4 dan 5 pergerakan untuk tiap kunang-kunang,100 iterasi FA, $\alpha = 1, \beta = 5, \rho = 0{,}5, Q = 100$, 20 semut, dan 229 iterasi ACO. Metode *hybrid* FA-ACO menghasilkan hasil terbaik 13828 km dengan solusi rata-rata 14400,28 km dan rata-rata waktu komputasi 36,01579 detik. Sedangkan metode ACO menghasilkan hasil terbaik 13882 km dengan solusi rata-rata 14463,24 dan rata-rata waktu komputasi 60,77814 detik. Dalam mencapai solusi terbaiknya, metode hybrid FA-ACO telah konvergen pada 16,86102 detik sementara metode ACO konvergen pada 25,07218 detik. Hasil perbandingan menunjukkan bahwa metode hybrid FA-ACO menghasilkan solusi yang lebih baik dengan rata-rata waktu komputasi 40,74% dan waktu konvergensi 32,75% lebih cepat dibandingkan metode ACO.

Kata kunci: *Multi-Depot Multiple Traveling Salesman Problem, Firefly Algorithm, Ant Colony Optimization*, metode *hybrid*.

SOLVING MULTI-DEPOT MULTIPLE TRAVELING SALESMAN PROBLEM USING HYBRID FIREFLY ALGORITHM – ANT COLONY OPTIMIZATION

By : Olief Ilmandira Ratu Farisi
Student Identity Number : 1213 201 010
Supervisor : Dr. Budi Setiyono, S.Si., M.T.
Co-supervisor : Dr. Ir. R. Imbang Danandjojo, M.T.

ABSTRACT

In general, Multi-Depot Multiple Traveling Salesman Problem (MmTSP) described as finding the shortest route by more than one salesman depart from several starting city, called depot, and having returned to the depot so that each city is visited by exactly one salesman. The optimum value is obtained by finding all possible routes. It implies the greater number of cities, the longer time required to solve the problem. This research proposed a heuristic method, the hybrid Firefly Algorithm (FA) – Ant Colony Optimization (ACO) to solve MmTSP. The tour of sea transport is used as case study.

In this study, the FA is implemented to find a local solution because the FA has fast convergence capability. Local solutions of the FA is then used to initialize the pheromone for the global solution search using the ACO. Furthermore, parameter experiment is required so that the hybrid FA-ACO method can obtain an optimal solution with minimum possible number of fireflies and ants. This study also compares the MmTSP's solution of case study using the hybrid FA-ACO and the ACO.

The experiment showed that $\gamma = 0.01$, 4 fireflies, 5 movements for each firefly, 100 iterations of the FA, $\alpha = 1, \beta = 5, \rho = 0.5, Q = 100$, 20 ants, and 229 iterations of the ACO are the best parameters to solve the case study using the hybrid FA-ACO. The hybrid FA-ACO obtained 13828 km as the best results with an average solution 14400.28 km and an average computation time 36.01579 seconds. While the ACO method obtained 13882 km as the best result with an average solution 14463.24 km and an average computation time 60.77814 seconds. In achieving the best solution, hybrid FA-ACO converged after 16.86102 seconds while ACO converged after 25.07218 seconds. The comparison showed that the hybrid FA-ACO yield better result with average computation time 40.74% and convergence time 32.75% faster than the ACO method.

Keywords: Multi-Depot Multiple Traveling Salesman Problem, Firefly Algorithm, Ant Colony Optimization, hybrid method.

KATA PENGANTAR

Syukur Alhamdulillah kepada Allah SWT atas rahmat dan hidayah-Nya, sehingga tesis berjudul "Penyelesaian *Multi-Depot Multiple Traveling Salesman Problem* Menggunakan *Hybrid Firefly Algorithm-Ant Colony Optimization*" dapat terselesaikan. Selama proses pengerjaan tesis, penulis telah mendapat dukungan dan bantuan dari berbagai pihak. Secara khusus, penulis mengucapkan terima kasih sebesar-besarnya kepada:

1. Bapak Dr. Budi Setiyono, S.Si., M.T. selaku Pembimbing I dan Bapak Dr. Ir. R. Imbang Danandjojo, M.T. selaku Pembimbing II yang telah banyak meluangkan waktu untuk membimbing penulis dengan segala saran.
2. Dr. Imam Mukhlash, S.Si., M.T., Prof. Dr. Mohammad Isa Irawan, M.T., Dr. Darmaji, S.Si., M.T. selaku penguji tesis dan Dr. Drs. Hariyanto, M.Si., Dr. Dwi Ratna Sulistyaningsih, S.Si., M.T. selaku penguji proposal tesis.
3. Prof. Dr. Ir. Adi Soeprijanto, MT selaku Direktur program Pascasarjana ITS, Prof. Dr. Erna Apriliani, M.Si selaku Ketua Jurusan Matematika FMIPA ITS, Dr. Subiono, M.S selaku Ketua Program Pascasarjana Matematika, dan Prof. Basuki Widodo, M.Sc., Ph.D selaku dosen wali penulis serta para dosen yang telah membagikan ilmu dalam perkuliahan.
4. Gulpi Qorik Oktagalu Pratamasunu, S.Pd., M.Kom., yang berperan sebagai Pembimbing III, atas ide-ide dalam menyempurnakan tesis ini.
5. Mama Luluk Ilmiati, Papa Salman Farisi, Mama Tutik Islamiyah, dan Daddy Sunarno atas doa, dukungan, dan kepercayaan yang tiada henti. Keluarga besar Ali Mahe dan Su'ud atas dukungan moral dan material.
6. Kepada Kementerian Pendidikan dan Kebudayaan Direktorat Jenderal Pendidikan Tinggi (Dirjen Dikti) yang telah memberikan bantuan beasiswa penuh kepada penulis selama menempuh studi pascasarjana melalui program Beasiswa Pendidikan Pascasarjana Dalam Negeri (BPPDN).
7. Para staf dan teman-teman di Jurusan Matematika yang tidak dapat penulis sebutkan satu per satu.

Penulis menyadari masih adanya keterbatasan dan kekurangan dalam tesis ini. Oleh karena itu, penulis mengharapkan kritik dan saran demi kesempurnaan tesis. Harapan penulis, semoga tesis ini dapat bermanfaat bagi penelitian di Indonesia. Amin.

Tesis ini, penulis persembahkan untuk Khaizuran Afkar Khawarizmi.

Surabaya, Juli 2015

Olief Ilmandira Ratu Farisi

DAFTAR ISI

Halaman

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman

BAB 1
PENDAHULUAN

Pada bab ini diuraikan alasan yang melatarbelakangi penelitian, rumusan masalah dan batasan masalah yang digunakan, tujuan-tujuan yang ingin dicapai dalam penelitian ini, dan manfaat yang diharapkan dari penelitian ini.

1.1 Latar Belakang

Traveling Salesman Problem (TSP) merupakan salah satu masalah optimasi kombinatorial. Prinsip dari TSP adalah bagaimana seorang *salesman* menemukan rute terpendek untuk mengunjungi seluruh kota pada suatu daerah tepat satu kali dan harus kembali ke kota awal keberangkatan. TSP dimodelkan sebagai permasalahan graf. Objek-objek pada TSP merupakan simpul graf. Hubungan objek-objek direpresentasikan oleh sisi graf berbobot.

TSP dapat diaplikasikan untuk permasalahan perencanaan kunjungan wisata, logistik, *integrated circuit* (IC), dan lain-lain. Objek TSP tidak hanya merepresentasikan kota tetapi juga dapat merepresentasikan tempat, *node*, komponen, dan lain-lain. Bobot dari hubungan objek-objek pada TSP dapat mendeskripsikan jarak tempuh antara dua kota, waktu tempuh, biaya, dan lain-lain.

Pada penerapannya, ada beberapa kasus yang tidak hanya memerlukan satu orang *salesman* saja. Tetapi dibutuhkan beberapa orang *salesman* untuk menyelesaikan suatu pekerjaan. Permasalahan seperti ini dikategorikan dalam *Multiple Traveling Salesman Problem* (mTSP). Pada mTSP, setiap *salesman* mencari rute terpendeknya sehingga semua kota dapat dikunjungi tepat satu kali, tetapi oleh satu *salesman* saja. Permasalahan mTSP dapat dikembangkan lagi dengan *multi-depot* (MmTSP), yaitu beberapa *salesman* berangkat dari kota keberangkatan yang berbeda-beda, yang disebut depot, dan harus kembali ke kota keberangkatannya masing-masing.

TSP termasuk dalam *Nondeterministic Polynomial-Hard* (NP-*hard*). Sehingga, untuk mencari nilai optimum TSP, harus menggunakan metode *Brute Force*, yaitu dengan mencari semua kemungkinan rute. Namun, semakin banyak

objek, semakin lama waktu yang diperlukan untuk menyelesaikan permasalahan TSP. Oleh karena itu, beberapa peneliti mengembangkan metode pendekatan untuk menyelesaikan TSP dan MmTSP.

Ghafurian dan Javadian (2011) melakukan penelitian mengenai penyelesaian MmTSP menggunakan *Ant Colony Optimization* (ACO). ACO adalah suatu metode yang mengadopsi cara kerja koloni semut dalam mencari makanan. Pada penelitian tersebut didapat bahwa ACO dapat menemukan solusi yang mendekati optimum dengan waktu yang lebih cepat daripada metode uji *Brute Force*.

ACO merupakan suatu algoritma yang diperkenalkan pertama kali oleh Marco Dorigo, dkk (1996) untuk menyelesaikan TSP. Pada penerapan TSP, setiap semut ditempatkan pada objek awal secara *random* yang kemudian berjalan melewati seluruh simpul graf. Setiap semut akan kembali ke sarangnya dan mempunyai rute masing-masing. Dari pencarian rute yang dihasilkan oleh semut-semut tersebut, solusi akhirnya adalah rute yang terpendek. Dari penelitian tersebut, semakin banyak objek pada TSP, semakin banyak semut dan iterasi yang diperlukan. Sehingga, semakin lama waktu yang diperlukan ACO untuk mendekati solusi optimum.

Gilang Kusuma Jati dan Suyanto (2011) mengusulkan metode lain untuk menyelesaikan TSP, yaitu menggunakan metode *Firefly Algorithm* (FA). FA adalah suatu metode yang mengadopsi cara kerja kunang-kunang. Pada penerapan TSP, setiap kunang-kunang merepresentasikan satu solusi. Setiap kunang-kunang akan bergerak mendekati kunang-kunang lain yang memiliki daya ketertarikan lebih besar. Hasil penelitian menunjukkan bahwa FA tanpa dikombinasikan dengan metode lain dapat digunakan untuk menyelesaikan TSP walaupun semula FA dikembangkan untuk menyelesaikan masalah optimasi kontinu. Namun, FA mudah terjebak pada solusi lokal optimum.

Dengan memanfaatkan kelebihan dan kekurangan pada FA dan ACO, Mimoun Younes (2013) mengembangkan kombinasi dari dua algoritma (*hybrid*) tersebut. Pada penelitian tersebut, metode *hybrid* FA-ACO diterapkan pada masalah optimasi *economic power dispatch*. Hasil penelitian menunjukkan bahwa solusi dan waktu komputasi *hybrid* FA-ACO lebih baik daripada metode ACO atau FA.

Dengan menggunakan metode *hybrid* FA-ACO, inisialisasi feromon awal pada metode ACO yang semula disamaratakan, akan diatur melalui solusi lokal FA. Sehingga, FA dapat merekomendasikan sisi-sisi yang merupakan bagian dari solusi optimum. Dengan cara ini, akan mengurangi banyak semut dan iterasi yang diperlukan untuk pencarian global menggunakan ACO. Oleh karena itu, hipotesis dalam penelitian ini adalah *hybrid* FA-ACO dapat digunakan untuk menyelesaikan MmTSP lebih baik daripada metode ACO.

Studi kasus yang digunakan dalam penelitian ini adalah studi kasus penentuan rute angkutan laut PT. (Persero) PELNI. Permasalahan tersebut masih perlu diteliti guna meningkatkan efektivitas dan efisiensi pelayanan angkutan laut oleh Kementerian Perhubungan. Data diperoleh dari penelitian Danandjojo (2014). Pada penelitian tersebut, terdapat lima puluh satu pelabuhan dengan lima pelabuhan sebagai pelabuhan fokus utama pembangunan tol laut.

Pada penelitian ini, diusulkan penyelesaian MmTSP menggunakan *hybrid* FA-ACO dengan memanfaatkan studi kasus penentuan rute angkutan laut. Penelitian ini juga akan mengobservasi waktu komputasi *hybrid* FA-ACO dibandingkan dengan ACO dalam menyelesaikan studi kasus.

1.2 Perumusan Masalah

Berdasarkan latar belakang tersebut, perumusan masalah dalam penelitian ini adalah bagaimana menyelesaikan MmTSP menggunakan *hybrid* FA-ACO dan bagaimana waktu komputasi *hybrid* FA-ACO dibandingkan dengan ACO dalam menyelesaikan MmTSP.

1.3 Batasan Masalah

Untuk menghindari meluasnya permasalahan yang akan diselesaikan, maka dalam penelitian ini masalah akan dibatasi pada

1. MmTSP yang dimaksud adalah MmTSP yang dimodelkan sebagai graf lengkap tak berarah.
2. Permasalahan studi kasus yang digunakan dalam penelitian adalah mencari rute angkutan laut penumpang terpendek dengan lima pelabuhan sebagai depot.

3. Banyak penumpang yang diangkut tidak diperhitungkan.
4. Terdapat satu kapal pada setiap depot.
5. Setiap pelabuhan hanya dapat menampung satu kapal.

1.4 Tujuan Penelitian

Berdasarkan perumusan masalah di atas, maka tujuan dari penelitian ini adalah

1. Menerapkan konsep dan cara kerja *hybrid* FA-ACO untuk menyelesaikan MmTSP.
2. Membandingkan waktu komputasi *hybrid* FA-ACO dengan ACO dalam menyelesaikan MmTSP.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah membantu PT. (Persero) PELNI menentukan rute di sekitar pelabuhan tol laut.

BAB 2
TINJAUAN PUSTAKA DAN DASAR TEORI

MmTSP merupakan pengembangan dari TSP. Permasalahan MmTSP ini memiliki kompleksitas yang lebih tinggi dibandingkan TSP. Penelitian yang terkait dengan penyelesaian MmTSP dilakukan oleh Soheil Ghafurian dan Nikbakhsh Javadian (2011). Penelitian tersebut menggunakan ACO karena ACO dianggap memiliki efisiensi yang tinggi. Pada tahap evaluasi, peneliti membandingkan dengan solusi eksak yang didapat dari Lingo 8.0. Hasil penelitian menunjukkan bahwa ACO mendekati solusi eksak dengan waktu yang lebih cepat dari Lingo 8.0.

Marco Dorigo, dkk (1996) mengenalkan metode *Ant Colony Optimization* (ACO) yang dinamakan *Ant System* (AS) untuk menyelesaikan TSP. Metode ini memberikan umpan balik positif yang menyebabkan pencarian solusi dalam waktu singkat. Metode ini memiliki komputasi yang terdistribusi sehingga menghindari kekonvergenan terlalu dini. Selain itu, metode ini juga menggunakan pendekatan *greedy* konstruktivis yang dapat membantu menemukan solusi yang dapat diterima pada tahap awal proses pencarian. Namun, untuk menghasilkan solusi terbaik dengan banyak semut yang disebar sedikit mungkin, kompleksitas waktu yang diperlukan adalah $\mathcal{O}(NC \cdot n^3)$ dengan NC adalah banyaknya iterasi. Jadi, metode ini memerlukan waktu yang cukup lama untuk menyelesaikan TSP dengan banyak objek.

Firefly Algorithm (FA) adalah metode *meta-heuristic* yang pertama kali dikenalkan oleh Xin She Yang (2010). FA semula didesain untuk menyelesaikan permasalahan optimasi kontinu. Untuk menyelesaikan masalah seperti TSP, FA didiskritisasi yang disebut sebagai *Evolutionary Discrete Firefly Algorithm* (EDFA). Penelitian mengenai penyelesaian TSP menggunakan FA dilakukan oleh Gilang Kusuma Jati dan Suyanto (2011). Pada EDFA untuk TSP, tiap kunang-kunang merepresentasikan satu permutasi solusi TSP. Hasil penelitian tersebut menunjukkan bahwa FA berjalan dengan baik untuk menyelesaikan TSP. Akan tetapi, FA mudah terjebak solusi lokal optimum karena pergerakan kunang-kunang

tidak memiliki arah. Sehingga, untuk mendapatkan hasil yang lebih baik, metode FA dapat dikombinasikan dengan metode lainnya.

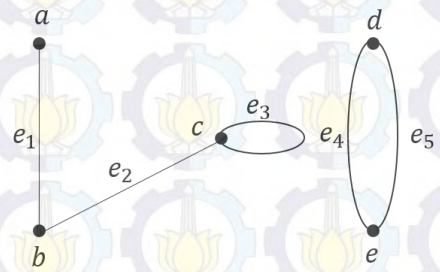
Berdasarkan tinjauan pustaka di atas, dasar teori yang menunjang penelitian ini dapat diuraikan sebagai berikut.

2.1 Teori Graf

Dasar-dasar teori graf yang digunakan untuk menunjang penelitian ini adalah sebagai berikut.

2.1.1 Definisi Graf

Suatu graf G adalah himpunan berhingga tak kosong dari objek-objek yang disebut simpul dan himpunan (yang mungkin kosong) pasangan tak terurut dari simpul yang berbeda pada G yang disebut sebagai sisi (Wilson dan Watkins, 1990). Himpunan simpul dari G dinotasikan dengan $V(G)$, sedangkan himpunan sisi dinotasikan dengan $E(G)$. Graf G dapat dinotasikan sebagai $G = (V, E)$.

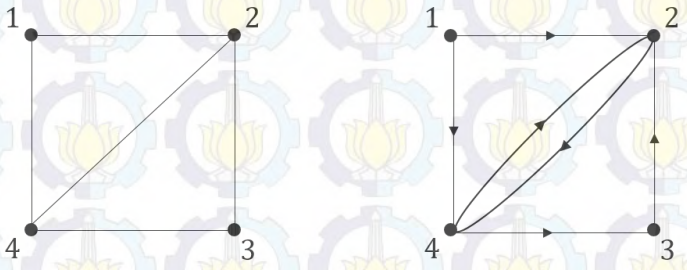


Gambar 2.1 Contoh graf

Pada Gambar 2.1, himpunan simpul $V(G) = \{a, b, c, d, e\}$ dan himpunan sisi $E(G) = \{(a, b), (b, c), (c, c), (d, e), (d, e)\} = \{e_1, e_2, e_3, e_4, e_5\}$. Sisi e_3 disebut *loop* yaitu sisi yang menghubungkan titik tunggal dengan dirinya sendiri. Simpul e_4 dan e_5 disebut sisi ganda karena kedua sisi tersebut menghubungkan dua simpul yang sama, yaitu d dan e. Simpul a bertetangga dengan simpul b karena dihubungkan oleh sisi e_1, sedangkan sisi e_1 dikatakan bersisian dengan simpul a dan b. Graf yang tidak memiliki *loop* dan sisi ganda dinamakan graf sederhana.

Suatu graf G memiliki order graf yang dinotasikan dengan $|G|$ dan ukuran graf yang dinotasikan dengan $||G||$. Order graf adalah banyak simpul pada graf. Sedangkan ukuran graf adalah banyaknya sisi pada graf. Setiap simpul pada graf memiliki *degree* (derajat) yang menunjukkan banyaknya sisi yang melekat pada simpul tersebut.

Berdasarkan orientasi arahnya, graf dapat dibedakan menjadi dua, yaitu graf tak berarah dan graf berarah. Graf tak berarah adalah graf yang sisinya tidak memiliki orientasi arah seperti pada Gambar 2.2(a). Graf berarah adalah graf yang sisinya memiliki orientasi arah seperti pada Gambar 2.2(b). Simpul pada graf tak berarah memiliki dua jenis derajat yaitu *out-degree* yang menunjukkan banyak sisi berarah yang berasal dari simpul tersebut dan *in-degree* yang menunjukkan banyaknya sisi berarah yang masuk ke simpul tersebut.



(a) (b)

Gambar 2.2 (a) Contoh graf tak berarah, $\deg(1) = 2$, $\deg(2) = 3$, $\deg(3) = 2$, $\deg(4) = 3$ (b) Contoh graf berarah, $\text{outdeg}(1) = 2$, $\text{outdeg}(2) = 1$, $\text{outdeg}(3) = 1$, $\text{outdeg}(4) = 2$, $\text{indeg}(1) = 0$, $\text{indeg}(2) = 3$, $\text{indeg}(3) = 1$, $\text{indeg}(4) = 2$.

2.1.2 Keterhubungan pada Graf

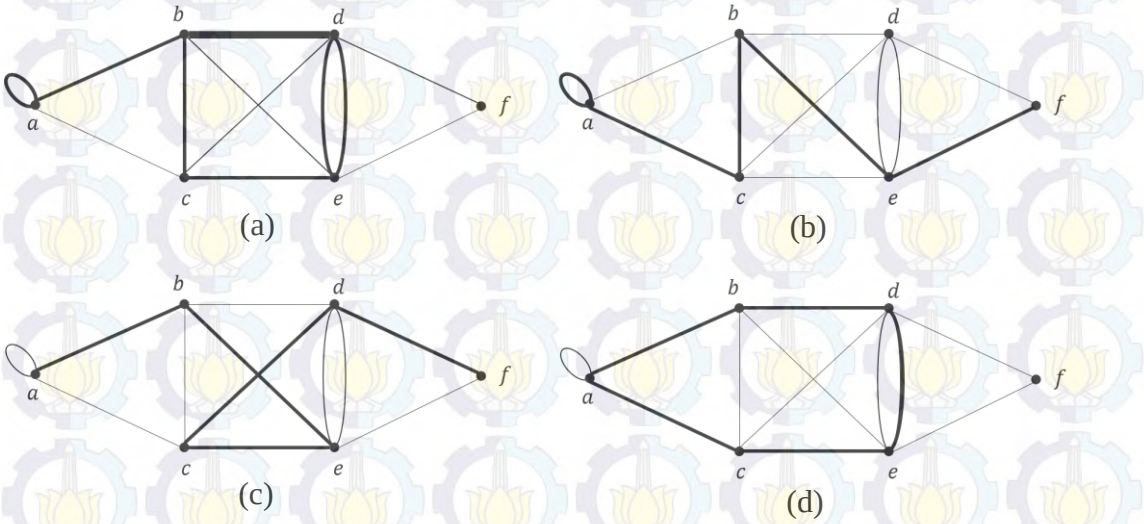
Pada suatu graf G, *walk* (jalan) dari simpul v_0 ke simpul v_n adalah barisan bergantian dari simpul dan sisi

$$W = \langle v_0, e_1, v_1, e_2, \dots, v_{n-1}, e_n, v_n \rangle$$

sedemikian hingga $endpts(e_i) = \{v_{i-1}, v_i\}$ untuk $i = 1,2, \dots, n$ (Gross dan Yellen, 2006). *Walk* dapat pula dinotasikan dengan $v_0 v_1 v_2 v_3 \dots v_{n-1} v_n$ yang berarti suatu *walk* dari v_0 ke v_n dengan panjang n.

Walk tertutup adalah *walk* yang simpul awalnya sama dengan simpul akhir. Dalam artian, *walk* tertutup dimulai dan berakhir pada simpul yang sama. Jika suatu *walk* dimulai dan berakhir di titik yang berbeda disebut dengan *walk* terbuka. Panjang dari suatu *walk* adalah banyaknya langkah sisi pada *walk*.

Suatu *walk* dengan semua barisan sisinya berbeda disebut sebagai *trail*. Sedangkan jika semua barisan sisi dan simpulnya berbeda maka *walk* tersebut dinamakan *path*. *Walk* tertutup dengan semua barisan sisi dan simpulnya berbeda akan membentuk suatu *cycle* (sirkuit).



Gambar 2.3 (a) *Walk* $aabdecbde$ dengan panjang 8. (b) *Trail* $aacbef$. (c) *Path* $abecdf$. (d) Sirkuit $abdec$.

Pada Gambar 2.3(a), *walk* dimulai dari simpul a melalui *loop* kembali ke simpul a, kemudian menuju ke b, kemudian ke d, dan seterusnya sampai berakhir di simpul e. Karena graf pada Gambar 2.3 tak berarah, *walk* dapat dimulai dari simpul e dan berakhir di simpul a yang dinotasikan sebagai $edbcedbaa$ dengan panjang 8. Suatu *walk* tidak harus berbeda semua simpul dan sisinya. Gambar 2.3(b) adalah contoh *trail* dari $aacbef$. Pada *trail* tersebut, semua sisi dilewati hanya satu kali tetapi simpul a dilewati dua kali. Gambar 2.3(c) adalah *path* $abecdf$ dengan semua sisi dan semua simpul dilewati hanya satu kali. Gambar 2.3(d) adalah

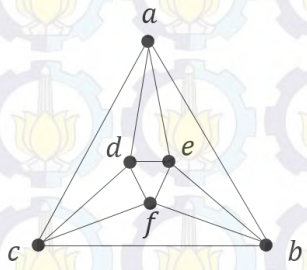
sirkuit $abdec$ yang dapat ditulis pula sebagai sirkuit $bdeca$, $decab$, $ecabd$, dan $cabde$ karena merupakan graf tak berarah.

Graf G dikatakan terhubung jika terdapat *path* pada G diantara sebarang pasangan simpul-simpul. Jika tidak, maka graf G disebut graf tak terhubung. Salah satu tipe graf terhubung adalah graf lengkap. Graf lengkap dengan n simpul (dinotasikan dengan K_n) adalah graf dengan setiap dua simpul yang berbeda dihubungkan oleh tepat satu sisi.

2.1.3 Graf Euler dan graf Hamilton

Suatu graf terhubung G merupakan graf Euler jika ada *trail* tertutup yang memuat semua sisi di G. *Trail* tersebut dinamakan *trail* Euler. Contoh tipe permasalahan graf Euler adalah penyusunan semua kartu pada domino, labirin, *Chinese Postman Problem*, dan lain-lain.

Suatu graf terhubung G merupakan graf Hamilton jika ada sirkuit yang memuat semua simpul di G. Sirkut tersebut dinamakan sirkuit Hamilton. Contoh tipe permasalahan graf Hamilton adalah masalah tur kuda pada papan catur, *gray code*, *Traveling Salesman Problem* (TSP), dan lain-lain.

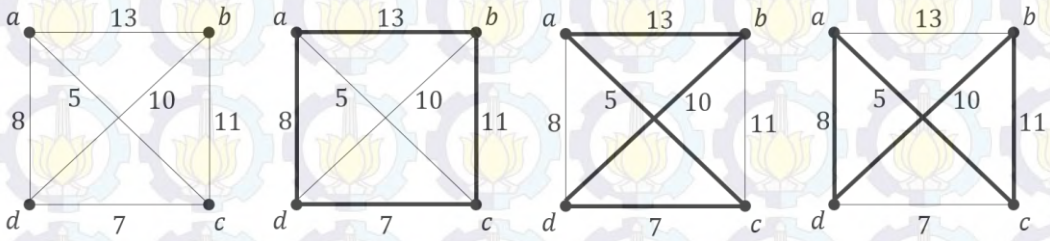


Gambar 2.4 Contoh graf Euler dan graf Hamilton

Graf pada Gambar 2.4 merupakan graf Euler karena ada *trail* tertutup $adeabefbcfdca$ yang memuat semua sisi dari graf. *Trail* $adeabefbcfdca$ merupakan *trail* Euler. Selain *trail* tersebut, masih ada *trail* Euler lainnya yang bisa ditemukan pada graf. Graf pada Gambar 2.4 juga merupakan graf Hamilton karena ada sirkuit $adefcba$ yang memuat semua simpul pada graf. Sirkuit $adefcba$ merupakan salah satu sirkuit Hamilton pada graf tersebut.

2.2 *Traveling Salesman Problem*

Traveling Salesman Problem (TSP) merupakan salah satu persoalan optimasi kombinatorial. TSP merupakan pengembangan dari masalah graf Hamilton. Prinsip TSP adalah mencari bobot minimum dari graf Hamilton yang berbobot. TSP dapat dinyatakan sebagai seorang *salesman* yang mengunjungi setiap kota di suatu daerah tepat satu kali dan kembali ke kota awal keberangkatan dengan rute yang menghasilkan bobot mimimum.



Gambar 2.5 (a) Graf K_4. (b),(c),(d) Kemungkinan sirkuit Hamilton dari graf K_4.

Diberikan graf lengkap K_4 tak berarah dengan bobot sisi yang merepresentasikan jarak (dalam km) seperti pada Gambar 2.5(a). Dengan asumsi kota awal dan kota akhir adalah kota a, banyaknya sirkuit Hamilton yang dapat dibentuk pada graf lengkap sebanyak 3, yaitu sirkuit $abcda$, $abdca$, dan $acbda$ yang ditunjukkan oleh Gambar 2.5(b),(c),(d). Solusi dari TSP tersebut adalah sirkuit $acbda$ dengan jarak minimum 34 km.

TSP dapat diformulasikan secara matematik sebagai berikut (Lenstra dan Rinnooy Kan, 1975). Diberikan himpunan berhingga dari kota-kota N dan anggota matriks jarak $c_{ij}, (i, j \in N)$ yang menyatakan jarak dari kota-i ke kota-j. Fungsi objektifnya adalah menentukan

$$\min_{\pi} \sum_{i \in N} c_{i\pi(i)}, \tag{2.1}$$

dengan π merupakan semua permutasi siklis dari N dan $\pi^k(i)$ adalah kota ke-k yang telah dilewati dari kota-i. Jika $N = \{1,2, \dots, n\}$, maka formulasi tersebut ekuivalen dengan

$$\min\left(\sum_{i=1}^{i=n-1} c_{v(i)v(i+1)} + c_{v(n)v(1)}\right), \tag{2.2}$$

dengan v mewakili semua permutasi dari N dan $v(k)$ adalah kota ke-k dari tur. Jika G adalah graf lengkap berarah dengan banyak simpul N dan bobot c_{ij}, maka solusi optimalnya adalah sirkuit Hamilton dengan total bobot minimum.

2.3 *Multiple Traveling Salesman Problem*

Multiple Traveling Salesman Problem (mTSP) adalah TSP dengan lebih dari satu *salesman*. Beberapa *salesman* ditempatkan pada satu kota keberangkatan yang disebut dengan depot dan harus kembali ke depot tersebut. Setiap *salesman* akan membentuk rutenya masing-masing sedemikian hingga semua kota pada suatu daerah dikunjungi dengan total bobot minimum. Setiap kota harus dikunjungi tepat satu kali oleh satu *salesman* saja.

Gambar 2.6 Contoh graf mTSP

mTSP dapat diformulasikan sebagai berikut (Bektas, 2006). Diberikan himpunan berhingga dari kota-kota N, himpunan sisi A, dan anggota matriks jarak $c_{ij}, (i, j \in N)$ yang menyatakan jarak dari kota-i ke kota-j. Diasumsikan kota-1 adalah depot dan ada m *salesman* pada depot tersebut. Didefinisikan variabel biner x_{ij} untuk setiap sisi $(i, j) \in A$, dengan x_{ij} bernilai 1 jika sisi (i, j) dilewati dalam tur dan x_{ij} bernilai 0 jika sisi (i, j) tidak dilewati. Fungsi objektifnya adalah

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{2.3}$$

dengan kendala

1. Tepat m *salesman* berangkat dari kota-1

$$\sum_{j \in N;(1,j) \in A} x_{ij} = m \tag{2.4}$$

2. Tepat m *salesman* kembali ke kota-1

$$\sum_{j \in N;(j,1) \in A} x_{j1} = m \tag{2.5}$$

3. Tepat satu tur yang datang pada setiap kota

$$\sum_{i \in N;(i,j) \in A} x_{ij} = 1, \forall j \in \{2,3,\dots,n\} \tag{2.6}$$

4. Tepat satu tur yang pergi pada setiap kota

$$\sum_{j \in N;(i,j) \in A} x_{ij} = 1, \forall i \in \{2,3,\dots,n\} \tag{2.7}$$

mTSP dapat dikembangkan lagi menjadi Multi-Depot mTSP (MmTSP). Pada MmTSP, beberapa *salesman* berangkat dari depot yang berbeda-beda dan harus kembali ke depotnya masing-masing. MmTSP dapat diformulasikan sebagai berikut (Kara dan Bektas, 2011).

Gambar 2.7 Contoh graf MmTSP

Diberikan himpunan berhingga dari kota-kota N, himpunan sisi A, dan anggota matriks jarak $c_{ij}, (i,j \in N)$ yang menyatakan jarak dari kota-i ke kota-j. Misalkan N dipartisi menjadi $N = N' \cup D$, dengan d kota pertama dari N adalah himpunan depot D. Terdapat m_i *salesman* ditempatkan di depot-i. Diberikan $N' = \{d+1, d+2, \dots, n\}$ adalah himpunan kota-kota yang akan dikunjungi.

Diasumsikan terdapat batasan banyaknya kota yang dapat dikunjungi pada suatu tur dengan batas bawah L dan batas atas U.

Didefinisikan variabel biner x_{ijk} bernilai 1 jika *salesman* depot-k berangkat dari kota-i ke kota-j dalam turnya dan x_{ijk} bernilai 0 jika sisi (i,j) tidak dilewati. Untuk sebarang *salesman*, u_i adalah banyaknya kota yang dilalui dari asalnya sampai kota-i. Sehingga, $1 \leq u_i \leq U, \forall i \geq 2$. Jika $x_{ikk} = 1$, maka $L \leq u_i \leq U$ harus memenuhi

$$\min \left(\sum_{k \in D} \sum_{j \in N'} \left(c_{kj} x_{kjk} + c_{jk} x_{jkk} \right) + \sum_{k \in D} \sum_{i \in N'} \sum_{j \in N'} c_{ij} x_{ijk} \right) \tag{2.8}$$

dengan kendala

1. Tepat m_k *salesman* berangkat dari masing-masing depot $k \in D$

$$\sum_{j \in N'} x_{kjk} = m_k, k \in D \tag{2.9}$$

2. Setiap kota dikunjungi tepat satu kali

$$\sum_{k \in D} x_{kjk} + \sum_{k \in D} \sum_{i \in N'} x_{ijk} = 1, j \in N' \tag{2.10}$$

3. Kontinuitas rute untuk kota yang dikunjungi

$$x_{kjk} + \sum_{i \in N'} x_{ijk} - x_{jkk} - \sum_{i \in N'} x_{jik} = 0, k \in D, j \in N' \tag{2.11}$$

4. Kontinuitas rute untuk depot

$$\sum_{j \in N'} x_{kjk} - \sum_{j \in N'} x_{jkk} = 0, k \in D \tag{2.12}$$

5. Batas atas tur

$$u_i + (U-2) \sum_{k \in D} x_{kik} + \sum_{k \in D} x_{ikk} \leq U - 1, i \in N' \tag{2.13}$$

6. Batas bawah tur

$$u_i + \sum_{k \in D} x_{kik} + (2-L) \sum_{k \in D} x_{ikk} \geq 2, i \in N' \tag{2.14}$$

7. Tur tidak diperbolehkan hanya satu kota

$$\sum_{k \in D} x_{kik} + \sum_{k \in D} x_{ikk} \leq 1, i \in N' \quad (2.15)$$

8. Eliminasi sub tur. Sub tur adalah tur tanpa depot sebagai kota awal atau kota akhir yang mungkin terdapat pada solusi

$$u_i - u_j + U \sum_{k \in D} x_{ijk} + (U-2) \sum_{k \in D} x_{jik} \leq U - 1, i \neq j;\ i, j \in N'. \quad (2.16)$$

2.4 *Ant Colony Optimization*

Ant Colony Optimization (ACO) adalah suatu algoritma optimasi yang meniru perilaku koloni semut. ACO pertama kali dikembangkan oleh Marco Dorigo, dkk pada 1991 untuk mencari lintasan terpendek. Koloni semut secara alami dapat menemukan lintasan terpendek dari sarang menuju sumber makanan dan kembali lagi.

Semut menggunakan substansi yang disebut feromon dalam komunikasi antara sesama individu. Feromon adalah zat kimia yang dihasilkan oleh kelenjar endokrin. Feromon berguna untuk mengenali sesama jenis, individu lain maupun kelompok dan untuk membantu proses reproduksi. Feromon yang dikeluarkan hanya dapat dikenali oleh individu dalam satu spesies. Setelah selang tertentu, feromon akan mengalami penguapan. Sehingga, akan mengurangi daya tarik lintasan.

Semut yang bergerak akan meninggalkan feromon di tanah sehingga menandai lintasan dengan jejak feromon. Ketika terdapat semut lain yang berjalan secara acak, semut tersebut akan dapat mendeteksi feromon yang ditinggalkan dan memutuskan jalan yang akan dilaluinya melalui besarnya probabilitas. Kemudian, semut tersebut juga meninggalkan jejak feromon sehingga memperbanyak kadar feromon pada lintasan tersebut.

Karakteristik dari ACO adalah memberikan umpan balik yang positif. Artinya, semakin banyak semut yang mengikuti jejak, semakin menarik jejak tersebut untuk diikuti. Probabilitas semut memilih lintasan bertambah sesuai dengan banyaknya semut sebelumnya yang memilih lintasan tersebut.

Pada ACO, semut buatan adalah agen dengan kemampuan dasar yang sederhana. Untuk dapat menggunakan ACO, terdapat beberapa asumsi yang membedakan koloni semut buatan dengan semut aslinya, antara lain sebagai berikut.

1. Semut buatan memiliki memori.
2. Semut buatan tidak sepenuhnya buta.
3. Semut buatan hidup di lingkungan dimana waktu adalah sesuatu yang diskrit.

2.4.1 *Ant System*

Ant System adalah algoritma ACO pertama yang dirumuskan untuk menyelesaikan TSP (Marco Dorigo, dkk, 1996). Pada algoritma ini, sebanyak m semut bekerja sama dan berkomunikasi menggunakan feromon. Agar dapat menyelesaikan TSP, setiap semut buatan memiliki karakterisktik sebagai berikut.

1. Semut memilih kota yang akan dikunjungi berdasarkan fungsi probabilitas yang dinamakan aturan transisi. Fungsi ini dipengaruhi oleh visibilitas (invers dari jarak) dan banyaknya jejak feromon yang menghubungkan dua kota.
2. Semut tidak diperbolehkan mengunjungi kota yang sama sebelum suatu tur terbentuk. Hal ini akan dikontrol oleh *tabu list*.
3. Ketika semut menyelesaikan suatu tur, semut akan meninggalkan jejak feromon pada setiap sisi (i, j) yang telah dikunjungi.

Cara kerja AS dapat dijelaskan sebagai berikut. Setiap semut ditempatkan di titik sebagai kota awal keberangkatan, yang berbeda dan acak. Semua semut akan mengunjungi satu per satu kota secara berulang kali sehingga akan menghasilkan suatu tur. Kota-kota yang telah dikunjungi oleh setiap semut disimpan dalam *tabu list*. Setelah semua semut menyelesaikan turnya, masing-masing panjang lintasan yang dihasilkan dihitung. Kemudian, banyak feromon akan diperbarui pada setiap semut melalui aturan pembaruan feromon global. Aturan tersebut meliputi penguapan pada semua sisi dan penambahan feromon pada sisi-sisi yang merupakan bagian dari tur. Semakin pendek tur yang dihasilkan, semakin banyak feromon yang ditinggalkan oleh semut pada sisi-sisinya. Hal ini mengakibatkan

sisi-sisi yang memuat banyak feromon akan lebih diminati pada tur-tur selanjutnya. Rute terpendek yang dihasilkan semut disimpan dan *tabu list* dikosongkan kembali untuk menyimpan kota-kota pada tur selanjutnya. Proses ini akan terus berulang sampai banyaknya tur mencapai banyak maksimum yang ditentukan atau sampai sistem tersebut berhenti untuk mencari solusi alternatif yang disebut stagnasi.

Diberikan N adalah himpunan kota-kota yaitu $N = \{1,2,\dots,n\}$ dan m adalah banyak semut yang disebar. Probabilitas semut-k dari kota-i memilih kota-j didefinisikan sebagai

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{u \in U_k}[\tau_{iu}(t)]^{\alpha}[\eta_{iu}]^{\beta}}, & \text{jika } j \in U_k \\ 0, & \text{lainnya} \end{cases}, \qquad (2.17)$$

dengan $\tau_{ij}(t)$ adalah intensitas *trail* sisi (i,j) pada waktu t, η_{ij} adalah visibilitas yang merupakan invers jarak ($\eta_{ij} = \frac{1}{d_{ij}}$) , dan U adalah himpunan kota-kota yang belum dikunjungi semut-k.

Intensitas *trail* $\tau_{ij}(t)$ merupakan banyaknya feromon yang tersebar pada sisi (i,j) pada waktu t. Setelah setiap semut menghasilkan satu tur, yaitu melewati *n* kota, intensitas *trail* akan diperbarui dengan aturan pembaruan feromon global yang didefinisikan sebagai

$$\tau_{ij}(t+n) = \rho \cdot \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \qquad (2.18)$$

dengan $(1-\rho)$ adalah koefisien yang menyatakan penguapan feromon dan $\Delta\tau_{ij}^{k}$ adalah kuantitas feromon yang dihasilkan oleh semut-k pada sisi (i,j) yang diformulasikan

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L_k}, & \text{jika semut-}k\text{ menggunakan sisi } (i,j) \text{ pada turnya} \\ 0, & \text{lainnya} \end{cases}, \qquad (2.19)$$

dengan Q adalah konstant dan L_k adalah panjang tur dari semut-k.

Secara umum, langkah-langkah *Ant System* dijabarkan sebagai berikut.

```
Input:
m semut dan n kota
Tentukan banyak iterasi NCmax
Untuk setiap sisi (i,j), inisialisasi τ_ij(t) dan Δτ_ij = 0
while NC<NCmax do
    for k = 1 to m do
        Tempatkan semut-k di kota awal secara random
        Simpan kota awal pada tabu list
    end
    repeat
        for k = 1 to m do
            Pilih kota-j dengan menghitung probabilitasnya
            Pindahkan semut-k ke kota-j
            Hitung panjang tur Lk
        end
    until tabu list terisi n kota
    for k = 1 to m do
        Pindahkan semut-k ke kota awal
        Hitung rute yang dihasilkan semut-k
        Hitung feromon yang dihasilkan semut-k
    end
    Simpan rute terpendek
    Update feromon trail dan set dan Δτ_ij = 0
    Kosongkan kembali tabu list
end
Output: tur terpendek
```

2.5 *Firefly Algorithm*

Firefly Algorithm (FA) adalah salah satu algoritma yang terinspirasi oleh alam, yaitu kunang-kunang. FA diperkenalkan pertama kali oleh Xin She Yang pada 2007 untuk menyelesaikan permasalahan optimasi kontinu. Kunang-kunang merupakan serangga yang menghasilkan cahaya sebagai sistem sinyal.

Cahaya pada kunang-kunang dihasilkan oleh pancaran cahaya biokomia. Cahaya ini berfungsi untuk menarik kunang-kunang lainnya sebagai tanda kawin, berkomunikasi, dan menarik mangsa. Cahaya ini juga sebagai peringatan rasa pahit saat musuh akan memangsa kunang-kunang.

Intensitas cahaya yang dipancarkan kunang-kunang dipengaruhi oleh jarak kunang-kunang satu dengan kunang-kunang lainnya dan penyerapan cahaya oleh udara. Intensitas cahaya akan semakin berkurang jika jaraknya semakin jauh. Demikian pula udara juga akan menyerap cahaya sehingga intensitas cahaya akan semakin berkurang lagi. Dua faktor ini yang menyebabkan penglihatan kunang-kunang terbatas.

Untuk dapat menggunakan FA, kunang-kunang diasumsikan memiliki karakteristik sebagai berikut.

1. Semua kunang-kunang adalah *unisex*. Sehingga, kunang-kunang akan tertarik kepada kunang-kunang lainnya tanpa memedulikan jenis kelamin.
2. Ketertarikan sebanding dengan intensitas cahaya. Dengan demikian, kunang-kunang yang memiliki cahaya lebih redup akan menghampiri kunang-kunang dengan cahaya yang lebih terang. Ketertarikan berkurang jika jarak antara dua kunang-kunang semakin jauh. Jika tidak ada kunang-kunang yang cahayanya terang, maka kunang-kunang akan berjalan secara *random*.
3. Intensitas cahaya pada kunang-kunang ditentukan oleh fungsi objektif permasalahan.

2.5.1 *Evolutionary Discrete Firefly Algorithm*

Evolutionary Discrete Firefly Algorithm adalah algoritma kunang-kunang yang digunakan untuk menyelesaikan permasalahan diskrit, seperti TSP. Penyelesaian TSP dengan EDFA dirumuskan oleh Gilang Kusuma Jati dan Suyanto pada 2011. Untuk menyelesaikan TSP, satu kunang-kunang mewakili satu permutasi solusi.

Cara kerja EDFA untuk TSP (Jati dkk, 2013) dapat dijelaskan sebagai berikut. Dipilih n kunang-kunang sebagai populasi awal, masing-masing mewakili satu *random* solusi. Sehingga terdapat n solusi sebagai solusi awal. Kemudian, setiap kunang-kunang mendekati kunang-kunang lain yang lebih menarik. Jika tidak ada kunang-kunang lain yang lebih menarik, maka kunang-kunang bergerak secara *random*. Ketika kunang-kunang bergerak, representasi solusi sebelumnya dari kunang-kunang tersebut berubah. Pergerakan setiap kunang-kunang menggunakan mutasi invers sebanyak m kali. Sehingga, pada akhir setiap iterasi akan ada $n + mn$ solusi. Pergerakan kunang-kunang ini menyebabkan perubahan solusi yang direpresentasikan oleh setiap kunang-kunang. Dari $n + mn$ solusi yang ada, pilih n solusi terbaik yang akan digunakan sebagai populasi awal untuk iterasi berikutnya.

Pada EDFA, fungsi ketertarikan kunang-kunang-j terhadap kunang-kunang-i $\beta(i,j)$ didefinisikan sebagai fungsi menurun monoton

$$\beta(i,j) = \beta_0 e^{-\gamma r^2}, \tag{2.20}$$

dengan r adalah jarak antara dua kunang-kunang, β_0 adalah ketertarikan pada saat $r = 0$ atau intensitas cahaya yang dipancarkan oleh kunang-kunang yang lebih terang, dan $\gamma \in [0, \infty)$ adalah koefisien penyerapan cahaya.

Intensitas cahaya merupakan suatu nilai yang merepresentasikan fungsi objektif dari permasalahan. Pada TSP, intensitas cahaya merupakan total jarak yang dihasilkan oleh kunang-kunang. Semakin kecil total jarak, semakin terang pula intensitas cahaya yang dipancarkan kunang-kunang. Intensitas cahaya dihitung sebagai invers dari total jarak.

Nilai penyerapan cahaya γ menentukan seberapa cepat kekonvergenan algoritma EDFA. Nilai ini juga menentukan karakteristik dari permasalahan. Jika $\gamma \to 0$, maka $\beta(r) = \beta_0$. Sehingga, ketertarikan kunang-kunang akan tetap ketika terlihat oleh kunang-kunang lain. Sedangkan jika $\gamma \to \infty$, maka nilai ketertarikan akan mendekati nol. Dalam artian, kunang-kunang tidak dapat terlihat oleh kunang-kunang lain akibat penyerapan cahaya yang terlalu besar. Sehingga, pemilihan koefisien γ berperan penting karena akan mempengaruhi ketertarikan kunang-kunang.

Jarak dari sebarang kunang-kunang-i dan kunang-kunang-j adalah banyaknya perbedaan sisi diantara keduanya dibandingkan dengan banyaknya kota yang diformulasikan sebagai

$$r = \frac{A}{N} \times 10, \tag{2.21}$$

dengan A adalah banyaknya sisi yang berbeda dan N adalah banyaknya kota.

Tabel 2.2 Representasi kunang-kunang-i dan kunang-kunang-j dan perbedaan sisi kunang-kunang-i dengan kunang-kunang-j

Representasi Kunang-kunang-i	3	4	2	8	9	7	5	10	1	6
Representasi Kunang-kunang-j	3	4	2	7	5	9	8	10	1	6

Pada Tabel 2.2, ada empat sisi yang dimiliki oleh kunang-kunang-i, yaitu 2-8, 8-9, 9-7, dan 7-5 yang tidak dimiliki oleh kunang-kunang-j. Sehingga, jarak antara dua kunang-kunang tersebut adalah $r = \frac{4}{10} \times 10 = 4$.

Pergerakan kunang-kunang-i yang tertarik dengan kunang-kunang-j yang lebih terang didefinisikan oleh

$$x_i = random\ (2, A), \tag{2.22}$$

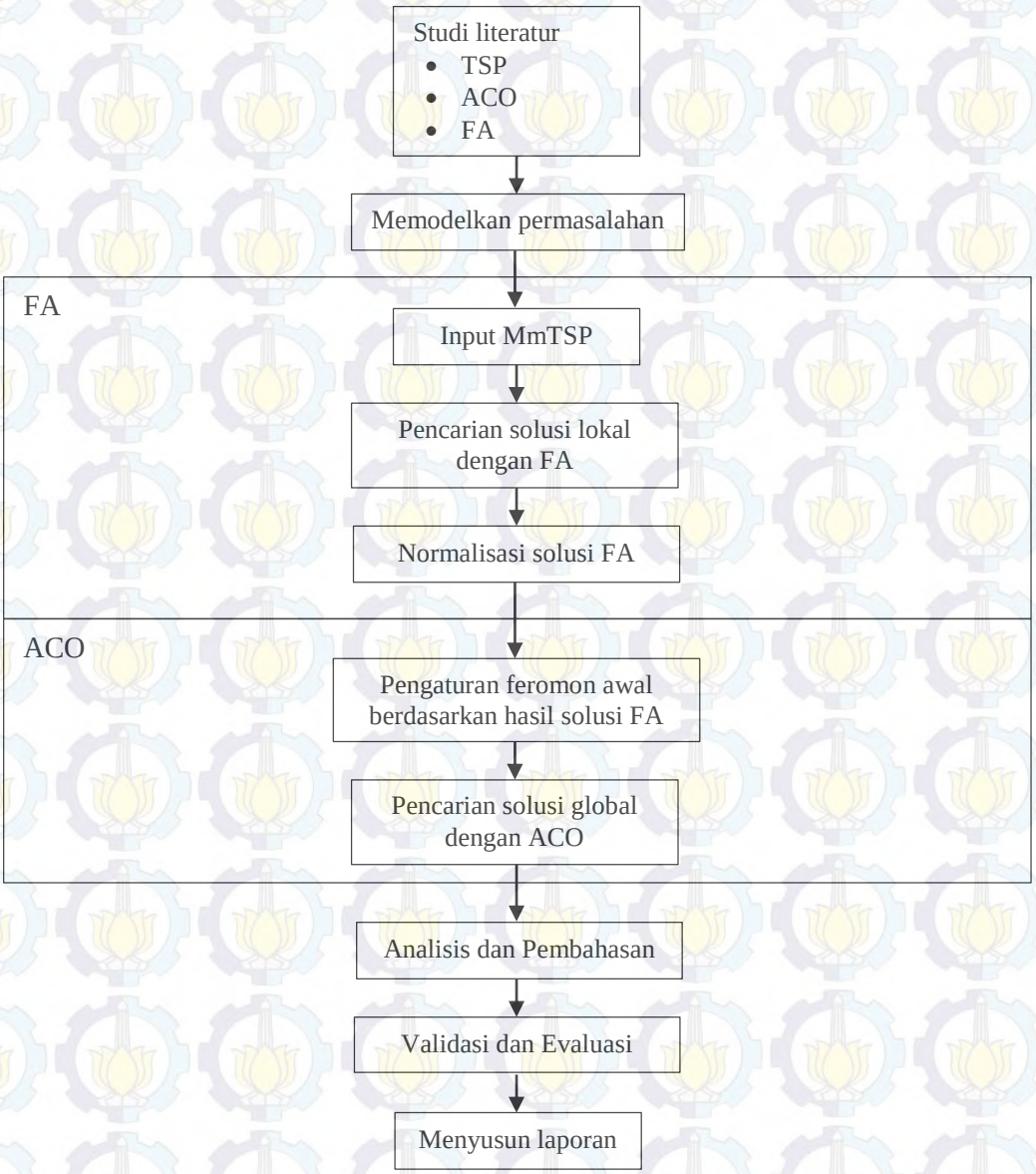
dengan A adalah banyak sisi yang berbeda dari kunang-kunang-i dengan kunang-kunang-j. Karena kunang-kunang mewakili permutasi dari solusi, maka pergerakan kunang-kunang dilakukan menggunakan mutasi invers. Sehingga, tidak merusak bentuk dari permutasi sebelumnya. Pergerakan ini mengakibatkan permutasi solusi pada setiap kunang-kunang berubah.

Secara umum, langkah-langkah *Firefly Algorithm* dijelaskan sebagai berikut.

```
Input:
Fungsi objektif f(x), x = (x_1, x_2, ..., x_d)^T
Inisialisasi populasi kunang-kunang x_i (i = 1,2, ..., n)
Definisikan koefisien penyerapan cahaya γ
for i = 1 to n do
    x_i merepresentasikan solusi awal
end
repeat
    for i = 1 to n do
        Tentukan x_j adalah kunang-kunang dengan ketertarikan lebih
        besar dari x_i
        if (x_j ≠ null) then
            Kunang-kunang-i bergerak mendekati kunang-kunang-j
            sebanyak m kali
        else
            Kunang-kunang-i bergerak secara random sebanyak m
            kali
        end
    end
    Pilih n kunang-kunang terbaik dari n + mn
until kriteria penghentian benar
Output: x_i min
```

BAB 3
METODE PENELITIAN

Pada bab ini diuraikan metode penelitian yang akan digunakan untuk mencapai tujuan penelitian. Tahapan metode penelitian antara lain sebagai berikut.



Gambar 3.1 Tahapan metode penelitian

Berdasarkan Gambar 3.1, tahapan metode penelitian dapat dijelaskan lebih rinci sebagai berikut.

1. Studi literatur

Pada tahap ini, dilakukan observasi TSP dan pemahaman teori mengenai TSP, MmTSP dan penyelesaiannya menggunakan FA dan ACO dengan mencari referensi yang menunjang penelitian. Referensi yang digunakan adalah tugas akhir, tesis, *paper*, jurnal, dan *internet*.

2. Memodelkan permasalahan

Pada tahap ini, didapat data PT. (Persero) PELNI untuk penentuan rute angkutan laut sebagai studi kasus dari penelitian ini. Pada studi kasus, terdapat lima puluh satu pelabuhan dengan lima kapal sebagai depot. Pada masing-masing depot hanya terdapat satu kapal. Fungsi objektifnya adalah mendapatkan rute dengan jarak minimum. Kendala yang terdapat pada permasalahan ini adalah hanya terdapat satu kapal pada masing-masing depot dan setiap pelabuhan hanya dapat menampung satu kapal. Sehingga, permasalahan tersebut dapat dimodelkan sebagai berikut.

Diberikan himpunan pelabuhan $N = \{1,2, \dots ,51\}$, dan himpunan sisi A, anggota matriks $c_{ij}, (i,j \in N)$ yang menyatakan jarak dari pelabuhan-i ke pelabuhan-j. Misalkan N dipartisi menjadi $N = N' \cup D$, dengan d pelabuhan pertama dari N adalah anggota himpunan *depot* D. Terdapat 1_i kapal ditempatkan di masing-masing *depot*-i. Diberikan $N' = \{d+1, d+2, \dots, n\}$ adalah himpunan pelabuhan yang akan dikunjungi.

Didefinisikan variabel biner x_{ijk} bernilai 1 jika kapal depot-k berangkat dari pelabuhan-i ke pelabuhan-j dalam turnya dan x_{ijk} bernilai 0 jika sisi (i,j) tidak dilewati. Diasumsikan terdapat batasan banyaknya pelabuhan yang dapat dikunjungi pada suatu tur dengan batas bawah L dan batas atas U. Untuk sebarang kapal, u_i adalah banyaknya pelabuhan yang dilalui dari asalnya sampai pelabuhan-i. Sehingga, $1 \leq u_i \leq U, \forall i \geq 2$. Jika $x_{ikk} = 1$, maka $L \leq u_i \leq U$ harus memenuhi

$$\min\left(\sum_{k\in D}\sum_{j\in N'}\left(c_{kj}x_{kjk}+c_{jk}x_{jkk}\right)+\sum_{k\in D}\sum_{i\in N'}\sum_{j\in N'}c_{ij}x_{ijk}\right) \quad (3.1)$$

dengan kendala

- Tepat 1_k kapal berangkat dari masing-masing *depot* $k \in D$.

$$\sum_{j\in N'}x_{kjk}=1_k, k\in D \quad (3.2)$$

- Setiap pelabuhan dikunjungi tepat satu kali.

$$\sum_{k\in D}x_{kjk}+\sum_{k\in D}\sum_{i\in N'}x_{ijk}=1, j\in N' \quad (3.3)$$

- Kontinuitas rute untuk pelabuhan yang dikunjungi.

$$x_{kjk}+\sum_{i\in N'}x_{ijk}-x_{jkk}-\sum_{i\in N'}x_{jik}=0, k\in D, j\in N' \quad (3.4)$$

- Kontinuitas rute untuk *depot*.

$$\sum_{j\in N'}x_{kjk}-\sum_{j\in N'}x_{jkk}=0, k\in D \quad (3.5)$$

- Batas atas tur.

$$u_i+(U-2)\sum_{k\in D}x_{kik}+\sum_{k\in D}x_{ikk}\le U-1, i\in N' \quad (3.6)$$

- Batas bawah tur.

$$u_i+\sum_{k\in D}x_{kik}+(2-L)\sum_{k\in D}x_{ikk}\ge 2, i\in N \quad (3.7)$$

- Tur tidak diperbolehkan hanya satu pelabuhan.

$$\sum_{k\in D}x_{kik}+\sum_{k\in D}x_{ikk}\le 1, i\in N' \quad (3.8)$$

- Eliminasi sub tur. Sub tur adalah tur tanpa *depot* sebagai pelabuhan awal atau pelabuhan akhir yang mungkin terdapat pada solusi.

$$u_i-u_j+U\sum_{k\in D}x_{ijk}+(U-2)\sum_{k\in D}x_{jik}\le U-1, \quad (3.9)$$

dengan $i\neq j;\ i,j\in N$.

3. *Input MmTSP*

Input pada permasalahan ini adalah matriks jarak berukuran 51×51. Bobot untuk suatu pelabuhan dengan pelabuhan itu sendiri adalah nol.

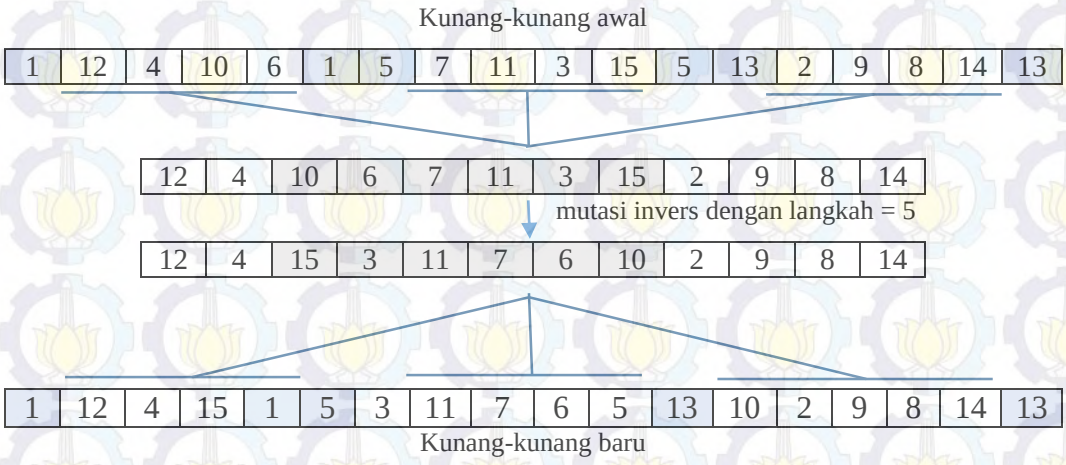
4. *Pencarian solusi lokal dengan FA*

Pada tahap ini dilakukan pencarian solusi lokal dengan menentukan banyaknya kunang-kunang p dan banyak pergerakan m yang dijalankan oleh tiap kunang-kunang. Algoritma FA dijalankan terlebih dahulu untuk mencari solusi lokal karena kemampuan kekonvergenan FA yang cepat. Gambar 4.1 menunjukkan representasi kunang-kunang untuk MMTSP dengan 1, 5, dan 13 sebagai depot.

1	12	4	10	6	1	5	7	11	3	15	5	13	2	9	8	14	13

Gambar 4.1 Contoh representasi kunang-kunang untuk MMTSP

Setiap kunang-kunang bergerak menuju kunang-kunang dengan ketertarikan lebih besar sebanyak m kali. Pergerakan kunang-kunang untuk MMTSP menggunakan mutasi invers dengan memperhatikan batas bawah dan batas atas pelabuhan yang boleh dikunjungi. Mutasi invers dilakukan pada pelabuhan-pelabuhan selain depot. Sebagai contoh, mutasi invers suatu kunang-kunang sebanyak lima langkah ditunjukkan pada Gambar 4.2. Diberikan pula batas bawah pelabuhan sebanyak 3 dan batas atas pelabuhan sebanyak 5.



Gambar 4.2 Mutasi invers kunang-kunang untuk MMTSP

Kemudian, dipilih banyak pelabuhan yang dapat dikunjungi oleh masing-masing depot secara random dari 3 sampai 5. Selanjutnya, dibentuk kunang-kunang baru dari hasil mutasi invers. Setiap iterasi akan menghasilkan $pm + p$ populasi kunang-kunang. Pilih p kunang-kunang terbaik untuk iterasi selanjutnya. Hal ini dilakukan sampai mencapai iterasi maksimum yang ditentukan.

Pada studi kasus, setiap kunang-kunang merepresentasikan solusi dari lima pelabuhan sebagai depot. Intensitas cahaya tiap kunang-kunang dihitung berdasarkan fungsi objektif. Semakin kecil jarak yang ditempuh, semakin besar intensitas cahaya yang dihasilkan kunang-kunang.

5. Normalisasi solusi FA

Dari $p + mp$ hasil solusi FA, terdapat beberapa kunang-kunang merujuk pada rute yang sama. Sehingga, diperlukan normalisasi solusi untuk dijadikan kandidat rute. Kunang-kunang dengan rute yang sama dihitung sebagai satu solusi. Pilih p kunang-kunang terbaik untuk dinormalisasi. Setelah dinormalisasi, akan didapat q kunang-kunang hasil dari normalisasi untuk pencarian solusi global.

6. Pengaturan feromon awal berdasarkan hasil solusi FA

Dari q solusi kunang-kunang, setiap sisi (i, j) bagian dari solusi akan diberikan penambahan feromon untuk inisialisasi feromon ACO dengan memberlakukan kelipatan

$$\Delta t_{i,j} = \frac{10}{q} \times (q - k + 1) \tag{3.10}$$

dengan (i, j) merupakan sisi pada solusi terbaik ke-k dari q solusi yang ada.

Setelah didapat solusi normalisasi, dilakukan pengaturan feromon awal dengan menambahkan feromon pada setiap sisi yang dilalui pada kandidat rute. Sehingga, sisi yang paling banyak dilalui pada kandidat rute akan mendapatkan feromon paling banyak. Feromon awal ini digunakan sebagai inisialisasi feromon untuk menjalankan ACO.

7. Pencarian solusi global dengan ACO

Pada tahap ini, ditentukan banyaknya semut dan banyak iterasi yang akan digunakan untuk menjalankan ACO. Algoritma ACO digunakan untuk mencari solusi global karena ACO dapat menemukan solusi yang lebih mendekati optimum. Output dari metode ini adalah rute dengan jarak minimum.

Pada studi kasus, sebanyak m semut ditempatkan di depot yang sama. Semut akan mengunjungi pelabuhan-pelabuhan selain depot berdasarkan probabilitasnya. Banyak pelabuhan-pelabuhan yang boleh dikunjungi juga ditentukan secara random dari batas bawah sampai batas atas yang ditentukan.

8. Analisis dan pembahasan

Pada tahap ini, dilakukan uji coba untuk menentukan parameter terbaik dari metode ACO dan *hybrid* FA-ACO dalam menyelesaikan studi kasus. Pada metode hybrid FA-ACO, uji coba dilakukan untuk menentukan minimal banyak kunang-kunang dan banyak semut yang diperlukan agar algoritma *hybrid* FA-ACO mendapatkan solusi optimum. Percobaan juga diperlukan untuk iterasi minimum dari studi kasus agar hasilnya mencapai hasil optimum.

9. Validasi dan evaluasi

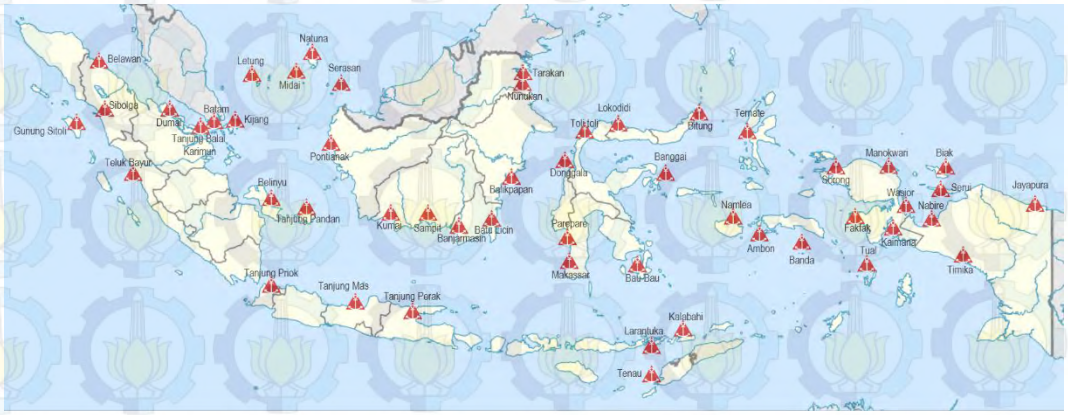
Uji validitas yang digunakan pada penelitian ini adalah uji validitas secara empiris, yaitu membandingkan hasil yang diperoleh dari perhitungan manual dengan perhitungan program. Hasil yang didapat dari *hybrid* FA-ACO akan dibandingkan dengan hasil yang didapat jika menggunakan ACO saja. Hal yang dibandingkan yaitu pencapaian hasil optimum, waktu komputasi, dan konvergensi.

10. Menyusun laporan

Hasil analisis dan perbandingan *hybrid* FA-ACO dengan FA dan ACO serta kesimpulan yang terkait dengan penyelesaian MmTSP dengan *hybrid* FA-ACO disusun dalam bentuk laporan tesis sebagai dokumentasi dari penelitian ini.

BAB 4
PEMBAHASAN DAN ANALISIS HASIL

Pada bab ini dijabarkan hasil penelitian tentang penyelesaian MmTSP menggunakan *hybrid* FA-ACO pada studi kasus rute angkutan laut. Studi kasus yang digunakan dalam uji coba adalah studi kasus rute angkutan laut pada 51 pelabuhan dengan lima pelabuhan sebagai depot. Persebaran 51 pelabuhan tersebut ditunjukkan oleh Gambar 4.1.



Gambar 4.1 Persebaran 51 Pelabuhan di Indonesia

Lima pelabuhan yang menjadi depot, yaitu Tanjung Priok, Belawan, Tanjung Perak, Makassar, dan Sorong yang termasuk dalam pelabuhan besar di Indonesia. Kelima pelabuhan tersebut nantinya merupakan pelabuhan fokus utama pembangunan tol laut. Pada penelitian ini, hanya ada satu kapal yang akan berangkat dari setiap depot dan kembali ke depot masing-masing. Hasil dari penelitian ini berupa rute dan jarak terpendek angkutan laut.

Studi kasus rute angkutan laut dapat dimodelkan dalam bentuk graf. Untuk mempermudah penulisan rute, 51 pelabuhan diberi kode seperti pada Tabel 4.1 berikut.

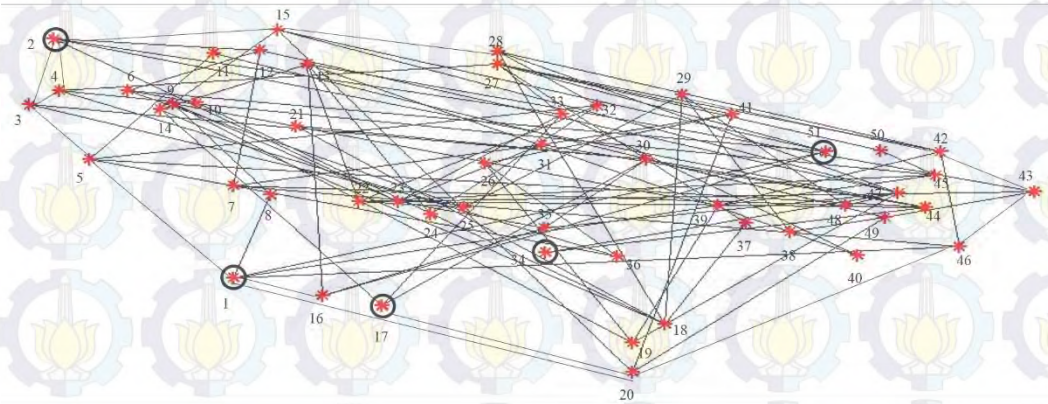
Tabel 4.1 Kode 51 pelabuhan

Pelabuhan	Kode
Tanjung Priok	1
Belawan	2
Gunung Sitoli	3
Sibolga	4
Teluk Bayur	5
Dumai	6
Blinyu	7
Tanjung Pandan	8
Batam	9
Kijang	10
Letung	11
Midai	12
Serasan	13
Tanjung Balai Karimun	14
Natuna	15
Tanjung Mas	16
Tanjung Perak	17

Pelabuhan	Kode
Kalabahi	18
Larantuka	19
Tenau	20
Pontianak	21
Kumai	22
Sampit	23
Banjarmasin	24
Batu Licin	25
Balikpapan	26
Nunukan	27
Tarakan	28
Bitung	29
Banggai	30
Donggala / Pantoloan	31
Loko Didi	32
Toli-toli	33
Makassar	34

Pelabuhan	Kode
Pare Pare	35
Bau-bau	36
Ambon	37
Banda	38
Namlea	39
Tual	40
Ternate	41
Biak	42
Jayapura	43
Nabire	44
Serui	45
Timika	46
Wasior	47
Fakfak	48
Kaimana	49
Manokwari	50
Sorong	51

Model graf dari studi kasus dapat digambarkan seperti pada Gambar 4.2 berikut.

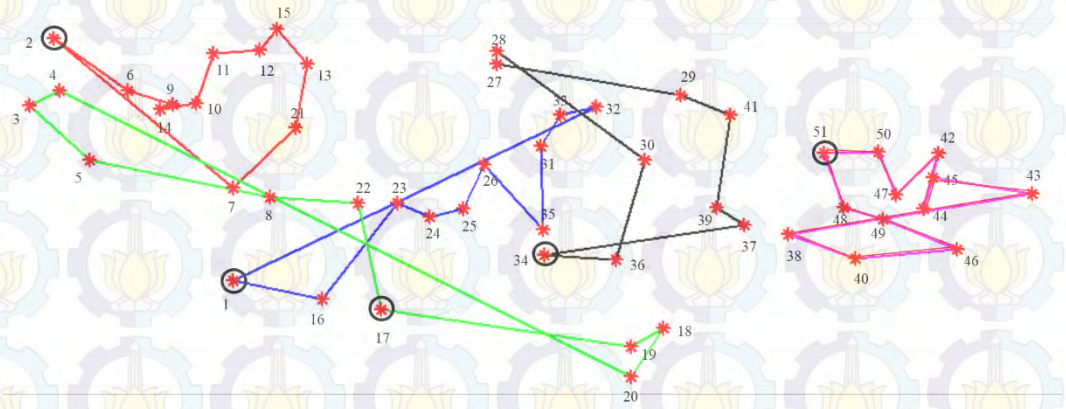


Gambar 4.2 Model graf studi kasus

Jarak antar pelabuhan (lihat Lampiran 1) dimodelkan sebagai matriks jarak sebagai berikut.

	1	2	3	4	5	6	⋯	50	51
1	0	863	772	764	556	689	⋯	1780	1577
2	863	0	584	614	885	291	⋯	2240	2248
3	772	584	0	75	219	828	⋯	2530	2327
4	764	614	75	0	211	858	⋯	2522	2319
5	556	885	219	211	0	1147	⋯	2330	2127
6	689	291	828	858	1147	0	⋯	2249	2058
	⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮	⋮
50	1780	2240	2530	2522	2330	2249	⋯	0	216
51	1577	2248	2327	2319	2127	2058	⋯	216	0

Contoh salah satu solusi MmTSP pada studi kasus ditunjukkan sebagai berikut.



Gambar 4.3 Salah satu solusi studi kasus

Model pemrograman linear MmTSP pada studi kasus adalah sebagai berikut.

Diberikan himpunan pelabuhan $N = \{1,2,\dots,51\}$, dan himpunan sisi A, anggota matriks $c_{ij}, (i,j \in N)$ yang menyatakan jarak dari pelabuhan-i ke pelabuhan-j. Misalkan N dipartisi menjadi $N = U \cup D$, dengan $d \in \{1,2,17,34,51\}$ adalah anggota himpunan *depot* D. Terdapat 1_i kapal ditempatkan di masing-masing *depot*-i. Diberikan $N' = \{d+1, d+2, \dots, n\}$ adalah himpunan pelabuhan yang akan dikunjungi.

Didefinisikan variabel biner x_{ijk} bernilai 1 jika kapal depot-k berangkat dari pelabuhan-i ke pelabuhan-j dalam turnya dan x_{ijk} bernilai 0 jika sisi (i,j) tidak dilewati. Diasumsikan terdapat batasan banyaknya pelabuhan yang dapat dikunjungi pada suatu tur dengan batas bawah L dan batas atas U yang memenuhi (Ghafurian dan Javadian, 2011) batas bawah L dan batas atas U adalah bilangan bulat acak sedemikian hingga $2 \leq L \leq 9$ dan $9 \leq U \leq 45$, berturut-turut. Sehingga, pada penelitian ini digunakan batas bawah $L = 8$ dan batas atas $U = 11$.

Untuk sebarang kapal, u_i adalah banyaknya pelabuhan yang dilalui dari asalnya sampai pelabuhan-i. Sehingga, $1 \leq u_i \leq 10, \forall i \geq 2$. Jika $x_{ikk} = 1$, maka $8 \leq u_i \leq 11$ harus memenuhi

$$\min\left(\sum_{k\in D}\sum_{j\in N'}\left(c_{kj}x_{kjk} + c_{jk}x_{jkk}\right) + \sum_{k\in D}\sum_{i\in N,}\sum_{j\in N,} c_{ij}x_{ijk}\right) \quad (4.1)$$

dengan kendala

1. Tepat 1_k kapal berangkat dari masing-masing *depot* $k \in D$.

$$\sum_{j\in N,} x_{kjk} = 1_k, k \in D \quad (4.2)$$

2. Setiap pelabuhan dikunjungi tepat satu kali.

$$\sum_{k\in D} x_{kjk} + \sum_{k\in D}\sum_{i\in N,} x_{ijk} = 1, j \in N' \quad (4.3)$$

3. Kontinuitas rute untuk pelabuhan yang dikunjungi.

$$x_{kjk} + \sum_{i\in N,} x_{ijk} - x_{jkk} - \sum_{i\in N'} x_{jik} = 0, k \in D, j \in N' \quad (4.4)$$

4. Kontinuitas rute untuk *depot*.

$$\sum_{j\in N'} x_{kjk} - \sum_{j\in N'} x_{jkk} = 0, k \in D \quad (4.5)$$

5. Batas atas tur.

$$u_i + 9\sum_{k\in D} x_{kik} + \sum_{k\in D} x_{ikk} \leq 10, i \in N' \quad (4.6)$$

6. Batas bawah tur.

$$u_i + \sum_{k\in D} x_{kik} + (-6)\sum_{k\in D} x_{ikk} \geq 2, i \in N \quad (4.7)$$

7. Tur tidak diperbolehkan hanya satu pelabuhan.

$$\sum_{k\in D} x_{kik} + \sum_{k\in D} x_{ikk} \leq 1, i \in N' \quad (4.8)$$

8. Eliminasi sub tur. Sub tur adalah tur tanpa *depot* sebagai pelabuhan awal atau pelabuhan akhir yang mungkin terdapat pada solusi.

$$u_i - u_j + 11\sum_{k\in D} x_{ijk} + 9\sum_{k\in D} x_{jik} \leq 10, \quad (4.9)$$

dengan $i \neq j;\ i,j \in N$.

4.1 Implementasi *Hybrid* FA-ACO

Implementasi algoritma dilakukan dengan mengembangkan setiap proses yang telah dipaparkan pada Bab 3. Pada subbab ini akan dibahas mengenai pencarian solusi lokal menggunakan FA, proses normalisasi solusi, pengaturan feromon berdasarkan hasil normalisasi, dan pencarian solusi global menggunakan ACO.

4.1.1 Pencarian Solusi Lokal Menggunakan FA

Pada metode FA, perlu ditentukan banyak kunang-kunang dan banyak pergerakan tiap kunang-kunang serta koefisien penyerapan cahaya γ. Sebagai ilustrasi, kunang-kunang yang digunakan sebanyak dua dengan satu pergerakan untuk tiap kunang-kunang dan $\gamma = 0{,}005$. Banyak pelabuhan pada studi kasus (N) adalah 51 dan banyak depot (D) adalah 5.

Langkah 1: Membentuk p representasi kunang-kunang secara random.

Pada tahap pertama ini, dibangun p kunang-kunang secara random sebagai populasi awal untuk pencarian solusi lokal FA. Misalkan dua kunang-kunang memiliki representasi seperti berikut.

Kunang-kunang-1

1	38	35	21	29	10	16	8	3	28	9	1	2	11
6	48	43	22	25	36	30	4	2	17	13	31	23	32
15	12	20	44	5	17	34	41	49	33	47	46	42	24
26	50	27	34	51	40	37	7	18	14	39	45	19	51

Kunang-kunang-2

1	38	35	21	29	10	16	8	3	28	9	1	2	11
6	48	43	20	5	12	44	13	15	2	17	33	41	22
36	31	49	24	30	25	42	17	34	4	47	46	23	32
26	50	27	34	51	40	37	7	18	14	39	45	19	51

Algoritma untuk membentuk p kunang-kunang dituliskan sebagai berikut.

```
% membentuk populasi kunang-kunang
for i = 1 to p do
    buat representasi kunang-kunang x(i);
end
```

Langkah 2: Menghitung jarak rute yang dihasilkan oleh masing-masing kunang-kunang.

- Kunang-kunang-1

Depot 1: 1-38-35-21-29-10-16-8-3-28-9-1

Jarak rute depot-1 $= 1463 + 830 + 851 + 1454 + 1700 + 617 + 335 + 942 + 1968 + 1305 + 672 = 12137$

Depot 2: 2-11-6-48-43-22-25-36-30-4-2

Jarak rute depot-2 $= 598 + 407 + 2061 + 806 + 2039 + 402 + 404 + 194 + 1913 + 614 = 9438$

Depot-17: 17-13-31-23-32-15-12-20-44-5-17

Jarak rute depot-17 $= 842 + 1438 + 603 + 862 + 1577 + 132 + 1685 + 997 + 2288 + 932 = 11356$

Depot-34: 34-41-49-33-47-46-42-24-26-50-27-34

Jarak rute depot-34 $= 755 + 625 + 1136 + 1125 + 1030 + 1049 + 1509 + 357 + 1197 + 1078 + 610 = 10471$

Depot-51: 51-40-37-7-18-14-39-45-19-51

Jarak rute depot-51 $= 407 + 325 + 1475 + 1219 + 1386 + 1582 + 706 + 1011 + 706 = 8817$

Total jarak yang dihasilkan oleh kunang-kunang-1 =

$12137 + 9438 + 11356 + 10471 + 8817 = 52219$

- Kunang-kunang 2

Depot 1: 1-38-35-21-29-10-16-8-3-28-9-1

Jarak rute depot-1 $= 1463 + 830 + 851 + 1454 + 1700 + 617 + 335 + 942 + 1968 + 1305 + 672 = 12137$

Depot-2: 2-11-6-48-43-20-5-12-44-13-15-2

Jarak rute depot-2 $= 598 + 407 + 2061 + 806 + 1448 + 1597 + 1514 + 2183 + 2104 + 115 + 673 = 13506$

Depot-17: 17-33-41-22-36-31-49-24-30-25-42-17

Jarak rute depot-17 $= 814 + 545 + 1252 + 686 + 522 + 1201 + 1422 + 796 + 598 + 1313 + 1534 = 10683$

Depot-34: 34-4-47-46-23-32-26-50-27-34

Jarak rute depot-34 $= 1525 + 2608 + 1030 + 1657 + 862 + 455 + 1197 + 1078 + 610 = 11022$

Depot-51: 51-40-37-7-18-14-39-45-19-51

Jarak rute depot-51 $= 407 + 325 + 1475 + 1219 + 1386 + 1582 + 706 + 1011 + 706 = 8817$

Total jarak yang dihasilkan oleh kunang-kunang-2 =

$= 12137 + 13506 + 10683 + 11022 + 8817 = 56165$

Secara umum, algoritma perhitungan jarak yang dihasilkan tiap kunang-kunang adalah sebagai berikut.

```
% menghitung jarak untuk tiap kunang-kunang
for i to p do
    jarak(i) = 0;
    for j to N-D-1 do
        if (sisi (j,j+1) bukan anggota depot)
            jarak(i) = jarak(i) + d(j,j+1);
        end
    end
end
```

Langkah 3: Menghitung intensitas cahaya

Intensitas cahaya merupakan invers dari total jarak yang dihasilkan kunang-kunang.

Intensitas cahaya kunang-kunang-1 $= I_1 = \frac{1}{total\ jarak} = \frac{1}{52219} = 0{,}00001915$

Intensitas cahaya kunang-kunang-2 $= I_2 = \frac{1}{total\ jarak} = \frac{1}{56165} = 0{,}0000178$

Algoritma untuk menghitung intensitas cahaya dari p kunang-kunang sebagai berikut.

```
% menghitung intensitas cahaya untuk tiap kunang-kunang
for i to p do
    intensitas(i) = 1/jarak(i);
end
```

Langkah 4: Mencari jarak antar kunang-kunang

Jarak antar kunang-kunang dapat dihitung dari pelabuhan-pelabuhan yang dapat dikunjungi selain depot. Banyak pelabuhan yang dapat dikunjungi $(N - D)$ adalah 46.

Urutan pelabuhan yang dikunjungi kunang-kunang-1

38	35	21	29	10	16	8	3	28	9	11	6	48	43	22	25
36	30	4	13	31	23	32	15	12	20	44	5	41	49	33	47
46	42	24	26	50	27	40	37	7	18	14	39	45	19		

Urutan pelabuhan yang dikunjungi kunang-kunang-2

38	35	21	29	10	16	8	3	28	9	11	6	48	43	20	5
12	44	13	15	33	41	22	36	31	49	24	30	25	42	4	47
46	23	32	26	50	27	40	37	7	18	14	39	45	19		

Dari representasi di atas, terdapat sisi 43-22, 22-25, 25-36, 36-30, 30-4, 4-13, 13-31, 31-23, 32-15, 15-12, 12-20, 20-44, 44-5, 5-41, 41-49, 49-33, 33-47, 46-42, 42-24, dan 24-26 yang dimiliki kunang-kunang-1 tetapi tidak dimiliki oleh kunang-kunang-2. Sehingga, terdapat 20 perbedaan sisi.

Jarak kunang-kunang-1 dan kunang-kunang-2 dihitung menggunakan persamaan (2.21) adalah

$$r = \frac{A}{N} \times 10 = \frac{20}{51} \times 10 = 3{,}9216.$$

Dari langkah tersebut, algoritma untuk mencari jarak kunang-kunang secara umum adalah sebagai berikut.

```
% menghitung jarak antar kunang-kunang
for i=1 to p do
    himpun pelabuhan-pelabuhan selain depot U(i);
end

for i=1 to p do
    for j=1 to p do
        if (i==j)
            a(i,j) = inf;
        else
            cari a(i,j) yaitu jarak U(i) dengan U(j);
        end
        hitung jarak menggunakan persamaan (2.21)
    end
end
```

Langkah 5: Menghitung daya tarik kunang-kunang

Daya tarik kunang-kunang terhadap kunang-kunang lain dihitung menggunakan persamaan (2.20). Daya tarik suatu kunang-kunang dengan dirinya sendiri sama dengan intensitas cahaya yang dimilikinya.

Daya tarik kunang-kunang-1 terhadap kunang-kunang-2

$$\beta(2,1) = I_1 e^{-0{,}005 \cdot 3.9216^2} = 0.00001915 e^{-0{,}005 \cdot 3.9216^2} = 0{,}00001886$$

Daya tarik kunang-kunang-2 terhadap kunang-kunang-1

$$\beta(1,2) = I_2 e^{-0{,}005 \cdot 3.9216^2} = 0.0000178 e^{-0{,}005 \cdot 3.9216^2} = 0{,}000017533$$

Untuk mencari daya tarik p kunang-kunang terhadap p kunang-kunang lainnya dapat dicari dengan algoritma berikut.

```
% menghitung daya tarik antar kunang-kunang
for i=1 to p do
    for j=1 to p do
        hitung daya tarik kunang-kunang menggunakan persamaan (2.20)
    end
end
```

Langkah 6: Kunang-kunang mencari kunang-kunang lain dengan daya tarik besar

- Kunang-kunang-1

  Daya tarik kunang-kunang-2 terhadap kunang-kunang-1 lebih kecil daripada daya tarik kunang-kunang 1 terhadap dirinya sendiri ($0{,}000017533 < 0{,}00001915$). Karena tidak ada kunang-kunang yang

daya tariknya lebih besar dari kunang-kunang-1, maka kunang-kunang-1 akan bergerak secara random.

- Kunang-kunang-2

  Daya tarik kunang-kunang-1 terhadap kunang-kunang-2 lebih besar daripada daya tarik kunang-kunang-2 terhadap dirinya sendiri ($0{,}0000178 > 0{,}00001886$). Sehingga, kunang-kunang-2 akan bergerak menuju kunang-kunang-1.

Langkah tersebut dapat digeneralisasi dengan algoritma berikut.

```
% kunang-kunang mencari kunang-kunang lain dengan daya tarik besar
for i=1 to p do
    Dapatkan kunang-kunang j dengan tingkat ketertarikan terbesar,
    dimana i≠j
    if (ada kunang-kunang j dengan tingkat ketertarikan terbesar)
        kunang-kunang i bergerak menuju kunang-kunang j;
    else kunang-kunang i bergerak secara random;
    end
end
```

Langkah 7: Pergerakan kunang-kunang menggunakan mutasi invers

- Kunang-kunang-1

Kunang-kunang-1 bergerak secara random. Pilih titik awal (P_1) dan banyak langkah secara random. Pemilihan banyak langkah dibatasi dari 1 hingga $46 - P_1$ ($N - d - P_1$). Misalkan $P_1 = 5$, maka banyak langkah $= rand\ (1{,}41) = 25$. Didapat,

				P_1											
38	35	21	29	10	16	8	3	28	9	11	6	48	43	22	25
36	30	4	13	31	23	32	15	12	20	44	5	41	49	33	47
46	42	24	26	50	27	40	37	7	18	14	39	45	19		

↓

38	35	21	29	49	41	5	44	20	12	15	32	23	31	13	4
30	36	25	22	43	48	6	11	9	28	3	8	16	10	33	47
46	42	24	26	50	27	40	37	7	18	14	39	45	19		

Hasil pergerakan di atas, kemudian dibentuk menjadi kunang-kunang baru (kunang-kunang-3) yang memenuhi batas atas dan batas bawah banyak pelabuhan. Selanjutnya, kunang-kunang-3 ini ditambahkan dalam populasi.

Kunang-kunang-3

1	38	35	21	29	49	41	5	44	20	1	2	12	15
32	23	31	13	4	30	36	2	17	25	22	43	48	6
11	9	28	3	8	17	34	16	10	33	47	46	42	24
26	50	34	51	27	40	37	7	18	14	39	45	19	51

- Kunang-kunang-2

Kunang-kunang-2 bergerak menuju kunang-kunang-1. Titik awal P_1 ditentukan di permulaan sisi kunang-kunang-2 yang berbeda dengan kunang-kunang-1. Pemilihan banyak langkah secara random dibatasi dari 1 hingga A. Sehingga, banyak langkah $= rand(1, A) = rand\ (1{,}20) = 16$.

38	35	21	29	10	16	8	3	28	9	11	6	48	43	20	5
12	44	13	15	33	41	22	36	31	49	24	30	25	42	4	47
46	23	32	26	50	27	40	37	7	18	14	39	45	19		

↓ P_1

38	35	21	29	10	16	8	3	28	9	11	6	48	43	4	42
25	30	24	49	31	36	22	41	33	15	13	44	12	5	20	47
46	23	32	26	50	27	40	37	7	18	14	39	45	19		

Hasil pergerakan di atas, kemudian dibentuk menjadi kunang-kunang baru (kunang-kunang-4) yang memenuhi batas atas dan batas bawah banyak pelabuhan. Selanjutnya, kunang-kunang-4 ini ditambahkan dalam populasi.

Kunang-kunang-4

1	38	35	21	29	10	16	8	3	28	9	1	2	11
6	48	43	4	42	25	30	24	2	17	49	31	36	22
41	33	15	13	44	12	17	34	5	20	47	46	23	32
26	50	34	51	27	40	37	7	18	14	39	45	19	51

Ulangi langkah 7 sesuai dengan banyak pergerakan tiap kunang-kunang (m) yang ditentukan.

Secara umum, algoritma untuk p kunang-kunang dengan m pergerakan mutasi invers adalah sebagai berikut.

```
%kunang-kunang bergerak menuju kunang-kunang dg daya tarik besar
for i = 1 to p do
    for mk = 1 to m do
        if (tidak ada kunang-kunang dengan daya tarik lebih besar)
            Tentukan P1 secara random sebagai posisi awal mutasi
            invers;
            Tentukan banyak langkah secara random;
            Lakukan mutasi invers;
            Bentuk kunang-kunang baru;
            Tambahkan kunang-kunang baru dalam populasi;
        else
            Tentukan P1 yang menunjukkan urutan posisi sisi yang
            berbeda dari kunang-kunang U(i) dengan U(j)
            Tentukan banyak langkah secara random dari 1 sampai a(i,j)
            Lakukan mutasi invers;
            Bentuk kunang-kunang baru;
            Tambahkan kunang-kunang baru dalam populasi;
        end
    end
end
```

Langkah 8: Menghitung jarak rute yang dihasilkan oleh kunang-kunang baru

- Kunang-kunang-3

Depot-1: 1-38-35-21-29-49-41-5-44-20-1

Jarak rute depot-1 $= 1463 + 830 + 851 + 1454 + 738 + 625 + 2029 + 2288 + 997 + 1108 = 12383$

Depot-2: 2-12-15-32-23-31-13-4-30-36-2

Jarak rute depot-2 $= 753 + 132 + 1577 + 862 + 603 + 1438 + 1418 + 1913 + 194 + 1873 = 10763$

Depot-17: 17-25-22-43-48-6-11-9-28-3-8-17

Jarak rute depot-17 $= 347 + 402 + 2039 + 806 + 2061 + 407 + 217 + 1305 + 1968 + 942 + 486 = 10980$

Depot-34: 34-16-10-33-47-46-42-24-26-50-34

Jarak rute depot-34 $= 584 + 617 + 1384 + 1125 + 1030 + 1049 + 1509 + 357 + 1197 + 1096 = 9948$

Depot-51: 51-27-40-37-7-18-14-39-45-19-51

Jarak rute depot-51 $= 936 + 1208 + 325 + 1475 + 1219 + 1386 + 1582 + 706 + 1011 + 706 = 10554$

Total jarak yang dihasilkan oleh kunang-kunang-3 =

$12383 + 10763 + 10980 + 9948 + 10554 = 54628$

- Kunang-kunang-4

Depot-1: 1-38-35-21-29-10-16-8-3-28-9-1

Jarak rute depot-1 $= 1463 + 830 + 851 + 1454 + 1700 + 617 + 335 + 942 + 1968 + 1305 + 672 = 12137$

Depot-2: 2-11-6-48-43-4-42-25-30-24-2

Jarak rute depot-2 $= 598 + 407 + 2061 + 806 + 2931 + 2623 + 1313 + 598 + 796 + 1137 = 13270$

Depot-17: 17-49-31-36-22-41-33-15-13-44-12-17

Jarak rute depot-17 $= 1454 + 1201 + 522 + 686 + 1252 + 545 + 1490 + 115 + 2104 + 2183 + 921 = 12473$

Depot-34: 34-5-20-47-46-23-32-26-50-34

Jarak rute depot-34 $= 1308 + 1597 + 1126 + 1030 + 1657 + 862 + 455 + 1197 + 1096 = 10328$

Depot-51: 51-27-40-37-7-18-14-39-45-19-51

Jarak rute depot-51 $= 936 + 1208 + 325 + 1475 + 1219 + 1386 + 1582 + 706 + 1011 + 706 = 10554$

Total jarak yang dihasilkan oleh kunang-kunang-4 =

$12137 + 13270 + 12473 + 10328 + 10554 = 58762$

Langkah 9: Memilih populasi kunang-kunang untuk iterasi selanjutnya

Karena pada contoh terdapat dua kunang-kunang sebagai populasi awal, maka diambil dua kunang-kunang yang menghasilkan jarak rute lebih baik.

Jarak rute kunang-kunang-1 = 52219

Jarak rute kunang-kunang-2 = 56165

Jarak rute kunang-kunang-3 = 54628

Jarak rute kunang-kunang-4 = 58762

Sehingga, kunang-kunang yang terpilih untuk iterasi selanjutnya adalah kunang-kunang-1 dan kunang-kunang-3.

Algoritma untuk membentuk populasi kunang-kunang baru adalah sebagai berikut.

```
% populasi kunang-kunang baru
Urutkan kunang-kunang dari hasil minimum
for i=1:p
   x(i) = kunang-kunang dengan minimum(i)
end
```

Langkah 10: Lakukan langkah 2-9 sampai memenuhi stopping condition.

Algoritma pencarian solusi lokal menggunakan FA dapat diuraikan sebagai berikut.

```
% membentuk populasi kunang-kunang
for i = 1 to p do
    buat representasi kunang-kunang x(i);
end
repeat
   % menghitung jarak untuk tiap kunang-kunang
   for i to p do
       jarak(i) = 0;
       for j to N-D-1 do
           if (sisi (j,j+1) bukan anggota depot)
               jarak(i) = jarak(i) + d(j,j+1);
           end
       end
   end
   % menghitung intensitas cahaya untuk tiap kunang-kunang
   for i to p do
       intensitas(i) = 1/jarak(i);
   end
   % menghitung jarak antar kunang-kunang
   for i=1 to p do
       himpun pelabuhan-pelabuhan selain depot U(i);
   end

   for i=1 to p do
       for j=1 to p do
           if (i==j)
               a(i,j) = inf;
           else
               cari a(i,j) yaitu jarak U(i) dengan U(j);
           end
           hitung jarak menggunakan persamaan (2.21)
       end
   end
   % menghitung daya tarik antar kunang-kunang
   for i=1 to p do
       for j=1 to p do
           hitung daya tarik kunang-kunang menggunakan persamaan (2.20)
       end
   end
   % kunang-kunang mencari kunang-kunang lain dengan daya tarik besar
   for i=1 to p do
        Dapatkan kunang-kunang j dengan tingkat ketertarikan terbesar,
        dimana i≠j
        if (ada kunang-kunang j dengan tingkat ketertarikan terbesar)
             kunang-kunang i bergerak menuju kunang-kunang j;
        else kunang-kunang i bergerak secara random;
        end
   end
   %kunang-kunang bergerak menuju kunang-kunang dg daya tarik besar
   for i = 1 to p do
```

```
        for mk = 1 to m do
            if (tidak ada kunang-kunang dengan daya tarik lebih besar)
                Tentukan P1 secara random sebagai posisi awal mutasi
                invers;
                Tentukan banyak langkah secara random;
                Lakukan mutasi invers;
                Bentuk kunang-kunang baru;
                Tambahkan kunang-kunang baru dalam populasi;
            else
                Tentukan P1 yang menunjukkan urutan posisi sisi yang
                berbeda dari kunang-kunang U(i) dengan U(j)
                Tentukan banyak langkah secara random dari 1 sampai
                a(i,j)
                Lakukan mutasi invers;
                Bentuk kunang-kunang baru;
                Tambahkan kunang-kunang baru dalam populasi;
            end
        end
    end
    % populasi kunang-kunang baru
    Urutkan kunang-kunang dari hasil minimum
    for i=1:p
       x(i) = kunang-kunang dengan minimum(i)
    end
until (memenuhi stopping condition)
```

4.1.2 Pengaturan Feromon Awal Berdasarkan Hasil Solusi FA

Dari p solusi kunang-kunang terbaik, akan terdapat beberapa solusi yang merujuk pada satu solusi yang sama. Oleh karena itu, diperlukan normalisasi agar solusi yang sama dihitung sebagai satu solusi saja. Dari q solusi hasil normalisasi, akan diberikan tambahan feromon untuk sisi (i,j) yang merupakan bagian dari solusi. Sisi pada solusi terbaik pertama mendapatkan feromon yang lebih banyak dibandingkan sisi pada solusi terbaik kedua. Tahap awal dari pengaturan feromon adalah mengatur inisialisasi feromon. Sebagai contoh, diberikan matriks feromon awal sebagai berikut.

	1	2	3	4	5	6	7	8	9	⋯	44	45	46	47	48	49	50	51
1	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
9	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
44	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
45	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
46	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
47	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
48	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
49	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
50	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
51	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1

Langkah 1: Normalisasi solusi lokal FA

Pada akhir iterasi FA, akan didapatkan solusi lokal FA dari populasi kunang-kunang yang digunakan. Misalkan populasi kunang-kunang akhir sebagai berikut.

Kunang-kunang-1

1	8	22	37	45	44	50	41	19	21	7	1	2	9
10	13	12	15	6	14	16	18	2	17	35	33	29	30
36	24	23	26	17	34	5	3	4	20	39	32	31	28
27	38	34	51	42	47	48	49	46	40	11	25	43	51

Kunang-kunang-2

1	8	22	37	45	44	50	41	19	21	7	1	2	9
10	13	12	15	6	14	16	18	2	17	35	33	29	30
36	24	23	26	17	34	5	3	4	20	39	32	31	28
27	38	34	51	42	47	48	49	46	40	11	25	43	51

Sehingga, hanya terdapat satu solusi hasil normalisasi, yaitu

Normalisasi solusi

1	8	22	37	45	44	50	41	19	21	7	1	2	9
10	13	12	15	6	14	16	18	2	17	35	33	29	30
36	24	23	26	17	34	5	3	4	20	39	32	31	28
27	38	34	51	42	47	48	49	46	40	11	25	43	51

Algoritma normalisasi solusi secara umum dapat ditentukan sebagai berikut.

```
% Normalisasi solusi lokal FA
Masukkan x(1) dalam normalisasi
for i = 2 to p do
    for j = 1 to i-1 do
        if (x(i)≠x(j))
            Masukkan x(i) dalam normalisasi
        end
    end
end
```

Langkah 2: Inisialisasi feromon awal

Tambahkan feromon pada sisi (i, j) yang merupakan bagian dari solusi normalisasi. Karena hanya terdapat satu solusi saja, maka feromon ditambahkan sebanyak

$$\Delta t_{i,j} = \frac{10}{q} \times (q - k + 1) = \frac{10}{1} \times (1 - 1 + 1) = 10.$$

	1	2	3	4	5	6	7	8	9	⋯	44	45	46	47	48	49	50	51
1	1	1	1	1	1	1	1	11	1	⋯	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	11	⋯	1	1	1	1	1	1	1	1
3	1	1	1	11	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
5	1	1	11	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
7	11	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
9	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
44	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	11	1
45	1	1	1	1	1	1	1	1	1	⋯	11	1	1	1	1	1	1	1
46	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
47	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	11	1	1	1
48	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	11	1	1
49	1	1	1	1	1	1	1	1	1	⋯	1	1	11	1	1	1	1	1
50	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
51	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1

Secara umum, algoritma untuk menginisialisasi feromon untuk q hasil solusi normalisasi dapat dituliskan sebagai berikut.

```
% Inisialisasi feromon
Set feromon awal terlebih dahulu
for i = 1 to q do
    Hitung penambahan feromon menggunakan persamaan (3.10)
    Tambahkan feromon dengan feromon awal;
    end
end
```

4.1.3 Pencarian Solusi Global Menggunakan ACO

Setelah dilakukan normalisasi dan pengaturan feromon, tahap selanjutnya adalah pencarian solusi global menggunakan ACO. Pada metode ACO diperlukan beberapa parameter antara lain α, β, ρ, Q dan banyak semut. Sebagai ilustrasi, semut yang digunakan sebanyak 1 dengan $\alpha = 1, \beta = 1, \rho = 0{,}5$, dan $Q = 10000$.

Langkah 1: Posisikan n semut pada salah satu depot

Diberikan 1 semut yang ditempatkan pada depot-1. Himpunan pelabuhan-pelabuhan yang dapat dikunjungi oleh semut tersebut adalah $U_k = \{3,4,5,6,7,8,$ 9,10,11,12,13,14,15,16,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,35,36, 37,38,39,40,41,42,43,44,45,46,47,48,49,50}

Algoritma untuk memposisikan n semut secara umum adalah sebagai berikut.

```
% Inisialisasi tur
for i = 1 to n do
   Masukkan depot ke-1 dalam tabu list semut(i);
End
```

Langkah 2: Menghitung visibilitas

Visibilitas merupakan invers dari jarak, sehingga didapat matriks visibilitas sebagai berikut.

	1	2	3	4	⋯	50	51
1	∞	0,001159	0,001295	0,001309	⋯	0,000562	0,000634
2	0,001159	∞	0,001712	0,001629	⋯	0,000446	0,000445
3	0,001295	0,001712	∞	0,013333	⋯	0,000395	0,00043
4	0,001309	0,001629	0,013333	∞	⋯	0,000397	0,000431
	⋮	⋮	⋮	⋮	⋱	⋮	⋮
50	0,000562	0,000446	0,000395	0,000397	⋯	∞	0,00463
51	0,000634	0,000445	0,00043	0,000431	⋯	0,00463	∞

Dari langkah di atas, algoritma untuk mencari visibilitas dapat ditulis sebagai berikut.

```
% menghitung visibilitas
visibilitas = 1/jarak antar pelabuhan;
```

Langkah 3: Menentukan pelabuhan yang akan dikunjungi selanjutnya

Penentuan pelabuhan yang yang akan dikunjungi dilakukan untuk setiap depot berdasarkan fungsi probabilitasnya pada persamaan (2.17). Banyaknya pelabuhan yang dapat dikunjungi untuk setiap depot ditentukan secara random dari batas bawah sampai batas atas.

Misalkan untuk depot-1, banyak pelabuhan yang dapat dikunjungi adalah 8.

- Pelabuhan pertama. Semut berada di pelabuhan 1.

$$p_{13}^{1}(1) = \frac{[\tau_{13}(1)]^1[\eta_{13}]^1}{\sum_{u\in U_1}[\tau_{1u}(t)]^1[\eta_{1u}]^1} = \frac{1 \cdot 0{,}001295}{0{,}1023} = 0{,}0127$$

$$p_{14}^{1}(1) = \frac{[\tau_{14}(1)]^1[\eta_{14}]^1}{\sum_{u\in U_1}[\tau_{1u}(t)]^1[\eta_{1u}]^1} = \frac{1 \cdot 0{,}001309}{0{,}1023} = 0{,}0128$$

$$p_{15}^{1}(1) = \frac{[\tau_{15}(1)]^1[\eta_{15}]^1}{\sum_{u\in U_1}[\tau_{1u}(t)]^1[\eta_{1u}]^1} = \frac{1 \cdot 0{,}0018}{0{,}1023} = 0{,}0176$$

$$p_{16}^{1}(1) = \frac{[\tau_{16}(1)]^1[\eta_{16}]^1}{\sum_{u\in U_1}[\tau_{1u}(t)]^1[\eta_{1u}]^1} = \frac{1 \cdot 0{,}0015}{0{,}1023} = 0{,}0142$$

Dengan cara yang sama, lakukan perhitungan probabilitas untuk semua pelabuhan yang dapat dikunjungi oleh semut. Kemudian, pilih suatu bilangan random $r = (0,1)$, contohnya 0,04. Bilangan ini untuk menentukan pelabuhan mana yang akan dikunjungi selanjutnya. Jika penjumlahan probabilitas secara bertahap lebih besar dari r, maka pelabuhan yang memiliki probabilitas akhir yang menjadi pelabuhan selanjutnya.

Iterasi 1: $p_{13}^{1}(1) = 0{,}0127 < r$

Iterasi 2: $p_{13}^{1}(1) + p_{14}^{1}(1) = 0{,}0127 + 0{,}0128 = 0{,}0255 < r$

Iterasi 3: $p_{13}^{1}(1) + p_{14}^{1}(1) + p_{15}^{1}(1) = 0{,}0127 + 0{,}0128 + 0{,}0176 = 0{,}0431 > r$

Pada iterasi ke-3 didapat bahwa penjumlahan probabilitas dari keempat pelabuhan lebih besar dari r. Sehingga, pelabuhan yang akan dikunjungi selanjutnya adalah pelabuhan 5. Masukkan pelabuhan 5 dalam tabu list semut 1.

- Pelabuhan kedua. Semut berada di pelabuhan 5.

$$p_{53}^{1}(1) = \frac{[\tau_{53}(1)]^1[\eta_{53}]^1}{\sum_{u\in U_1}[\tau_{5u}(t)]^1[\eta_{5u}]^1} = \frac{11 \cdot 0{,}0046}{0{,}0841} = 0{,}6017$$

$$p_{54}^{1}(1) = \frac{[\tau_{54}(1)]^1[\eta_{54}]^1}{\sum_{u\in U_1}[\tau_{5u}(t)]^1[\eta_{5u}]^1} = \frac{1 \cdot 0{,}0047}{0{,}0841} = 0{,}0558$$

$$p_{56}^{1}(1) = \frac{[\tau_{56}(1)]^1[\eta_{56}]^1}{\sum_{u\in U_1}[\tau_{5u}(t)]^1[\eta_{5u}]^1} = \frac{1 \cdot 0{,}0008}{0{,}0841} = 0{,}0095$$

$$p_{57}^{1}(1) = \frac{[\tau_{57}(1)]^1[\eta_{57}]^1}{\sum_{u\in U_1}[\tau_{5u}(t)]^1[\eta_{5u}]^1} = \frac{1 \cdot 0{,}0009}{0{,}0841} = 0{,}0011$$

Dengan cara yang sama, lakukan perhitungan probabilitas untuk semua pelabuhan yang dapat dikunjungi oleh semut. Kemudian, pilih suatu bilangan random $r = (0,1)$, contohnya 0,5.

Iterasi 1: $p_{53}^{1}(1) = 0{,}6017 > r$

Pada iterasi ke-1 didapat bahwa penjumlahan probabilitas pertama lebih besar dari r. Sehingga, pelabuhan yang akan dikunjungi selanjutnya adalah pelabuhan 3. Masukkan pelabuhan 3 dalam tabu list semut 1.

Lakukan perhitungan probabilitas sampai menemukan pelabuhan ke-8 untuk depot-1. Ulangi langkah 3 untuk mencari pelabuhan yang akan dikunjungi dari depot-depot lainnya.

Langkah-langkah untuk menentukan pelabuhan yang akan dikunjungi dapat dituliskan secara umum sesuai algoritma berikut.

```
% menentukan pelabuhan yang akan dikunjungi
for i= 1 to n do
    for k = 1 to D do
        Simpan pelabuhan awal(k) sebagai depot dalam tabu list;
        Tentukan u = random(batas bawah, batas atas)
        for m = 1 to u do
            Hitung probabilitas semut(i) memililih pelabuhan yang akan
            dikunjungi selanjutnya menggunakan persamaan (2.17)
            Ambil suatu bilangan random r dari interval (0,1)
            for s = 1 to length(probabilitas) do
                if (r<sum(probabilitas(1:m)))
                    pelabuhan selanjutnya adalah m;
                    break
                end
            end
        end
    end
end
```

Langkah 4: Menghitung jarak rute yang dihasilkan semut

Setelah tabu list penuh, dihitung jarak rute yang dihasilkan oleh semut. Misalkan rute yang dihasilkan semut adalah 1-5-3-6-8-22-37-45-44-50-41-1-2-7-9-10-13-12-15-19-21-14-16-2-17-18-35-33-29-30-36-24-23-26-17-34-4-20-39-32-31-28-27-38-34-51-42-47-48-49-46-40-11-25-43-51.

Depot-1: 1-5-3-6-8-22-37-45-44-50-41-1

Jarak rute depot-1 $= 556 + 219 + 828 + 533 + 352 + 1110 + 620 + 132 + 230 + 509 + 1478 = 6567$

Depot-2: 2-7-9-10-13-12-15-19-21-14-16-2

Jarak rute depot-2 $= 610 + 195 + 42 + 409 + 79 + 132 + 1446 + 1200 + 369 + 704 + 1038 = 6224$

Depot-17: 17-18-35-33-29-30-36-24-23-26-17

Jarak rute depot-17 $= 749 + 541 + 452 + 398 + 727 + 194 + 602 + 155 + 448 + 583 = 4849$

Depot-34: 34-4-20-39-32-31-28-27-38-34

Jarak rute depot-34 $= 1525 + 101 + 414 + 605 + 259 + 291 + 83 + 1015 + 740 = 5033$

Depot-51: 51-42-47-48-49-46-40-11-25-43-51

Jarak rute depot-51 $= 321 + 211 + 483 + 210 + 241 + 669 + 2367 + 1326 + 1705 + 624 = 8157$

Total jarak yang dihasilkan oleh semut =

$6567 + 6224 + 4849 + 5033 + 8157 = 30830$

Secara umum, jarak yang dihasilkan oleh n semut seperti pada langkah 4 dapat dilakukan dengan algoritma berikut.

```
% menghitung jarak yang dihasilkan oleh semut
for i to n do
    L(i) = 0;
    for j to N-D-1 do
        if (sisi (j,j+1) bukan anggota depot)
            L(i) = L(i) + d(j,j+1);
        end
    end
end
```

Langkah 5: Menghitung penambahan feromon pada setiap sisi (i, j) yang dilalui semut

Setiap sisi (i, j) pada tur akan diberi tambahan feromon, kecuali sisi yang merupakan depot dengan depot.

$$\Delta\tau_{ij}^{1} = \frac{Q}{L_1} = \frac{10000}{30830} = 0{,}3244$$

Karena hanya terdapat satu semut yang digunakan, maka

$$\Delta\tau_{ij} = \sum_{k=1}^{1} \Delta\tau_{ij}^{k} = 0{,}3244.$$

Sehingga, didapat matriks penambahan feromon

	1	2	3	4	5	6	7	8	⋯	45	46	47	48	49	50	51
1	0	0	0	0	0,324	0	0	0	⋯	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0,324	0	⋯	0	0	0	0	0	0	0
3	0	0	0	0	0	0,324	0	0	⋯	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	⋯	0	0	0	0	0	0	0
5	0	0	0,324	0	0	0	0	0	⋯	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0,324	⋯	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	⋯	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0		⋯	0	0	0	0	0	0	0
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
45	0	0	0	0	0	0	0	0	⋯	0	0	0	0	0	0	0
46	0	0	0	0	0	0	0	0	⋯	0	0	0	0	0	0	0
47	0	0	0	0	0	0	0	0	⋯	0	0	0	0,324	0	0	0
48	0	0	0	0	0	0	0	0	⋯	0	0	0	0	0,324	0	0
49	0	0	0	0	0	0	0	0	⋯	0	0,324	0	0	0	0	0
50	0	0	0	0	0	0	0	0	⋯	0	0	0	0	0	0	0
51	0	0	0	0	0	0	0	0	⋯	0	0	0	0	0	0	0

Langkah penambahan feromon secara umum dapat dituliskan dalam algoritma berikut.

```
% menghitung penambahan feromon
for i to n do
Δτ_{jj+1} = 0
    for j to N-D-1 do
        if (sisi (j,j+1) bagian dari tur)
            Hitung Δτ^i_{jj+1} penambahan feromon menggunakan persamaan(2.19)
        end
    end
    Hitung Δτ_{jj+1} = Δτ_{jj+1} + Δτ^i_{jj+1}
end
```

Langkah 6: Update trail

Update trail meliputi penguapan feromon dan penambahan feromon dihitung berdasarkan persamaan (2.18)

$$\tau_{ij} = (1 - 0.5) \times \begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 11 & \cdots & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \cdots & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 11 & 1 & 1 & 1 & 1 & \cdots & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \cdots & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 11 & 1 & 1 & 1 & 1 & 1 & \cdots & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \cdots & 1 & 1 & 1 & 1 & 1 & 1 \\
11 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \cdots & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \cdots & 1 & 1 & 1 & 1 & 1 & 1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \cdots & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \cdots & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \cdots & 1 & 1 & 11 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \cdots & 1 & 1 & 1 & 11 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \cdots & 11 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \cdots & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \cdots & 1 & 1 & 1 & 1 & 1 & 1
\end{pmatrix}$$

$$+\begin{pmatrix}
0 & 0 & 0 & 0 & 0{,}324 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0{,}324 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0{,}324 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0{,}324 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0{,}324 & \cdots & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & & \cdots & 0 & 0 & 0 & 0 & 0 & 0\\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0{,}324 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0{,}324 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0{,}324 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

$$=\begin{pmatrix}
0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}824 & 0{,}5 & 0{,}5 & 5{,}5 & \cdots & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5\\
0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}824 & 0{,}5 & \cdots & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5\\
0{,}5 & 0{,}5 & 0{,}5 & 5{,}5 & 0{,}5 & 0{,}824 & 0{,}5 & 0{,}5 & \cdots & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5\\
0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & \cdots & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5\\
0{,}5 & 0{,}5 & 5{,}824 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & \cdots & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5\\
0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}824 & \cdots & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5\\
5{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & \cdots & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5\\
0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & \cdots & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5\\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\
0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & \cdots & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5\\
0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & \cdots & 0{,}5 & 0{,}5 & 5{,}824 & 0{,}5 & 0{,}5 & 0{,}5\\
0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & \cdots & 0{,}5 & 0{,}5 & 0{,}5 & 5{,}824 & 0{,}5 & 0{,}5\\
0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & \cdots & 5{,}824 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5\\
0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & \cdots & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5\\
0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & \cdots & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5
\end{pmatrix}$$

Langkah 6 dapat dituliskan secara umum menggunakan algoritma berikut.

```
% update feromon trail
Hitung feromon mengunakan persamaan (2.18)
```

Langkah 7: Kosongkan kembali tabu list. Ulangi langkah 3-6 sampai memenuhi stopping condition.

Langkah-langkah pencarian solusi global menggunakan ACO dapat diuraikan sebagai berikut.

```
% Inisialisasi tur
for i = 1 to n do
   Masukkan depot ke-1 dalam tabu list semut(i);
end
% menghitung visibilitas
visibilitas = 1/jarak antar pelabuhan;
repeat
   % menentukan pelabuhan yang akan dikunjungi
   for i= 1 to n do
       for k = 1 to D do
```

```
            Simpan pelabuhan awal(k) sebagai depot dalam tabu list;
            Tentukan u = random(batas bawah, batas atas)
            for m = 1 to u do
                Hitung probabilitas semut(i) memililih pelabuhan yang
                akan dikunjungi selanjutnya menggunakan persamaan (2.17)
                Ambil suatu bilangan random r dari interval (0,1)
                for s = 1 to length(probabilitas) do
                    if (r<sum(probabilitas(1:m)))
                        pelabuhan selanjutnya adalah m;
                        break
                    end
                end
            end
        end
    end
    % menghitung jarak yang dihasilkan oleh semut
    for i to n do
        L(i) = 0;
        for j to N-D-1 do
            if (sisi (j,j+1) bukan anggota depot)
                L(i) = L(i) + d(j,j+1);
            end
        end
    end
    % menghitung penambahan feromon
    for i to n do
    Δτ_{jj+1} = 0
        for j to N-D-1 do
            if (sisi (j,j+1) bagian dari tur)
                Hitung Δτ^i_{jj+1} penambahan feromon menggunakan
                persamaan(2.19)
            end
        end
        Hitung Δτ_{jj+1} = Δτ_{jj+1} + Δτ^i_{jj+1}
    end
    % update feromon trail
    Hitung feromon mengunakan persamaan (2.18)
    Kosongkan kembali tabu list
until (memenuhi stopping condition)
```

4.2 Uji Validitas Program

Uji validitas yang digunakan pada penelitian ini adalah uji validitas secara empiris, yaitu membandingkan hasil yang diperoleh dari perhitungan manual dengan perhitungan program. Pada perhitungan manual, diambil 2 kunang-kunang dengan 1 pergerakan untuk masing-masing kunang-kunang dan 1 semut. Parameter yang digunakan antara lain $\gamma = 0{,}005, \alpha = 1, \beta = 1, \rho = 0{,}5, Q = 10000$.

Hasil perhitungan manual didapat kunang-kunang yang dipilih untuk iterasi selanjutnya adalah kunang-kunang-1 dengan total rute 52219 dan kunang-kunang-3 dengan total rute 54628 sebagai berikut.

Kunang-kunang-1

1	38	35	21	29	10	16	8	3	28	9	1	2	11
6	48	43	22	25	36	30	4	2	17	13	31	23	32
15	12	20	44	5	17	34	41	49	33	47	46	42	24
26	50	27	34	51	40	37	7	18	14	39	45	19	51

Kunang-kunang-3

1	38	35	21	29	49	41	5	44	20	1	2	12	15
32	23	31	13	4	30	36	2	17	25	22	43	48	6
11	9	28	3	8	17	34	16	10	33	47	46	42	24
26	50	34	51	27	40	37	7	18	14	39	45	19	51

Pada program, hasil yang didapat ditunjukkan pada Gambar 4.4 berikut.

```
of =

    52219    56165    54628    58762

k =

  Columns 1 through 14

     1    38    35    21    29    10    16     8     3    28     9     1     2    11
     1    38    35    21    29    49    41     5    44    20     1     2    12    15

  Columns 15 through 28

     6    48    43    22    25    36    30     4     2    17    13    31    23    32
    32    23    31    13     4    30    36     2    17    25    22    43    48     6

  Columns 29 through 42

    15    12    20    44     5    17    34    41    49    33    47    46    42    24
    11     9    28     3     8    17    34    16    10    33    47    46    42    24

  Columns 43 through 56

    26    50    27    34    51    40    37     7    18    14    39    45    19    51
    26    50    34    51    27    40    37     7    18    14    39    45    19    51
```

Gambar 4.4 Hasil program untuk populasi kunang-kunang selanjutnya dan total jarak yang dihasilkan

Setelah memenuhi banyak iterasi yang diinginkan, hasil solusi FA dinormalisasi agar tidak terdapat solusi ganda. Hasil normalisasi solusi yang didapat pada perhitungan manual pada Langkah 1 subbab 4.1.2 adalah sebagai berikut.

Normalisasi solusi

1	8	22	37	45	44	50	41	19	21	7	1	2	9
10	13	12	15	6	14	16	18	2	17	35	33	29	30
36	24	23	26	17	34	5	3	4	20	39	32	31	28
27	38	34	51	42	47	48	49	46	40	11	25	43	51

Normalisasi solusi yang dihasilkan oleh program ditunjukkan seperti Gambar 4.5 berikut.

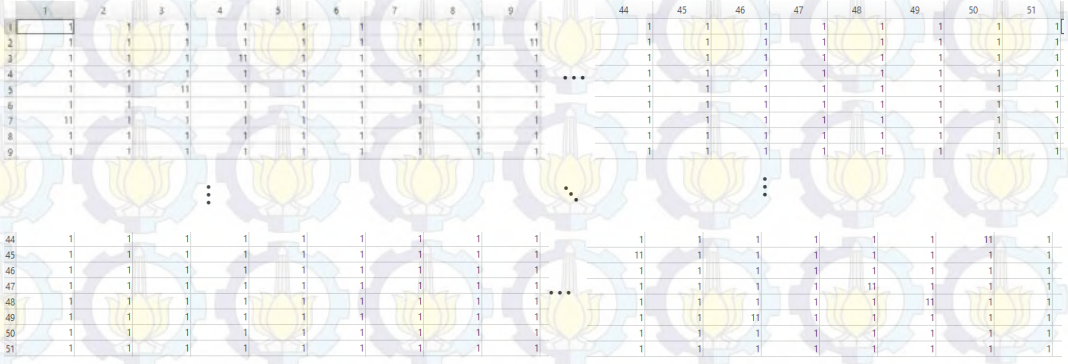
```
norm_solusi =

  Columns 1 through 14

    1     8    22    37    45    44    50    41    19    21     7     1     2     9

  Columns 15 through 28

   10    13    12    15     6    14    16    18     2    17    35    33    29    30

  Columns 29 through 42

   36    24    23    26    17    34     5     3     4    20    39    32    31    28

  Columns 43 through 56

   27    38    34    51    42    47    48    49    46    40    11    25    43    51
```

Gambar 4.5 Hasil program normalisasi solusi

Setelah dinormalisasi, feromon ditambahkan pada sisi yang termasuk dalam rute. Pada perhitungan manual Langkah 2 subbab 4.1.2, inisialisasi feromon yang didapatkan berupa matriks berukuran 51x51 berikut.

	1	2	3	4	5	6	7	8	9	⋯	44	45	46	47	48	49	50	51
1	1	1	1	1	1	1	1	11	1	⋯	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	11	⋯	1	1	1	1	1	1	1	1
3	1	1	1	11	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
5	1	1	11	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
7	11	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
9	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
44	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	11	1
45	1	1	1	1	1	1	1	1	1	⋯	11	1	1	1	1	1	1	1
46	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
47	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	11	1	1	1
48	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	11	1	1
49	1	1	1	1	1	1	1	1	1	⋯	1	1	11	1	1	1	1	1
50	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
51	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1

Inisialisasi feromon setelah dinormalisasi pada program ditunjukkan seperti Gambar 4.6 berikut.



Gambar 4.6 Hasil program inisialisasi feromon

Perhitungan probabilitas Langkah 3 pada subbab 4.1.3 dengan pelabuhan-1 sebagai depot dihitung untuk empat pelabuhan pertama saja. Hasil yang didapatkan antara lain $p_{13}^{1}(1) = 0{,}0127, p_{14}^{1}(1) = 0{,}0128, p_{15}^{1}(1) = 0{,}0176,$ dan $p_{16}^{1}(1) = 0{,}0142$. Dengan perlakuan yang sama, hasil yang didapatkan pada program adalah sebagai berikut.

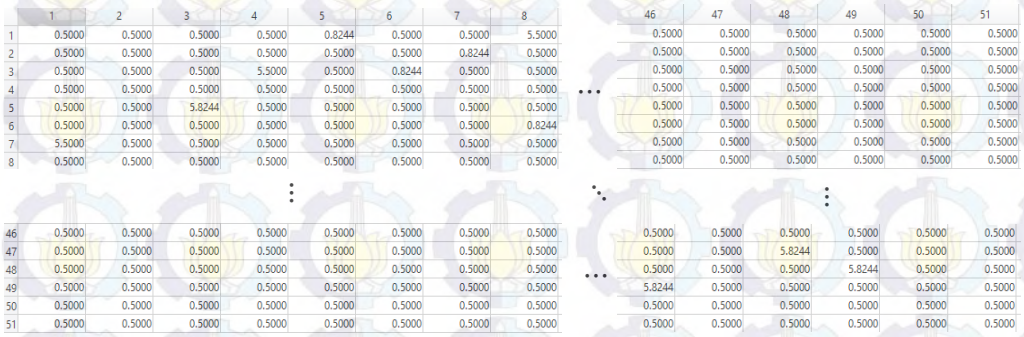
```
prob =

  Columns 1 through 8

    0.0127    0.0128    0.0176    0.0142    0.0183    0.4843    0.0145    0.0153

  Columns 9 through 16

    0.0095    0.0111    0.0122    0.0134    0.0131    0.0416    0.0089    0.0097

  Columns 17 through 24

    0.0088    0.0233    0.0269    0.0210    0.0194    0.0151    0.0128    0.0078

  Columns 25 through 32

    0.0088    0.0068    0.0085    0.0101    0.0080    0.0086    0.0125    0.0102

  Columns 33 through 40

    0.0073    0.0067    0.0079    0.0059    0.0066    0.0052    0.0045    0.0056

  Columns 41 through 46

    0.0052    0.0050    0.0052    0.0062    0.0055    0.0055
```

Gambar 4.7 Hasil program perhitungan probabilitas untuk pelabuhan 1

Langkah 6 pada subbab 4.1.3 menunjukkan hasil perhitungan manual *update trail* dari metode ACO seperti berikut.

$$\begin{pmatrix}
0.5 & 0.5 & 0.5 & 0.5 & 0.824 & 0.5 & 0.5 & 5.5 & \cdots & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\
0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.824 & 0.5 & \cdots & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\
0.5 & 0.5 & 0.5 & 5.5 & 0.5 & 0.824 & 0.5 & 0.5 & \cdots & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\
0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & \cdots & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\
0.5 & 0.5 & 5.824 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & \cdots & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\
0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.824 & \cdots & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\
5.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & \cdots & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\
0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & \cdots & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & \cdots & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\
0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & \cdots & 0.5 & 0.5 & 5.824 & 0.5 & 0.5 & 0.5 \\
0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & \cdots & 0.5 & 0.5 & 0.5 & 5.824 & 0.5 & 0.5 \\
0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & \cdots & 5.824 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\
0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & \cdots & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\
0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & \cdots & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5
\end{pmatrix}$$

Hasil *update trail* pada program ditunjukkan pada Gambar 4.8 berikut.



Gambar 4.8 Hasil program *update trail*

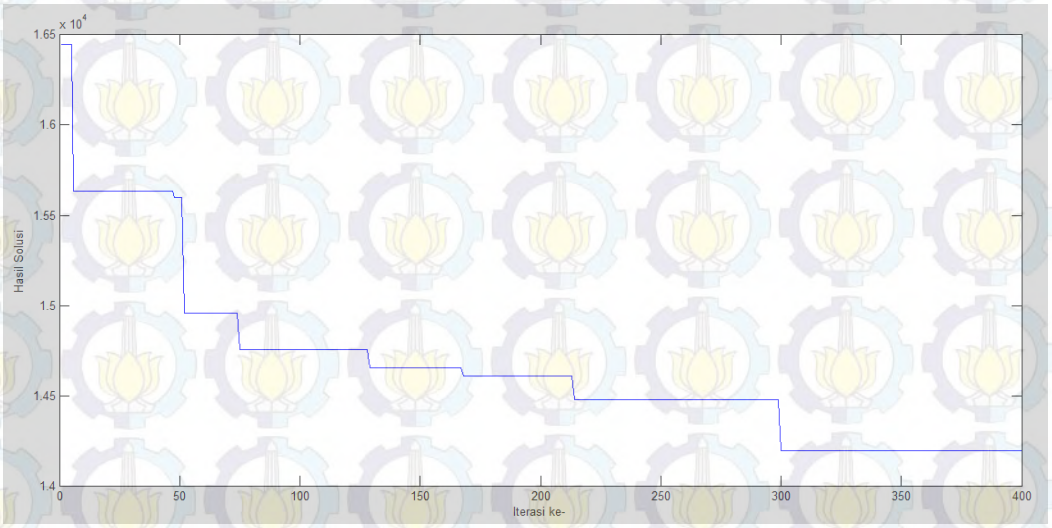
Berdasarkan perbandingan perhitungan manual dengan hasil program, dapat disimpulkan program *hybrid* FA-ACO telah berjalan sesuai dengan perhitungan manual.

4.3 Hasil Uji Coba dan Analisis

Uji coba dilakukan dengan menggunakan aplikasi Matlab yang dijalankan pada perangkat dengan spesifikasi Processor Intel(R) Core(TM) i7 CPU 2.20 GHz dan 4.00 GB RAM. Pada penelitian ini disajikan hasil uji coba dari dua metode, yaitu ACO dan *hybrid* FA-ACO. Pada tahap uji coba dilakukan penentuan banyaknya semut dan iterasi untuk metode ACO. Untuk metode *hybrid* FA-ACO, dilakukan penentuan banyak kunang-kunang, semut, dan iterasi. Tahap selanjutnya adalah membandingkan hasil uji coba metode *hybrid* FA-ACO dengan metode ACO dalam menyelesaikan studi kasus.

4.3.1 Uji Coba Metode ACO

Tahap pertama dalam uji coba metode ACO adalah menentukan banyak iterasi minimum yang diperlukan untuk mencapai solusi yang konvergen. Parameter yang digunakan pada uji coba ini antara lain $\alpha = 1, \beta = 5, \rho = 0{,}5, Q = 100$ dan semut sebanyak jumlah kota (Dorigo, 1996). Sehingga, pada uji coba ini semut yang digunakan sebanyak 51. Metode ACO dijalankan dengan 5000 iterasi (Dorigo, 1996). Tetapi, jika setelah 100 iterasi berturut-turut tidak terdapat peningkatan solusi, maka iterasi dihentikan (Liu, dkk, 2009). Dengan mengambil iterasi terbanyak yang dibutuhkan dari 10 kali percobaan, didapat bahwa iterasi minimal yang diperlukan ACO untuk mencapai konvergensi adalah 300. Banyak iterasi ini akan digunakan pada uji coba selanjutnya. Hasil uji coba penentuan banyak iterasi minimum ditunjukkan pada Gambar 4.9.



Gambar 4.9 Uji coba penentuan banyak iterasi minimum

Tahap kedua adalah menentukan parameter α dan β yang mempengaruhi probabilitas penentuan pelabuhan yang akan dikunjungi. Nilai yang akan diujicobakan adalah $\alpha \in \{0{,}5{,}1{,}2\}$ dan $\beta \in \{1{,}2{,}5\}$ (Dorigo, 1996). Pada uji coba ini, pengaturan *default* yang digunakan antara lain $\rho = 0{,}5, Q = 100$, semut sebanyak 51 dan banyak iterasi adalah 300.

Tabel 4.2 Penentuan nilai α dan β

α	β	Hasil	Waktu
0,5	1,0	21038	81,4154
	2,0	17169	80,0412
	5,0	14659	83,1577
1,0	1,0	19088	80,0881
	2,0	16072	79,0493
	5,0	14231	81,7190
2,0	1,0	18872	79,8144
	2,0	16532	77,6645
	5,0	15361	81,1222

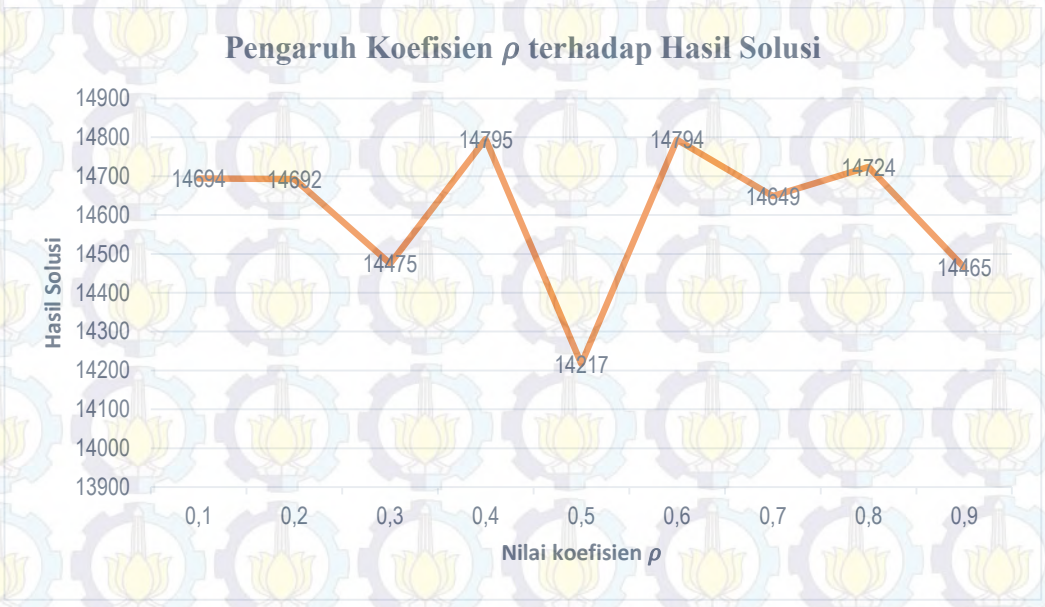
Berdasarkan Tabel 4.2, hasil terbaik diperoleh jika $\alpha = 1$ dan $\beta = 5$. Hal ini berarti pada studi kasus ini, visibilitas memiliki prioritas yang lebih besar dibandingkan dengan feromon. Dari 3 nilai α yang diujicobakan, hasil optimum didapat ketika $\beta = 5$. Waktu komputasi terbaik diperoleh ketika $\alpha = 2$ dan $\beta = 5$. Dengan mempertimbangkan solusi yang dihasilkan dan waktu komputasi, maka pada studi kasus ini, dipilih $\alpha = 1$ dan $\beta = 5$.

Tahap ketiga uji coba ACO adalah mencari parameter ρ dimana $(1 - \rho)$ menyatakan penguapan feromon. Nilai parameter ρ yang diujicobakan memiliki interval $(0,1)$. Pada uji coba ini, pengaturan *default* yang digunakan antara lain $\alpha = 1,\ \beta = 5, Q = 100$, semut sebanyak 51 dan banyak iterasi adalah 300.

Tabel 4.3 Nilai koefisien ρ

ρ	Hasil	Waktu
0,1	14694	82,37993
0,2	14692	82,47564
0,3	14475	82,2798
0,4	14795	82,13375
0,5	14217	82,33667
0,6	14794	82,56143
0,7	14649	82,74688
0,8	14724	82,22003
0,9	14465	82,59745

Dari Tabel 4.3, hubungan antara nilai koefisien ρ dan hasil solusi dapat ditunjukkan pada Gambar 4.10 berikut.



Gambar 4.10 Grafik hubungan koefisien ρ terhadap hasil solusi

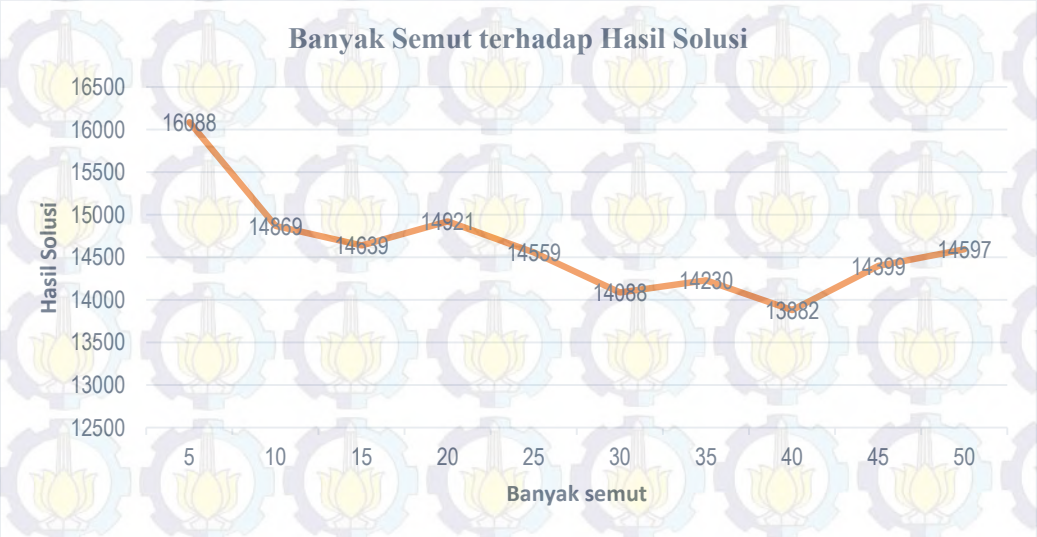
Gambar 4.10 menunjukkan pengaruh koefisien ρ terhadap studi kasus. Dari sembilan nilai ρ, hasil terbaik diperoleh ketika $\rho = 0{,}5$. Berdasarkan hasil uji coba pada Tabel 4.3 waktu komputasi terbaik 82,13375 detik diperoleh ketika $\rho = 0{,}4$. Pada uji coba tersebut, untuk semua nilai ρ yang diujicobakan tidak terdapat perbedaan waktu komputasi yang cukup signifikan. Sehingga, pada studi kasus ini, dipilih $\rho = 0{,}5$.

Tahap selanjutnya adalah menentukan banyak semut yang optimal. Nilai parameter $\alpha = 1$, $\beta = 5$, $\rho = 0{,}5$, dan $Q = 100$. Banyak n semut yang diujicobakan adalah $n \in \{5,10,15,20,25,30,35,40,45,50\}$ dengan iterasi sebanyak 300 kali. Hasil uji coba penentuan banyak semut untuk menjalankan ACO pada studi kasus ditunjukkan pada Tabel 4.4 berikut.

Tabel 4.4 Penetuan banyak semut optimal ACO

Banyak semut	Hasil	Waktu
5	16088	8,169784
10	14869	16,40132
15	14639	24,41994
20	14921	32,88892
25	14559	41,32483
30	14088	50,2965
35	14230	57,64817
40	13882	62,98049
45	14399	73,93611
50	14597	82,45433

Hasil uji coba yang didapat pada Tabel 4.4 dapat direpresentasikan dengan grafik untuk mengetahui hubungan banyak semut dengan hasil solusi seperti pada Gambar 4.11 berikut.



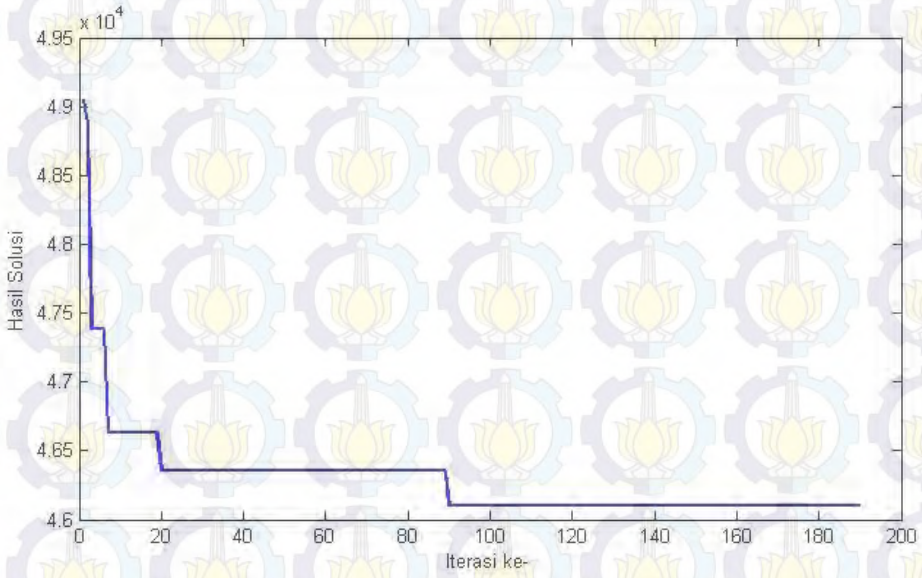
Gambar 4.11 Grafik hubungan banyak semut terhadap hasil solusi

Dari Gambar 4.11 terlihat bahwa banyak semut minimal yang dapat menghasilkan solusi optimum adalah 40 semut. Waktu komputasi yang diperlukan adalah 62,98049 detik.

4.3.2 Uji Coba Metode *Hybrid* FA-ACO

Uji coba metode *hybrid* FA-ACO ini meliputi penentuan nilai koefisien penyerapan cahaya γ, banyak kunang-kunang dan pergerakannya, banyak semut, dan banyak iterasi untuk pencarian lokal FA dan pencarian global FA. Pada setiap uji coba, hanya satu nilai saja yang diubah.

Tahap pertama dalam uji coba metode *hybrid* FA-ACO adalah menentukan banyak iterasi minimum FA yang diperlukan untuk mencapai solusi yang konvergen. Parameter yang digunakan untuk pencarian lokal FA antara lain $\gamma = 0{,}07$ dengan 15 kunang-kunang dan 11 pergerakan untuk tiap kunang-kunang (Jati dan Suyanto, 2010). Dengan mengambil iterasi terbanyak yang dibutuhkan dari 10 kali percobaan, didapat bahwa iterasi minimal yang diperlukan FA untuk mencapai konvergensi adalah 100. Banyak iterasi ini akan digunakan pada uji coba selanjutnya. Hasil uji coba penentuan banyak iterasi minimum ditunjukkan pada Gambar 4.12.



Gambar 4.12 Uji coba penentuan banyak iterasi minimum pencarian lokal FA

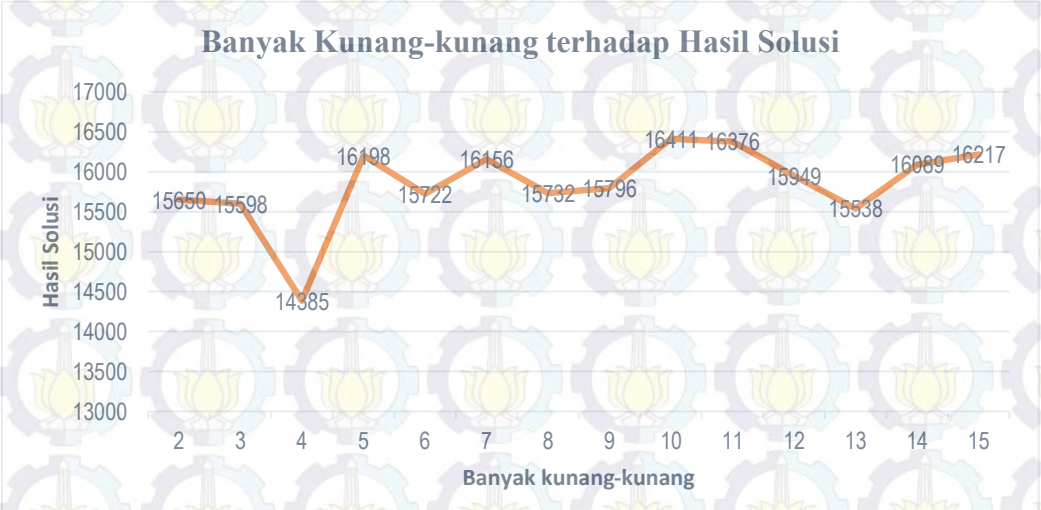
Tahap kedua adalah mencari banyak kunang-kunang optimal. Banyak kunang-kunang yang diujicobakan adalah 2 sampai 15 (Jati dan Suyanto, 2010). Pada uji coba ini, parameter yang digunakan untuk pencarian global ACO sesuai dengan hasil yang didapat pada uji coba ACO sebelumnya, yaitu $\alpha = 1, \beta = 5, \rho =$

0,5, $Q = 100$ dengan 40 semut dan 300 iterasi. Hasil uji coba penentuan banyak kunang-kunang untuk menjalankan *hybrid* FA-ACO pada studi kasus ditunjukkan pada Tabel 4.5.

Tabel 4.5 Penetuan banyak kunang-kunang optimal

Banyak kunang-kunang	Hasil	Waktu
2	15650	94,96176
3	15598	101,0158
4	14385	107,2189
5	16198	112,8496
6	15722	119,6184
7	16156	125,7431
8	15732	132,3029
9	15796	137,7738
10	16411	144,1912
11	16376	150,0479
12	15949	156,675
13	15538	162,2745
14	16089	167,6822
15	16217	173,826

Hasil uji coba yang didapat pada Tabel 4.5 dapat direpresentasikan dengan grafik untuk mengetahui hubungan banyak kunang-kunang dengan hasil solusi seperti pada Gambar 4.13 berikut.



Gambar 4.13 Hubungan banyak kunang-kunang dengan hasil solusi

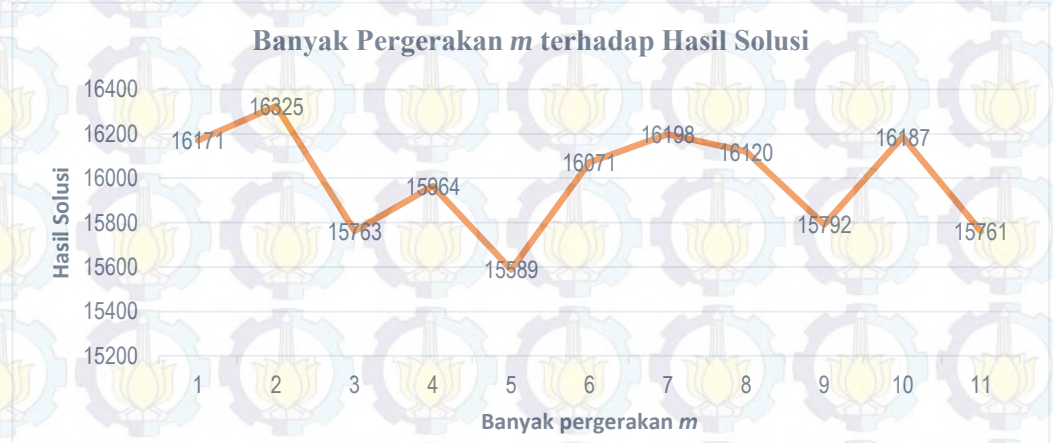
Dari Tabel 4.5 dan Gambar 4.13, didapat sebanyak 4 kunang-kunang untuk menghasilkan solusi terbaik dengan waktu komputasi 107.2189. Sehingga, pada studi kasus ini, banyak kunang-kunang yang digunakan adalah 4.

Tahap ketiga uji coba metode *hybrid* FA-ACO adalah menentukan banyak pergerakan untuk tiap kunang-kunang. Penentuan banyak pergerakan kunang-kunang diujicobakan dari nilai 1 hingga 11 (Jati dan Suyanto, 2010). Hasil uji coba penentuan banyak pergerakan masing-masing kunang-kunang untuk menjalankan *hybrid* FA-ACO pada studi kasus ditunjukkan pada Tabel 4.6.

Tabel 4.6 Penentuan banyak pergerakan m untuk tiap kunang-kunang

Banyak pergerakan	Hasil	Waktu
1	16171	87,38962
2	16325	89,61266
3	15763	91,3517
4	15964	93,58011
5	15589	95,88416
6	16071	97,91237
7	16198	100,1068
8	16120	101,2708
9	15792	103,419
10	16187	105,8723
11	15761	107,7854

Hasil yang telah diperoleh pada Tabel 4.6 direpresentasikan pada Gambar 4.14.



Gambar 4.14 Hubungan pergerakan tiap kunang-kunang dengan hasil solusi

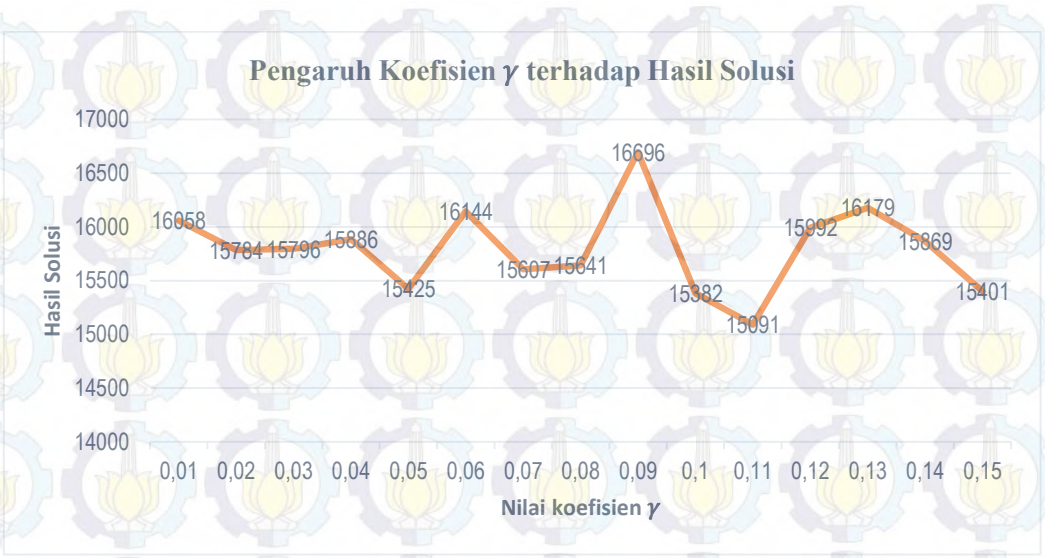
Tabel 4.6 dan Gambar 4.14 menunjukkan bahwa dengan 4 kunang-kunang diperlukan pergerakan untuk masing-masing kunang-kunang sebanyak 5. Waktu komputasi yang dibutuhkan adalah 95,88416 detik.

Tahap keempat adalah menentukan koefisien penyerapan cahaya γ. Jati dan Suyanto (2010) meneliti bahwa γ yang menghasilkan nilai terbaik terletak pada interval $[0{,}01{,}0{,}15]$. Uji coba ini menggunakan parameter hasil dari uji coba sebelumnya. Tabel 4.7 menunjukkan pengaruh nilai koefisien γ terhadap hasil solusi.

Tabel 4.7 Nilai koefisien γ

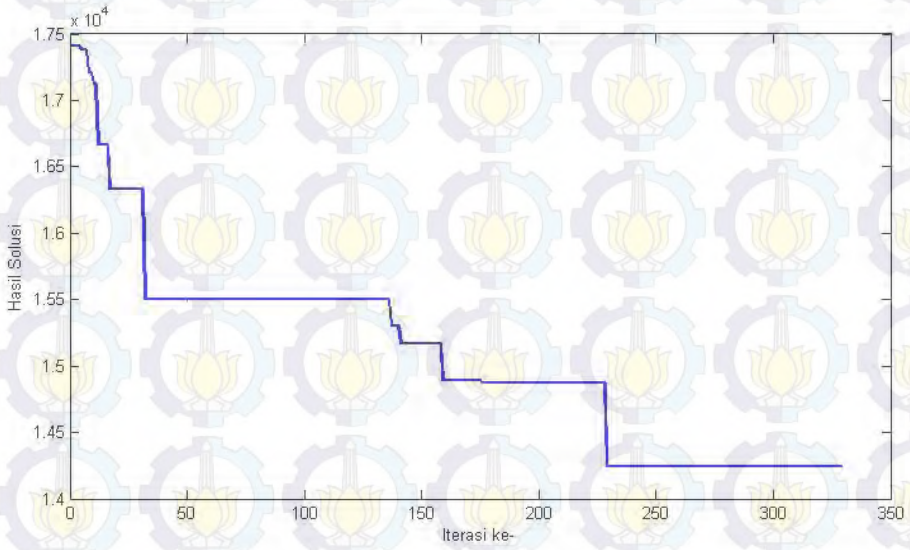
γ	Hasil	Waktu
0,01	16058	96,07631
0,02	15784	96,34295
0,03	15796	95,94018
0,04	15886	96,08202
0,05	15425	95,62379
0,06	16144	96,31628
0,07	15607	96,66601
0,08	15641	96,53021
0,09	16696	95,3499
0,10	15382	96,1339
0,11	15091	97,71056
0,12	15992	96,88888
0,13	16179	96,78503
0,14	15869	96,59213
0,15	15401	96,76168

Hasil uji coba penentuan nilai koefisien penyerapan cahaya γ pada Tabel 4.7 dapat direpresentasikan dengan grafik seperti pada Gambar 4.15. Dari Tabel 4.7 dan Gambar 4.15, terlihat bahwa hasil terbaik diperoleh ketika $\gamma = 0{,}11$ dengan waktu komputasi 97,71056 detik. Dengan demikian, pada studi kasus ini dipilih $\gamma = 0{,}11$.



Gambar 4.15 Hubungan koefisien penyerapan cahaya γ dengan hasil solusi

Tahap kelima adalah menentukan iterasi minimum yang diperlukan pencarian global ACO. Dengan mengambil iterasi terbanyak yang dibutuhkan dari 10 kali percobaan, didapat bahwa iterasi minimal yang diperlukan pencarian solusi global ACO untuk mencapai konvergensi adalah 229. Hasil uji coba penentuan banyak iterasi minimum ditunjukkan pada Gambar 4.16.



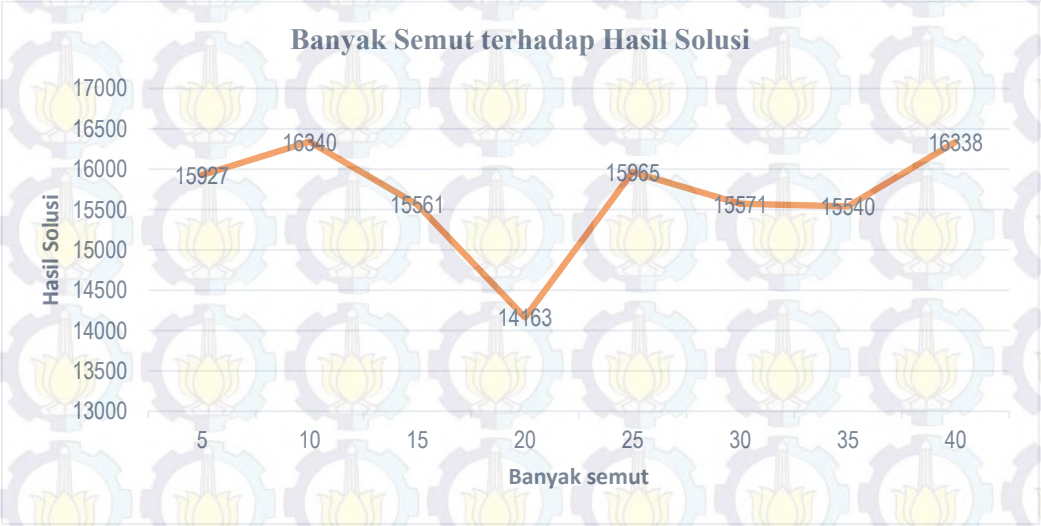
Gambar 4.16 Uji coba penentuan banyak iterasi minimum pencarian global ACO

Tahap selanjutnya adalah uji coba penentuan banyak semut untuk pencarian global ACO dilakukan untuk banyak semut dengan kelipatan 5 mulai dari 5 sampai 40. Uji coba hanya dibatasi sampai 40 semut saja karena dari hasil uji coba metode ACO sebelumnya banyak semut optimal adalah 40.

Tabel 4.8 Penetuan banyak semut optimal untuk pencarian global ACO

Banyak semut	Hasil	Waktu
5	15927	20,3944
10	16340	28,21828
15	15561	36,42925
20	14163	44,30809
25	15965	52,59493
30	15571	60,1798
35	15540	68,16099
40	16338	76,49842

Hasil uji coba penentuan banyak semut untuk menjalankan *hybrid* FA-ACO pada studi kasus ditunjukkan pada Tabel 4.8. Hasil uji coba yang didapat pada Tabel 4.8 dapat direpresentasikan dengan grafik untuk mengetahui hubungan banyak semut dengan hasil solusi seperti pada Gambar 4.17 berikut.



Gambar 4.17 Grafik hubungan banyak semut terhadap hasil solusi

Dari Gambar 4.17 terlihat bahwa banyak semut minimal yang dapat menghasilkan solusi optimum adalah 20 semut. Waktu komputasi yang diperlukan adalah 44.30809 detik.

4.4 Perbandingan Metode *Hybrid* FA-ACO dengan ACO

Tahap selanjutnya adalah membandingkan metode ACO dengan *hybrid* FA-ACO, baik dari hasil terbaik yang dapat ditemukan dan waktu komputasinya. Berdasarkan beberapa uji coba di atas, metode ACO yang terbaik diinisialisasi sebagai berikut.

(1) Parameter standar yang digunakan pada uji coba adalah $\alpha = 1$, $\beta = 5$, $\rho = 0{,}5$, dan $Q = 100$.
(2) Banyak semut yang diperlukan untuk menghasilkan solusi optimum studi kasus adalah 40.
(3) Iterasi minimal yang diperlukan untuk menghasilkan solusi optimum studi kasus sebanyak 300 iterasi.

Hasil uji coba metode ACO tersebut digunakan sebagai acuan untuk melakukan uji coba selanjutnya, yaitu uji coba metode *hybrid* FA-ACO. Berdasarkan beberapa uji coba di atas, didapatkan inisialisasi metode *hybrid* FA-ACO yang terbaik adalah sebagai berikut.

(1) Parameter standar yang digunakan dalam pencarian lokal FA adalah $\gamma = 0{,}11$.
(2) Banyak kunang-kunang minimal dan pergerakannya untuk mencapai solusi optimum adalah 4 dan 5, berturut-turut.
(3) Banyak semut yang diperlukan dalam pencarian global ACO adalah 20.
(4) Iterasi yang diperlukan untuk pencarian lokal FA dan pencarian global ACO adalah sebanyak 100 dan 229, berturut-turut.

Perbandingan metode ACO dengan *hybrid* FA-ACO dilakukan dengan menjalankan kedua metode tersebut sebanyak 50 kali percobaan. Dari 50 kali percobaan, akan dibandingkan beberapa hal berikut.

(1) Solusi terbaik yang berhasil ditemukan oleh masing-masing metode.
(2) Rata-rata solusi dari 50 kali percobaan. Rata-rata ini bertujuan untuk menyimpulkan suatu metode mudah terjebak lokal optimum atau tidak.

(3) Rata-rata waktu komputasi dari 50 kali percobaan.

(4) Waktu konvergensi untuk mencapai solusi optimum.

Tabel 4.9 Hasil solusi metode ACO untuk 50 percobaan

	Hasil	Waktu		Hasil	Waktu
1	14805	63,04299	26	14593	60,68913
2	14610	62,67013	27	14413	60,04852
3	14257	62,19059	28	14621	60,03004
4	14706	62,75386	29	14336	60,02923
5	14551	61,83609	30	14235	60,44221
6	14738	62,45683	31	14218	59,95182
7	14107	62,41257	32	14664	60,65964
8	14370	62,44707	33	14348	60,47776
9	13882	62,98049	34	13882	60,2617
10	14864	62,6825	35	14610	60,11612
11	14693	60,49879	36	14793	60,15227
12	14505	60,4158	37	14497	60,18529
13	14856	60,05361	38	14216	60,30743
14	14564	60,3746	39	14594	60,41146
15	14149	60,7041	40	14695	60,30444
16	14197	60,38985	41	14596	60,08112
17	14277	60,40417	42	14516	60,15991
18	14468	60,51613	43	14692	59,93025
19	14600	60,3378	44	14318	59,9833
20	14318	60,3485	45	14408	60,11049
21	14025	60,33186	46	14306	59,86428
22	14700	60,23336	47	14588	60,6241
23	14818	60,31501	48	14055	60,99906
24	14580	60,40328	49	14404	61,01977
25	14487	60,39761	50	14437	60,8703

Hasil terbaik yang didapat pada metode ACO untuk 50 kali percobaan adalah 13882 km. Rata-rata solusi dari 50 percobaan untuk metode ACO adalah 14463,24 km dengan rata-rata waktu komputasi 60,77814 detik.

Tabel 4.10 Hasil solusi metode *hybrid* FA-ACO untuk 50 percobaan

	Hasil	Waktu		Hasil	Waktu
1	14664	35,91105	26	13828	33,28346
2	14660	35,86679	27	14482	36,28486
3	14290	35,75269	28	14242	36,09445
4	14540	35,79397	29	13952	35,97861
5	14489	35,86069	30	14648	35,84072
6	14635	35,7304	31	14043	35,93092
7	14497	35,87138	32	13967	35,19612
8	14573	35,93507	33	15022	35,98057
9	14421	35,79084	34	13832	31,3275
10	14424	35,53941	35	14356	35,68823
11	14349	35,67664	36	14667	36,00629
12	14581	35,62591	37	13975	33,39738
13	14621	35,6674	38	14641	35,85087
14	13956	33,3102	39	14142	36,04381
15	14417	35,68784	40	14442	36,07409
16	14565	35,54677	41	14475	36,00288
17	14540	35,59742	42	14416	35,81041
18	14554	35,78967	43	14781	35,94123
19	14568	36,16137	44	14595	36,11852
20	14474	36,15302	45	13999	33,35354
21	14548	36,03102	46	14325	36,21462
22	14395	36,08348	47	14493	36,17546
23	14323	36,57492	48	13937	36,20821
24	14326	36,37064	49	14489	36,01579
25	14504	36,24745	50	14351	35,94776

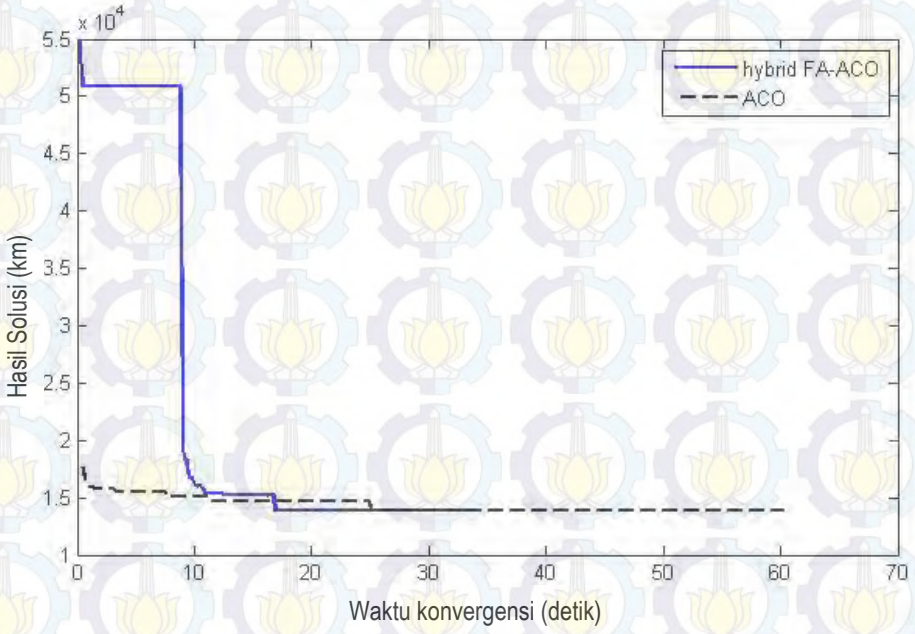
Hasil terbaik yang didapat pada metode *hybrid* FA-ACO untuk 50 kali percobaan adalah 13828 km. Rata-rata solusi dari 50 percobaan untuk metode *hybrid* FA-ACO adalah 14400,28 km dengan rata-rata waktu komputasi 36,01579 detik. Perbandingan metode *hybrid* FA-ACO dengan ACO disajikan pada Tabel 4.11 berikut.

Tabel 4.11 Perbandingan metode hybrid FA-ACO dengan metode ACO untuk 50 percobaan

Metode	Solusi terbaik yang didapat	Rata-rata solusi	Rata-rata waktu komputasi
Hybrid FA-ACO	13828	14400,28	36,01579
ACO	13882	14463,24	60,77814

Dari Tabel 4.11 dapat disimpulkan bahwa metode *hybrid* FA-ACO lebih baik dalam mencapai solusi optimum dengan rata-rata waktu komputasi 24,76235 detik lebih cepat daripada metode ACO. Dengan kata lain, rata-rata waktu komputasi metode *hybrid* FA-ACO lebih cepat 40,74% daripada rata-rata waktu komputasi ACO.

Perbandingan hasil dan waktu komputasi metode *hybrid* FA-ACO dengan metode ACO untuk mencapai solusi terbaik ditunjukkan pada Gambar 4.18 berikut.

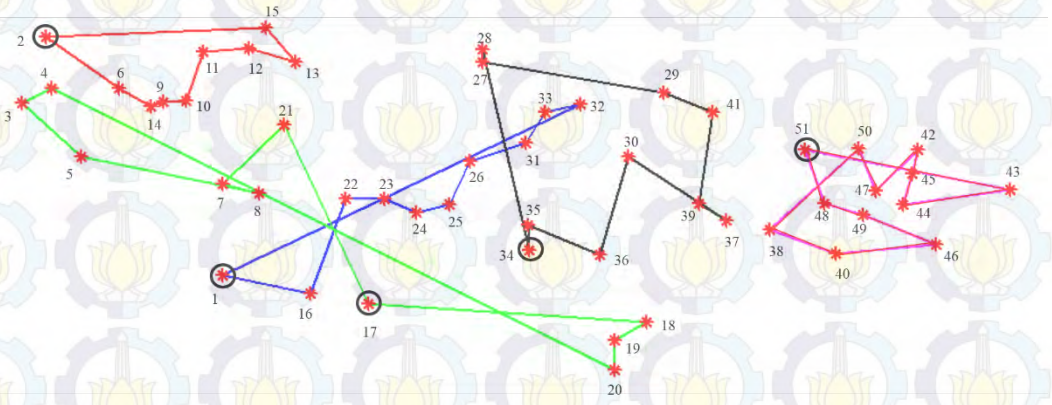


Gambar 4.18 Perbandingan metode *hybrid* FA-ACO dengan metode ACO

Hasil terbaik yang diperoleh metode *hybrid* FA-ACO adalah 13828 dengan waktu komputasi 33,28346 detik. Namun, metode *hybrid* FA-ACO telah konvergen pada 16,86102 detik. Sedangkan metode ACO menghasilkan solusi terbaik yaitu

13882 dengan waktu komputasi 60,2617 detik dan telah konvergen pada 25,07218 detik. Hal ini menunjukkan bahwa metode *hybrid* FA-ACO dapat menemukan solusi yang lebih baik dari metode ACO dan lebih cepat konvergen 32,75% dibandingkan metode ACO.

Graf optimal hasil solusi metode *hybrid* FA-ACO ditunjukkan oleh Gambar 4.19 berikut.



Gambar 4.19 Graf optimal hasil metode *hybrid* FA-ACO

Pada Gambar 4.19, kapal depot 1 (Tanjung Priok) mengunjungi pelabuhan-pelabuhan di Kalimantan. Sedangkan kapal depot 17 (Tanjung Perak) mengunjungi beberapa pelabuhan di Sumatra dan Nusa Tenggara. Hal ini disebabkan kapal depot 1 dan depot 2 (Belawan) telah mendapatkan rute terpendek yang memenuhi batas atas dan batas bawah. Sehingga, pelabuhan 3, 4, dan 5 dikunjungi oleh kapal depot 17 yang merupakan kapal terdekat dari ketiga depot yang tersisa. Hasil rute tersebut menjadi tidak sesuai dengan realitas yang ada.

Untuk mendapatkan rute studi kasus yang sesuai dengan kenyataan di lapangan, dilakukan beberapa penyesuaian pada banyak pelabuhan yang dapat dikunjungi oleh setiap depot berdasarkan keadaan geografis di sekitar depot. Sebagai contoh, pelabuhan 3 (Gunung Sitoli), pelabuhan 4 (Sibolga), dan pelabuhan 5 (Teluk Bayur) hanya dapat dikunjungi oleh kapal depot Belawan karena letak ketiga pelabuhan tersebut jauh dari depot lainnya. Jika penyesuaian ini tidak dilakukan, maka kapal depot Belawan akan mengunjungi pelabuhan lainnya yang lebih dekat untuk memenuhi batas atas dan menghindari ketiga pelabuhan tersebut. Akibatnya, ketiga pelabuhan tersebut dikunjungi oleh kapal depot lainnya yang

jaraknya lebih jauh. Hal ini menyebabkan rute yang dihasilkan kurang efektif jika diterapkan pada realitas yang ada.

Dengan menggunakan metode *hybrid* FA-ACO yang telah disesuaikan, diperoleh graf studi kasus seperti pada Gambar 4.20, dengan rute tiap depot sebagai berikut.

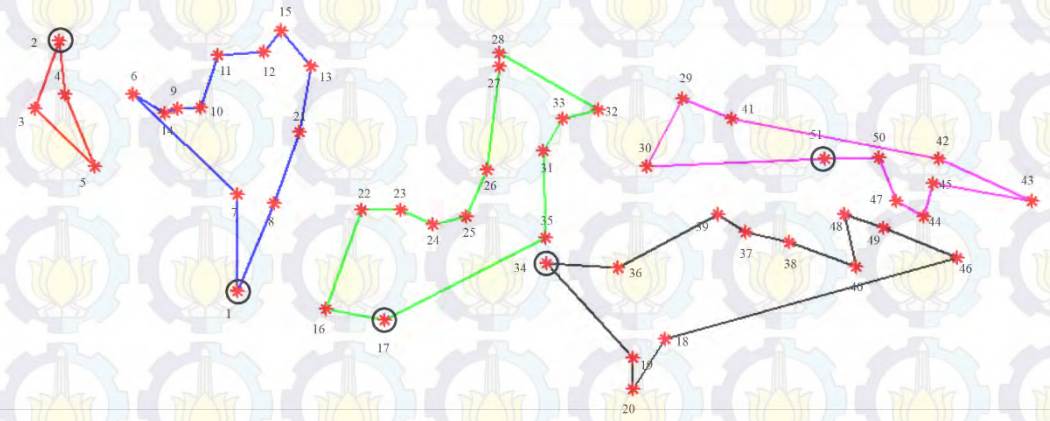
Depot 1 memiliki rute 1-7-6-14-9-10-11-12-15-13-21-8-1

Depot 2 memiliki rute 2-4-5-3-2

Depot 17 memiliki rute 17-16-22-23-24-25-26-27-28-32-33-31-35-17

Depot 34 memiliki rute 34-19-20-18-46-49-48-40-38-37-39-36-34

Depot 51 memiliki rute 51-50-47-44-45-43-42-41-29-30-51.



Gambar 4.20 Graf studi kasus hasil metode *hybrid* FA-ACO

Dari graf tersebut, rute studi kasus diilustrasikan oleh Gambar 4.21 berikut.

Gambar 4.21 Rute studi kasus hasil metode *hybrid* FA-ACO

Dengan demikian, didapatkan rute dari lima puluh satu pelabuhan yang diusulkan metode *hybrid* FA-ACO dengan depot Tanjung Priok, Belawan, Tanjung Perak, Makassar, dan Sorong adalah sebagai berikut.

1. Kapal keberangkatan dari Tanjung Priok memiliki rute Tanjung Priok, Belinyu, Dumai , Tanjung Pandan, Batam, Kijang, Letung, Midai, Natuna, Serasan, Pontianak, Tanjung Balai Karimun, dan Tanjung Priok dengan jarak tempuh 2618 km.
2. Kapal keberangkatan dari Belawan memiliki rute Belawan, Sibolga, Teluk Bayur, Gunung Sitoli, dan Belawan dengan jarak tempuh 1628 km.
3. Kapal keberangkatan dari Tanjung Perak memiliki rute Tanjung Perak, Tanjung Mas, Kumai, Sampit, Banjarmasin, Batu Licin, Balikpapan, Nunukan, Tarakan, Lokodidi, Toli-Toli, Donggala, Parepare, dan Tanjung Perak dengan jarak tempuh 3413 km.
4. Kapal keberangkatan dari Makassar memiliki rute Makassar, Larantuka, Tenau, Kalabahi, Timika, Kaimana, Fakfak, Tual, Banda, Ambon, Bau-Bau, dan Makassar dengan jarak tempuh 3094 km.
5. Kapal keberangkatan dari Sorong memiliki rute Sorong, Manokwari, Wasior, Nabire, Serui, Jayapura, Biak, Ternate, Bitung, Banggai, dan Sorong dengan jarak tempuh 3532 km.

Dari Gambar 4.21, rute penyesuaian lebih dapat diterapkan pada kenyataan di lapangan. Rute penyesuaian tersebut menghasilkan total rute 14285 km. Total rute tersebut lebih besar dibandingkan total rute tanpa penyesuaian.

"*Halaman ini sengaja dikosongkan*."

BAB 5
KESIMPULAN DAN SARAN

Pada bab ini akan diuraikan beberapa kesimpulan dari pembahasan dan analisis hasil yang telah dikerjakan pada Bab 4 yang disertai dengan saran untuk penelitian selanjutnya.

5.1 Kesimpulan

Beberapa kesimpulan yang dapat diambil dari penelitian penyelesaian MmTSP menggunakan *hybrid* FA-ACO pada studi kasus angkutan laut penumpang dari 51 pelabuhan dengan Tanjung Priok, Belawan, Tanjung Perak, Makassar, dan Sorong sebagai depot, antara lain sebagai berikut.

1. Terdapat beberapa faktor yang mempengaruhi *output* metode FA antara lain koefisien penyerapan cahaya γ, banyak kunang-kunang, banyak pergerakan tiap kunang-kunang, dan banyak iterasi. Pada pencarian solusi lokal menggunakan FA dalam menyelesaikan studi kasus didapat bahwa nilai $\gamma = 0{,}01$, kunang-kunang yang diperlukan sebanyak 4 dan 5 pergerakan untuk tiap kunang-kunang serta 100 iterasi FA.
2. Pada metode ACO, faktor-faktor yang berpengaruh antara lain koefisien α dan β, nilai ρ, konstanta Q, banyak semut, dan banyak iterasi. Untuk menyelesaikan studi kasus, beberapa parameter yang dapat menghasilkan solusi optimum pada pencarian solusi global ACO antara lain $\alpha = 1, \beta = 5, \rho = 0{,}5, Q = 100$, 20 semut, dan 229 iterasi.
3. Hasil terbaik yang didapat pada metode *hybrid* FA-ACO dalam menyelesaikan studi kasus dengan matriks jarak yang ditunjukkan pada Lampiran 1, untuk 50 kali percobaan adalah 13828 km. Rata-rata solusi dari 50 percobaan untuk metode *hybrid* FA-ACO adalah 14400,28 km dengan rata-rata waktu komputasi 36,01579 detik. Hasil terbaik yang didapat pada metode ACO adalah 13882 km. Rata-rata solusi metode ACO adalah 14463,24 km dengan rata-rata waktu komputasi 60,77814 detik.

4. Waktu yang diperlukan metode *hybrid* FA-ACO untuk mencapai hasil terbaik 13828 km adalah 33,28346 detik dan telah konvergen pada 16,86102 detik. Sedangkan metode ACO mencapai solusi terbaik 13882 km dalam waktu 60,2617 detik dan telah konvergen pada 25,07218 detik.
5. Metode *hybrid* FA-ACO lebih baik dalam mencapai solusi optimum dengan rata-rata waktu komputasi 40.74% dan waktu konvergensi 32,75% lebih cepat dibandingkan metode ACO.

5.2 Saran

Studi kasus yang digunakan pada penelitian ini adalah studi kasus angkutan laut yang dibatasi untuk permasalahan MmTSP. Sehingga, hasil penelitian hanya memperhatikan faktor jarak saja tanpa memperhatikan realitas yang ada. Untuk memperoleh hasil yang sesuai dengan realitas, pada penelitian ini dilakukan beberapa penyesuaian. Oleh karena itu, diperlukan penelitian lebih lanjut mengenai suatu metode yang dapat menyelesikan studi kasus dengan melakukan *cluster* terlebih dahulu agar sesuai dengan kendala-kendala yang ada dan dapat diterapkan di lapangan.

Hasil penelitian menunjukkan bahwa metode *hybrid* FA-ACO menghasilkan solusi yang lebih baik dengan waktu komputasi lebih cepat dibandingkan metode ACO. Kesimpulan ini didukung dengan penelitian sebelumnya yang menyatakan *hybrid* FA-ACO lebih baik dibandingkan dengan ACO. Namun, belum ada bukti analitis yang mendukung penelitian. Sehingga, diharapkan terdapat penelitian lebih lanjut mengenai bukti analitis yang dapat menunjukkan bahwa waktu komputasi *hybrid* FA-ACO lebih baik dari metode ACO untuk semua permasalahan.

DAFTAR PUSTAKA

Bektas, T. (2006), "The Multiple Traveling Salesman Problem: An Overview of Formulations and Solution Procedures", *Omega*, vol. 34, hal. 209-219.

Danandjojo, R.I. (2014), *Pengembangan Varian Model VRP untuk Penentuan Rute Angkutan Laut Penumpang*, Disertasi Jurusan Transportasi, Institut Teknologi Bandung, Bandung.

Dorigo, M., Maniezzo, V, dan Colorni, A. (1996), "The Ant System: Optimization by a Colony of Cooperating Agents", *IEEE Transactions on System, Man, And Cybernetics-Part B: Cybernetics*, Vol. 26, No. 1, hal. 29-41.

Ghafurian, S. dan Javadian, N. (2011), "An Ant Colony Algorithm for Solving Fixed Destination Multi-Depot Multiple Traveling Salesman Problems", *Applied Soft Computing*, vol. 11, hal. 1256-1262.

Gross, J.L. dan Yellen, J, (2006), *Graph Theory and Its Applications*, 2nd edition, Chapman & Hll/CRC, Boca Raton.

Jati, G.K. dan Suyanto (2011), "Evolutionary Discrete Firefly Algorithm for Travelling Salesman Problem", *Proceedings of Second International Conference ICAIS 2011, LNAI 6943*, Eds. Bouchachia, A., University of Klagenfurt, Klagenfurt, hal. 393-403.

Jati, G.K., Manurung, R., dan Suyanto (2013), "Discrete Firefly Algorithm for Traveling Salesman Problem: A New Movement Scheme", *Swarm Intelligence and Bio-Inspired Computation*, hal. 295-312.

Kara, I., dan Bektas, T. (2006), "Integer Linear Programming Formulations of Multiple Salesman Problems and Its Variations", *Europe Journal of Operation Research*, vol. 174, hal. 1449-1458.

Lenstra, J.K. dan Rinnooy Kan, A.H.G. (1975), "Some Simple Applications of Travelling Salesman Problem", *Opl Res. Q., Pergamon Press*, Vol. 26, 4, i, hal. 717-733.

Liu, W., Li, S., dan Zhao, F. (2009), "An Ant Colony Optimization Algorithm for The Multiple Traveling Salesman Problem", *IEEE ICIEA 2009*, hal. 1533-1537.

Jati, G.K., Manurung, R., and Suyanto. 2013. Discrete Firefly Algorithm for Traveling Salesman Problem: A New Movement Scheme. *Swarm Intelligence and Bio-Inspired Computation*, 295-312.

Jellouli, O., 2001. Intelligent dynamic programming for the generalized traveling salesman problem. In: *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, vol. 4. IEEE Computer Society, Washington, DC, 2765-2768.

Jiang, H., Zhou, Z., Zhou, P., Chen, G.L. 2005. Union Search: A New Metaheuristic Algorithm to the Traveling Salesman Problem. J. *Univ. Sci. Technol. China*, 35: 367-375.

Land, A., Doig, A.1960. An Automatic Method of Solving Discrete Programming Problems. *Econometrica*. 28 (3), 497-520.

Lenstra, J.K. and Rinnooy Kan, A.H.G. 1975. Some Simple Applications of Travelling Salesman Problem. *Opl Res. Q., Pergamon Press*, 26(4): 717-733.

Lin, S., Kernighan, B. 1973. "An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Oper. Res*, 21(2): 498-516.

Yang, X.S. 2010. *Nature-Inspired Metaheuristic Algorithm*, 2nd edition. United Kingdom: Luniver Press.

TSPLIB95: Ruprecht—Karls—Universitat Heildelberg, 2011. ,http://www.iwr.uini-heidelberg.de/groups/comopt/software/TSPLIB95/ (accessed 10.05.15).

Wilson, R.J. dan Watkins, J.J, (1990), *Graphs: An Introductory Approuch: A first Course in Discrete Mathematics*, John Wiley & Sons, Inc, Kanada.

Yang, X.S., (2010), *Nature-Inspired Metaheuristic Algorithm*, 2nd edition, Luniver Press, United Kingdom.

Younes, M. (2013), "A novel hybrid FFA-ACO Algorithm for Economic Power Dispatch", *CEAI*, Vol.15, hal. 67-77.

Lampiran 1

Kode	Pelabuhan	Tanjung Priok	Belawan	Gunung Sitoli	Sibolga	Teluk Bayur	Dumai	Blinyu	Tanjung Pandan
1	Tanjung Priok		863	772	764	556	689	534	222
2	Belawan	863		584	614	885	291	610	711
3	Gunung Sitoli	772	584		75	219	828	1,167	942
4	Sibolga	764	614	75		211	858	1,197	937
5	Teluk Bayur	556	885	219	211		1,147	1,078	725
6	Dumai	689	291	828	858	1,147		420	533
7	Blinyu	534	610	1,167	1,197	1,078	420		162
8	Tanjung Pandan	222	711	942	937	725	533	162	
9	Batam	672	389	945	976	1,137	199	195	345
10	Kijang	640	423	979	1,009	1,184	232	166	283
11	Letung	1,034	598	1,154	1,184	1,359	407	341	458
12	Midai	879	753	1,309	1,339	1,514	562	496	613
13	Serasan	800	832	1,388	1,418	1,593	641	575	692
14	Tanjung Balai Karimun	728	383	939	970	1,151	193	276	381
15	Natuna	746	673	1,229	1,259	1,290	482	360	374
16	Tanjung Emas	235	1,038	988	980	788	848	511	335
17	Tanjung Perak	384	1,130	1,150	1,142	932	951	598	486
18	Kalabahi	1,093	1,852	1,848	1,840	1,648	1,661	1,219	1,076
19	Larantuka	1,003	1,762	1,758	1,750	1,558	1,571	1,129	986
20	Tenau / Kupang	1,108	1,806	1,797	101	1,597	1,619	1,282	1,163
21	Pontianak	420	1,292	1,231	1,223	1,031	512	256	215
22	Kumai	363	951	1,110	1,102	910	761	423	352
23	Sampit	466	1,094	1,217	1,209	1,017	904	567	470
24	Banjarmasin	505	1,137	1,257	1,249	1,057	947	609	513
25	Batu Licin	648	1,287	1,400	1,392	1,200	1,096	759	638
26	Balikpapan	765	1,680	1,530	1,522	1,309	1,252	897	772
27	Nunukan	1,252	1,655	2,005	1,997	1,805	1,465	1,317	1,266
28	Tarakan	1,109	2,025	1,968	1,960	1,654	1,483	1,337	1,117
29	Bitung	1,427	1,934	2,200	2,192	2,000	1,744	1,596	1,472
30	Banggai	1,150	2,067	1,921	1,913	1,696	1,666	1,297	1,172
31	Donggala / Pantoloan	970	1,633	1,723	1,715	1,523	1,443	1,105	1,026
32	Loko Didi	1,229	1,892	1,982	1,974	1,782	1,702	1,364	1,285
33	Toli-toli	1,142	1,805	1,895	1,887	1,695	1,615	1,277	1,198
34	Makassar	762	1,679	1,533	1,525	1,308	1,278	909	784
35	Pare Pare	783	1,682	1,568	1,560	1,311	1,307	934	790
36	Bau-bau	956	1,873	1,727	1,719	1,502	1,472	1,103	978
37	Ambon	1,331	2,002	2,081	2,073	1,881	1,812	1,475	1,334
38	Banda	1,463	2,134	2,213	2,205	2,013	1,944	1,607	1,466
39	Namlea	1,245	1,916	1,995	1,987	1,795	1,726	1,389	1,248
40	Tual	1,656	2,327	2,406	2,398	2,206	2,137	1,800	1,659
41	Ternate	1,478	2,036	2,228	2,220	2,029	1,845	1,622	1,501
42	Biak	1,866	2,782	2,635	2,623	2,411	2,378	2,011	1,869
43	Jayapura	2,189	2,825	2,939	2,931	2,739	2,635	2,315	2,173
44	Nabire	1,738	2,374	2,488	2,480	2,288	2,184	1,864	1,722
45	Serui	1,870	2,506	2,620	2,612	2,420	2,316	1,996	1,854
46	Timika	1,942	2,613	2,692	2,684	2,492	2,423	2,086	1,964
47	Wasior	1,866	2,326	2,616	2,608	2,416	2,335	1,993	1,851
48	Fak-Fak	1,583	2,252	2,333	2,325	2,133	2,061	1,724	1,603
49	Kaimana	1,793	2,462	2,543	2,535	2,343	2,271	1,934	1,813
50	Manokwari	1,780	2,240	2,530	2,522	2,330	2,249	1,907	1,765
51	Sorong	1,577	2,248	2,327	2,319	2,127	2,058	1,720	1,579

Kode	Pelabuhan	Batam	Kijang	Letung	Midai	Serasan	Tanjung Balai Karimun	Natuna	Tanjung Emas
1	Tanjung Priok	672	640	1,034	879	800	728	746	235
2	Belawan	389	423	598	753	832	383	673	1,038
3	Gunung Sitoli	945	979	1,154	1,309	1,388	939	1,229	988
4	Sibolga	976	1,009	1,184	1,339	1,418	970	1,259	980
5	Teluk Bayur	1,137	1,184	1,359	1,514	1,593	1,151	1,290	788
6	Dumai	199	232	407	562	641	193	482	848
7	Blinyu	195	166	341	496	575	276	360	511
8	Tanjung Pandan	345	283	458	613	692	381	374	335
9	Batam		42	217	372	451	55	302	674
10	Kijang	42		155	330	409	88	290	617
11	Letung	217	155		155	234	243	187	1,078
12	Midai	372	330	155		79	418	132	923
13	Serasan	451	409	234	79		497	115	844
14	Tanjung Balai Karimun	55	88	243	418	497		339	704
15	Natuna	302	290	187	132	115	339		723
16	Tanjung Emas	674	617	1,078	923	844	704	723	
17	Tanjung Perak	736	704	1,076	921	842	791	810	205
18	Kalabahi	1,352	1,320	1,904	1,749	1,670	1,386	1,536	897
19	Larantuka	1,262	1,230	1,814	1,659	1,580	1,296	1,446	807
20	Tenau / Kupang	1,420	1,388	1,840	1,685	1,606	1,475	1,494	920
21	Pontianak	329	290	614	459	380	369	300	464
22	Kumai	562	530	990	835	756	617	636	266
23	Sampit	705	673	1,134	979	900	760	779	325
24	Banjarmasin	747	716	1,176	1,021	942	803	822	343
25	Batu Licin	897	865	1,326	1,171	1,092	953	971	466
26	Balikpapan	1,035	1,003	1,450	1,295	1,216	1,091	1,110	596
27	Nunukan	1,285	1,493	1,789	1,634	1,555	1,321	573	1,058
28	Tarakan	1,305	1,456	1,761	1,606	1,527	1,341	558	1,021
29	Bitung	1,564	1,700	2,068	1,913	1,834	1,600	1,261	1,234
30	Banggai	1,472	1,404	1,849	1,694	1,615	1,491	1,510	972
31	Donggala / Pantoloan	1,244	1,212	1,672	1,517	1,438	1,299	1,318	777
32	Loko Didi	1,503	1,471	1,931	1,776	1,697	1,558	1,577	1,036
33	Toli-toli	1,416	1,384	1,844	1,689	1,610	1,471	1,490	949
34	Makassar	1,084	1,016	1,461	1,306	1,227	1,103	1,122	584
35	Pare Pare	1,072	1,040	1,465	1,310	1,231	1,128	1,146	622
36	Bau-bau	1,278	1,210	1,655	1,500	1,421	1,297	1,316	778
37	Ambon	1,613	1,581	2,042	1,887	1,808	1,668	1,616	1,136
38	Banda	1,745	1,713	2,174	2,019	1,940	1,800	1,748	1,268
39	Namlea	1,527	1,495	1,956	1,801	1,722	1,582	1,530	1,050
40	Tual	1,938	1,906	2,367	2,212	2,133	1,993	1,941	1,461
41	Ternate	1,665	1,728	2,169	2,014	1,935	1,702	1,363	1,283
42	Biak	2,169	2,112	2,579	2,424	2,345	2,205	1,672	1,690
43	Jayapura	2,455	2,416	2,789	2,634	2,555	2,491	1,966	1,994
44	Nabire	2,004	1,965	2,338	2,183	2,104	2,040	1,515	1,543
45	Serui	2,136	2,097	2,470	2,315	2,236	2,172	1,647	1,675
46	Timika	2,224	2,192	2,653	2,498	2,419	2,279	2,298	1,747
47	Wasior	2,155	2,093	2,466	2,311	2,232	2,192	1,643	1,671
48	Fak-Fak	1,862	1,830	2,291	2,136	2,057	1,918	1,756	1,386
49	Kaimana	2,072	2,040	2,501	2,346	2,267	2,128	1,966	1,596
50	Manokwari	2,069	2,007	2,380	2,225	2,146	2,106	1,557	1,585
51	Sorong	1,859	1,827	2,287	2,132	2,053	1,914	1,669	1,382

Kode	Pelabuhan	Tanjung Perak	Kalabahi	Larantuka	Tenau / Kupang	Pontianak	Kumai	Sampit	Banjarmasin
1	Tanjung Priok	384	1,093	1,003	1,108	420	363	466	505
2	Belawan	1,130	1,852	1,762	1,806	1,292	951	1,094	1,137
3	Gunung Sitoli	1,150	1,848	1,758	1,797	1,231	1,110	1,217	1,257
4	Sibolga	1,142	1,840	1,750	101	1,223	1,102	1,209	1,249
5	Teluk Bayur	932	1,648	1,558	1,597	1,031	910	1,017	1,057
6	Dumai	951	1,661	1,571	1,619	512	761	904	947
7	Blinyu	598	1,219	1,129	1,282	256	423	567	609
8	Tanjung Pandan	486	1,076	986	1,163	215	352	470	513
9	Batam	736	1,352	1,262	1,420	329	562	705	747
10	Kijang	704	1,320	1,230	1,388	290	530	673	716
11	Letung	1,076	1,904	1,814	1,840	614	990	1,134	1,176
12	Midai	921	1,749	1,659	1,685	459	835	979	1,021
13	Serasan	842	1,670	1,580	1,606	380	756	900	942
14	Tanjung Balai Karimun	791	1,386	1,296	1,475	369	617	760	803
15	Natuna	810	1,536	1,446	1,494	300	636	779	822
16	Tanjung Emas	205	897	807	920	464	266	325	343
17	Tanjung Perak		749	659	752	462	279	289	263
18	Kalabahi	749		90	142	1,290	970	816	725
19	Larantuka	659	90		118	1,200	880	726	635
20	Tenau / Kupang	752	142	118		1,226	945	906	840
21	Pontianak	462	1,290	1,200	1,226		376	520	562
22	Kumai	279	970	880	945	376		200	245
23	Sampit	289	816	726	906	520	200		155
24	Banjarmasin	263	725	635	840	562	245	155	
25	Batu Licin	347	630	540	812	712	402	316	225
26	Balikpapan	583	696	606	889	836	533	448	357
27	Nunukan	924	1,030	940	1,214	1,175	975	885	793
28	Tarakan	932	1,096	1,006	1,268	1,147	938	847	756
29	Bitung	1,100	596	506	792	1,454	1,151	1,060	969
30	Banggai	981	808	718	635	1,235	880	887	796
31	Donggala / Pantoloan	642	728	638	1,047	1,058	693	603	512
32	Loko Didi	901	987	897	1,306	1,317	952	862	771
33	Toli-toli	814	900	810	1,219	1,230	865	775	684
34	Makassar	593	420	330	635	847	492	499	408
35	Pare Pare	603	541	451	719	851	495	534	443
36	Bau-bau	787	614	524	441	1,041	686	693	602
37	Ambon	994	366	456	500	1,428	1,110	1,046	955
38	Banda	1,126	498	588	632	1,560	1,242	1,178	1,087
39	Namlea	908	280	370	414	1,342	1,024	960	869
40	Tual	1,319	691	781	825	1,753	1,435	1,371	1,280
41	Ternate	1,141	582	672	741	1,555	1,252	1,162	1,071
42	Biak	1,534	828	918	1,091	1,965	1,729	1,600	1,509
43	Jayapura	1,852	1,240	1,330	1,448	2,175	2,039	1,904	1,813
44	Nabire	1,401	789	879	997	1,724	1,588	1,453	1,362
45	Serui	1,533	921	1,011	1,129	1,856	1,720	1,585	1,494
46	Timika	1,605	902	992	1,017	2,039	1,729	1,657	1,566
47	Wasior	1,529	831	921	1,126	1,852	1,710	1,581	1,490
48	Fak-Fak	1,244	582	672	715	1,677	1,371	1,298	1,212
49	Kaimana	1,454	792	882	925	1,887	1,581	1,508	1,422
50	Manokwari	1,443	745	835	1,040	1,766	1,624	1,495	1,404
51	Sorong	1,239	616	706	782	1,673	1,356	1,292	1,201

Kode	Pelabuhan	Batu Licin	Balikpapan	Nunukan	Tarakan	Bitung	Banggai	Donggala / Pantoloan	Loko Didi
1	Tanjung Priok	648	765	1,252	1,109	1,427	1,150	970	1,229
2	Belawan	1,287	1,680	1,655	2,025	1,934	2,067	1,633	1,892
3	Gunung Sitoli	1,400	1,530	2,005	1,968	2,200	1,921	1,723	1,982
4	Sibolga	1,392	1,522	1,997	1,960	2,192	1,913	1,715	1,974
5	Teluk Bayur	1,200	1,309	1,805	1,654	2,000	1,696	1,523	1,782
6	Dumai	1,096	1,252	1,465	1,483	1,744	1,666	1,443	1,702
7	Blinyu	759	897	1,317	1,337	1,596	1,297	1,105	1,364
8	Tanjung Pandan	638	772	1,266	1,117	1,472	1,172	1,026	1,285
9	Batam	897	1,035	1,285	1,305	1,564	1,472	1,244	1,503
10	Kijang	865	1,003	1,493	1,456	1,700	1,404	1,212	1,471
11	Letung	1,326	1,450	1,789	1,761	2,068	1,849	1,672	1,931
12	Midai	1,171	1,295	1,634	1,606	1,913	1,694	1,517	1,776
13	Serasan	1,092	1,216	1,555	1,527	1,834	1,615	1,438	1,697
14	Tanjung Balai Karimun	953	1,091	1,321	1,341	1,600	1,491	1,299	1,558
15	Natuna	971	1,110	573	558	1,261	1,510	1,318	1,577
16	Tanjung Emas	466	596	1,058	1,021	1,234	972	777	1,036
17	Tanjung Perak	347	583	924	932	1,100	981	642	901
18	Kalabahi	630	696	1,030	1,096	596	808	728	987
19	Larantuka	540	606	940	1,006	506	718	638	897
20	Tenau / Kupang	812	889	1,214	1,268	792	635	1,047	1,306
21	Pontianak	712	836	1,175	1,147	1,454	1,235	1,058	1,317
22	Kumai	402	533	975	938	1,151	880	693	952
23	Sampit	316	448	885	847	1,060	887	603	862
24	Banjarmasin	225	357	793	756	969	796	512	771
25	Batu Licin		150	554	516	699	598	296	555
26	Balikpapan	150		437	378	628	680	196	455
27	Nunukan	554	437		83	528	998	328	587
28	Tarakan	516	378	83		521	944	291	550
29	Bitung	699	628	528	521		727	496	485
30	Banggai	598	680	998	944	727		690	975
31	Donggala / Pantoloan	296	196	328	291	496	690		259
32	Loko Didi	555	455	587	550	485	975	259	
33	Toli-toli	468	368	500	463	398	888	172	87
34	Makassar	210	292	610	556	727	388	328	587
35	Pare Pare	204	247	562	510	738	509	280	539
36	Bau-bau	404	486	804	750	533	194	522	781
37	Ambon	782	888	883	875	380	414	851	691
38	Banda	914	1,020	1,015	1,007	512	546	983	823
39	Namlea	696	802	797	789	294	328	765	605
40	Tual	1,107	1,213	1,208	1,200	705	739	1,176	1,016
41	Ternate	800	729	630	622	147	561	597	458
42	Biak	1,313	1,334	1,180	1,150	746	947	1,167	1,057
43	Jayapura	1,705	1,590	1,471	1,474	1,050	1,311	1,458	1,361
44	Nabire	1,210	1,139	1,020	1,023	599	860	1,007	910
45	Serui	1,342	1,271	1,152	1,155	731	992	1,139	1,042
46	Timika	1,401	1,507	1,446	1,439	950	1,033	1,414	1,261
47	Wasior	1,354	1,283	1,164	1,167	727	988	1,152	1,038
48	Fak-Fak	1,043	1,122	1,023	1,015	528	675	991	839
49	Kaimana	1,253	1,332	1,233	1,225	738	885	1,201	1,049
50	Manokwari	1,268	1,197	1,078	1,081	641	902	1,066	952
51	Sorong	1,028	1,036	936	929	440	660	904	751

Kode	Pelabuhan	Toli-toli	Makassar	Pare Pare	Bau-bau	Ambon	Banda	Namlea	Tual
1	Tanjung Priok	1,142	762	783	956	1,331	1,463	1,245	1,656
2	Belawan	1,805	1,679	1,682	1,873	2,002	2,134	1,916	2,327
3	Gunung Sitoli	1,895	1,533	1,568	1,727	2,081	2,213	1,995	2,406
4	Sibolga	1,887	1,525	1,560	1,719	2,073	2,205	1,987	2,398
5	Teluk Bayur	1,695	1,308	1,311	1,502	1,881	2,013	1,795	2,206
6	Dumai	1,615	1,278	1,307	1,472	1,812	1,944	1,726	2,137
7	Blinyu	1,277	909	934	1,103	1,475	1,607	1,389	1,800
8	Tanjung Pandan	1,198	784	790	978	1,334	1,466	1,248	1,659
9	Batam	1,416	1,084	1,072	1,278	1,613	1,745	1,527	1,938
10	Kijang	1,384	1,016	1,040	1,210	1,581	1,713	1,495	1,906
11	Letung	1,844	1,461	1,465	1,655	2,042	2,174	1,956	2,367
12	Midai	1,689	1,306	1,310	1,500	1,887	2,019	1,801	2,212
13	Serasan	1,610	1,227	1,231	1,421	1,808	1,940	1,722	2,133
14	Tanjung Balai Karimun	1,471	1,103	1,128	1,297	1,668	1,800	1,582	1,993
15	Natuna	1,490	1,122	1,146	1,316	1,616	1,748	1,530	1,941
16	Tanjung Emas	949	584	622	778	1,136	1,268	1,050	1,461
17	Tanjung Perak	814	593	603	787	994	1,126	908	1,319
18	Kalabahi	900	420	541	614	366	498	280	691
19	Larantuka	810	330	451	524	456	588	370	781
20	Tenau / Kupang	1,219	635	719	441	500	632	414	825
21	Pontianak	1,230	847	851	1,041	1,428	1,560	1,342	1,753
22	Kumai	865	492	495	686	1,110	1,242	1,024	1,435
23	Sampit	775	499	534	693	1,046	1,178	960	1,371
24	Banjarmasin	684	408	443	602	955	1,087	869	1,280
25	Batu Licin	468	210	204	404	782	914	696	1,107
26	Balikpapan	368	292	247	486	888	1,020	802	1,213
27	Nunukan	500	610	562	804	883	1,015	797	1,208
28	Tarakan	463	556	510	750	875	1,007	789	1,200
29	Bitung	398	727	738	533	380	512	294	705
30	Banggai	888	388	509	194	414	546	328	739
31	Donggala / Pantoloan	172	328	280	522	851	983	765	1,176
32	Loko Didi	87	587	539	781	691	823	605	1,016
33	Toli-toli		500	452	694	778	910	692	1,103
34	Makassar	500		121	194	608	740	522	933
35	Pare Pare	452	121		315	698	830	612	1,023
36	Bau-bau	694	194	315		414	546	328	739
37	Ambon	778	608	698	414		132	86	325
38	Banda	910	740	830	546	132		218	193
39	Namlea	692	522	612	328	86	218		411
40	Tual	1,103	933	1,023	739	325	193	411	
41	Ternate	545	755	839	561	321	453	235	633
42	Biak	1,144	1,141	1,235	947	655	523	741	719
43	Jayapura	1,448	1,505	1,595	1,311	939	807	1,025	1,024
44	Nabire	997	1,054	1,144	860	488	356	574	573
45	Serui	1,129	1,186	1,276	992	620	488	706	705
46	Timika	1,348	1,227	1,317	1,033	663	531	749	669
47	Wasior	1,125	1,182	1,272	988	616	484	702	701
48	Fak-Fak	926	869	959	675	293	161	379	218
49	Kaimana	1,136	1,079	1,169	885	503	371	589	428
50	Manokwari	1,039	1,096	1,186	902	530	398	616	615
51	Sorong	838	854	944	660	327	195	413	407

Kode	Pelabuhan	Ternate	Biak	Jayapura	Nabire	Serui	Timika	Wasior	Fak-Fak
1	Tanjung Priok	1,478	1,866	2,189	1,738	1,870	1,942	1,866	1,583
2	Belawan	2,036	2,782	2,825	2,374	2,506	2,613	2,326	2,252
3	Gunung Sitoli	2,228	2,635	2,939	2,488	2,620	2,692	2,616	2,333
4	Sibolga	2,220	2,623	2,931	2,480	2,612	2,684	2,608	2,325
5	Teluk Bayur	2,029	2,411	2,739	2,288	2,420	2,492	2,416	2,133
6	Dumai	1,845	2,378	2,635	2,184	2,316	2,423	2,335	2,061
7	Blinyu	1,622	2,011	2,315	1,864	1,996	2,086	1,993	1,724
8	Tanjung Pandan	1,501	1,869	2,173	1,722	1,854	1,964	1,851	1,603
9	Batam	1,665	2,169	2,455	2,004	2,136	2,224	2,155	1,862
10	Kijang	1,728	2,112	2,416	1,965	2,097	2,192	2,093	1,830
11	Letung	2,169	2,579	2,789	2,338	2,470	2,653	2,466	2,291
12	Midai	2,014	2,424	2,634	2,183	2,315	2,498	2,311	2,136
13	Serasan	1,935	2,345	2,555	2,104	2,236	2,419	2,232	2,057
14	Tanjung Balai Karimun	1,702	2,205	2,491	2,040	2,172	2,279	2,192	1,918
15	Natuna	1,363	1,672	1,966	1,515	1,647	2,298	1,643	1,756
16	Tanjung Emas	1,283	1,690	1,994	1,543	1,675	1,747	1,671	1,386
17	Tanjung Perak	1,141	1,534	1,852	1,401	1,533	1,605	1,529	1,244
18	Kalabahi	582	828	1,240	789	921	902	831	582
19	Larantuka	672	918	1,330	879	1,011	992	921	672
20	Tenau / Kupang	741	1,091	1,448	997	1,129	1,017	1,126	715
21	Pontianak	1,555	1,965	2,175	1,724	1,856	2,039	1,852	1,677
22	Kumai	1,252	1,729	2,039	1,588	1,720	1,729	1,710	1,371
23	Sampit	1,162	1,600	1,904	1,453	1,585	1,657	1,581	1,298
24	Banjarmasin	1,071	1,509	1,813	1,362	1,494	1,566	1,490	1,212
25	Batu Licin	800	1,313	1,705	1,210	1,342	1,401	1,354	1,043
26	Balikpapan	729	1,334	1,590	1,139	1,271	1,507	1,283	1,122
27	Nunukan	630	1,180	1,471	1,020	1,152	1,446	1,164	1,023
28	Tarakan	622	1,150	1,474	1,023	1,155	1,439	1,167	1,015
29	Bitung	147	746	1,050	599	731	950	727	528
30	Banggai	561	947	1,311	860	992	1,033	988	675
31	Donggala / Pantoloan	597	1,167	1,458	1,007	1,139	1,414	1,152	991
32	Loko Didi	458	1,057	1,361	910	1,042	1,261	1,038	839
33	Toli-toli	545	1,144	1,448	997	1,129	1,348	1,125	926
34	Makassar	755	1,141	1,505	1,054	1,186	1,227	1,182	869
35	Pare Pare	839	1,235	1,595	1,144	1,276	1,317	1,272	959
36	Bau-bau	561	947	1,311	860	992	1,033	988	675
37	Ambon	321	655	939	488	620	663	616	293
38	Banda	453	523	807	356	488	531	484	161
39	Namlea	235	741	1,025	574	706	749	702	379
40	Tual	633	719	1,024	573	705	669	701	218
41	Ternate		614	918	467	599	837	595	415
42	Biak	614		315	140	115	1,049	211	501
43	Jayapura	918	315		451	319	1,353	516	806
44	Nabire	467	140	451		132	902	316	355
45	Serui	599	115	319	132		1,034	197	487
46	Timika	837	1,049	1,353	902	1,034		1,030	451
47	Wasior	595	211	516	316	197	1,030		483
48	Fak-Fak	415	501	806	355	487	451	483	
49	Kaimana	625	711	1,016	565	697	241	693	210
50	Manokwari	509	125	430	230	111	944	86	397
51	Sorong	366	321	624	173	305	741	302	189

Kode	Pelabuhan	Kaimana	Manokwari	Sorong
1	Tanjung Priok	1,793	1,780	1,577
2	Belawan	2,462	2,240	2,248
3	Gunung Sitoli	2,543	2,530	2,327
4	Sibolga	2,535	2,522	2,319
5	Teluk Bayur	2,343	2,330	2,127
6	Dumai	2,271	2,249	2,058
7	Blinyu	1,934	1,907	1,720
8	Tanjung Pandan	1,813	1,765	1,579
9	Batam	2,072	2,069	1,859
10	Kijang	2,040	2,007	1,827
11	Letung	2,501	2,380	2,287
12	Midai	2,346	2,225	2,132
13	Serasan	2,267	2,146	2,053
14	Tanjung Balai Karimun	2,128	2,106	1,914
15	Natuna	1,966	1,557	1,669
16	Tanjung Emas	1,596	1,585	1,382
17	Tanjung Perak	1,454	1,443	1,239
18	Kalabahi	792	745	616
19	Larantuka	882	835	706
20	Tenau / Kupang	925	1,040	782
21	Pontianak	1,887	1,766	1,673
22	Kumai	1,581	1,624	1,356
23	Sampit	1,508	1,495	1,292
24	Banjarmasin	1,422	1,404	1,201
25	Batu Licin	1,253	1,268	1,028
26	Balikpapan	1,332	1,197	1,036
27	Nunukan	1,233	1,078	936
28	Tarakan	1,225	1,081	929
29	Bitung	738	641	440
30	Banggai	885	902	660
31	Donggala / Pantoloan	1,201	1,066	904
32	Loko Didi	1,049	952	751
33	Toli-toli	1,136	1,039	838
34	Makassar	1,079	1,096	854
35	Pare Pare	1,169	1,186	944
36	Bau-bau	885	902	660
37	Ambon	503	530	327
38	Banda	371	398	195
39	Namlea	589	616	413
40	Tual	428	615	407
41	Ternate	625	509	366
42	Biak	711	125	321
43	Jayapura	1,016	430	624
44	Nabire	565	230	173
45	Serui	697	111	305
46	Timika	241	944	741
47	Wasior	693	86	302
48	Fak-Fak	210	397	189
49	Kaimana		607	399
50	Manokwari	607		216
51	Sorong	399	216	

BIOGRAFI PENULIS

Olief Ilmandira Ratu Farisi. Itulah nama yang diberikan oleh orang tua penulis, Luluk Ilmiati dan Salman Farisi, ketika manusia baru terlahir di Pasuruan pada 25 Oktober 1989. Dibesarkan sebagai anak tunggal, tidak menjadikannya anak manja dan bergantung kepada orang tua.

Keinginannya untuk bersekolah membuat orang tuanya mendaftarkannya sebagai siswa di Taman Kanak-kanak Shandy Putra Pasuruan. Dua tahun dijalani, dan akhirnya orang tuanya kembali direpotkan untuk mencari sekolah yang mau menerima anak di bawah umur. SDN Kebonsari I Pasuruan-lah yang menjadi tempat belajarnya selama enam tahun. Tahun 2001, ia lulus dan menempuh pendidikan selanjutnya di SMPN 2 Pasuruan. Tamat SMP pada tahun 2004, ia melanjutkan pendidikannya di SMAN 1 Pasuruan dan lulus pada tahun 2007. Setelah satu tahun mengumpulkan biaya kuliah, pada tahun 2008, ia diterima di S1 Pendidikan Matematika Universitas Negeri Malang. Penulis menyelesaikan studi S1-nya pada Maret 2012.

Penulis mengajar di SMK Nasional Malang pada Tahun Ajaran 2012/2013 selama satu tahun sebelum akhirnya melanjutkan studi magister di Jurusan Matematika Institut Teknologi Sepuluh Nopember pada 2013 melalui Beasiswa Pendidikan Pascasarjana Dalam Neeri (BPPDN) DIKTI untuk calon dosen. Penulis menempuh pendidikan masternya selama dua tahun.

Penulis merupakan anggota dari Keluarga Silat Nasional Indonesia Perisai Diri. Dia bergabung sejak duduk di bangku SMP hingga saat ini. Beberapa kejuaraan pencak silat tingkat Jawa Timur pernah ia raih. Selama di perguruan tinggi, ia pernah menjadi asisten dosen untuk matakuliah Dasar Pemrograman Komputer dan Discrete Method. Selain itu, ia merupakan tentor SMP dan SMA untuk mata pelajaran Matematika di suatu lembaga bimbingan belajar. Penulis dapat dihubungi melalui email olief.ilmandira@gmail.com.

A Hybrid Firefly Algorithm – Ant Colony Optimization for Traveling Salesman Problem

Olief Ilmandira Ratu Farisi[1], Budi Setiyono[2], R. Imbang Danandjojo[3]
[1,2]Study Program Mathematics, Faculty of Mathematics and Science, Institut Teknologi Sepuluh Nopember
Kampus ITS Keputih, Sukolilo, Surabaya, 60111, Jawa Timur, Indonesia
[3]Badan Penelitian dan Pengembangan Perhubungan, Kementerian Perhubungan Republik Indonesia
Kementerian Perhubungan Republik Indonesia, Jakarta, Indonesia
Email: [1]olief.ilmandira@gmail.com, [2]budi@matematika.its.ac.id, [3]dj_imbang@yahoo.co.id

Abstrak. *Pada penelitian ini, dikembangkan suatu metode baru yaitu hybrid firefly algorithm-ant colony optimization (hybrid FA-ACO) untuk menyelesaikan masalah traveling salesman problem (TSP). ACO memiliki komputasi terdistribusi sehingga dapat mencegah konvergensi dini dan FA memiliki kemampuan konvergensi yang cepat dalam pencarian solusi. Untuk memperbaiki solusi dan mempercepat waktu konvergensi, digunakan metode kombinasi. Pendekatan kombinasi ini meliputi pencarian solusi dengan FA dan pencarian global dengan ACO. Solusi lokal dari FA dinormalisasi dan digunakan untuk menginisialisasi feromon untuk pencarian global ACO. Hasil dari hybrid FA-ACO dibandingkan dengan FA dan ACO. Hasil penelitian menunjukkan bahwa metode yang diusulkan dapat menemukan solusi yang lebih baik tanpa terjebak lokal optimum dengan waktu komputasi lebih pendek.*
Kata kunci: *Traveling Salesman Problem, Firefly Algorithm, Ant Colony Optimization, metode hybrid.*

Abstract. *In this paper, we develop a novel method hybrid firefly algorithm-ant colony optimization for solving traveling salesman problem. The ACO has distributed computation to avoid premature convergence and the FA has a very great ability to search solutions with a fast speed to converge. To improve the result and convergence time, we used hybrid method. The hybrid approach involves local search by the FA and global search by the ACO. Local solution of FA is normalized and is used to initialize the pheromone for the global solution search using the ACO. The outcome are compared with FA and ACO itself. The experiment showed that the proposed method can find the solution much better without trapped into local optimum with shorter computation time.*
Keywords: *Traveling Salesman Problem, Firefly Algorithm, Ant Colony Optimization, hybrid method.*

1. Introduction

The traveling salesman problem (TSP) is a problem in combinatorial optimization studied in operational research, computational mathematics, and artificial intelligence. It can be described simply: a salesman need to find the shortest tour of visiting a set of cities and return to starting city such that each city is visited exactly once. An exact solution can be obtained by finding the possibility of all existing solutions. When a large number of cities is given, it will be hard to get the exact solution, thus, the TSP is NP hard problem. There has not been a reliable method to ensure the optimal solution.

Many methods have been developed to solve TSP such as branch and bound (Land and Doig, 1960), Lin Kernighan local search (Lin and Kernighan, 1973), heuristic search (Jiang et al., 2005), and dynamic programming (Jellouli, 2001). Besides, there are meta-heuristic methods have been proposed, such that GA (Braun, 1991), PSO (Clerc, 2004), ACO (Dorigo et al., 1996), and FA (Jati and Suyanto, 2011). These minimization problems of meta-heuristic methods allow solutions to be found closer to the optimum but with high cost in time.

One of meta-heurisctic methods is Ant Colony Optimization (ACO). ACO is an algorithm that mimics the behavior of ant colonies. ACO was first developed to solve TSP. ACO has distributed computation to avoid premature convergence. But, the more number of cities, the more number of ants and iterations required. Therefore, it converges to the optimal solution slowly.

Several methods are proposed to improve the convergence time of ACO method. Hlaing and Khine (Hlaing and Khine, 2011) adopted candidate set strategy and a dynamic updating rule to improve the convergence time of conventional ACO. While Hassan, et al (Hassan, et al., 2013) proposed a method that uses the concept of ant colony system together with the parallel search of genetic algorithm to obtain the optimal solution quickly for larger TSP problems.

Another method to solve TSP is using Firefly Algorithm (FA). FA is based on the flashing behavior of fireflies. On the application of TSP, each firefly represents one permutation solution. Each firefly moves toward a brighter firefly. Since, a firefly moves with no direction, FA easily trapped into local optimum.

In this regards to improve results and convergence time for solving TSP, we developed a practical combination strategy for two evolutionary algorithms (FA-ACO) based on the ant colony algorithm (ACO) and firefly algorithm (FA). FA is used for local search because of its fast convergence time to find the local solution. While ACO is used for global search to avoid the local optimum and find the optimal solution based on the local solutions by FA. In this way, we not only avoid local optimum situation but also get the better result with faster convergence time.

2. Traveling Salesman Problem

TSP can be stated formally as follows (Lenstra and Rinnooy, 1975). Given a finite set of cities N and element of distance matrix $c_{ij}\left(i, j \in N\right)$ be the distance from city-i to city-j. The objective function is formulated by Equation (1).

$$\min_{\pi} \sum_{i \in N} c_{i\pi(i)} \tag{1}$$

where π runs over cyclic permutation of N and $\pi^{k}\left(i\right)$ is k-th city reached by the salesman from city-i. If $N = \left\{1, 2, \ldots, n\right\}$ then an equivalent formulation is Equation (2).

$$\min\left(\sum_{i=1}^{n-1} c_{v(i)v(i+1)} + c_{v(n)v(1)} \right) \tag{2}$$

where v runs over all permutation of N and $v\left(k\right)$ is k-th city of a tour.

3. Firefly Algorithm

Firefly algorithm (FA) is a meta-heuristic optimization algorithm and nature-inspired algorithm based on the flashing behavior of fireflies. FA was first developed by Xin She Yang (Yang, 2010) for solving continuous optimization problem. FA has adapted by discretizing to solve permutation problem. The evolutionary discrete firefly algorithm (EDFA) has been developed for solving TSP by Jati and Suyanto. FA uses the following three idealized rules.
(1) All firefly are unisex. (2) Attractiveness is proportional to their brightness and decreases as the distance increases. For any two flashing fireflies, the less bright one will move towards the brighter one. If there is no brighter one, then a firefly will move randomly. (3) The brightness (light intensity) of a firefly depends on the objective function.

The intensity of the light emitted by fireflies is affected by distance and the light absorption by air. Light intensity will decrease if the distance is farther. Similarly, the air will also absorb the light so that the light intensity will decrease again. These two factors that cause fireflies limited vision.

Figure 1 illustrated fireflies's movement. Each firefly represents one different solution initially. The closer to the optimum solution, the brighter the light emmited by fireflies. A firefly moves toward the brightest one. In the end, fireflies gather at the same solution.

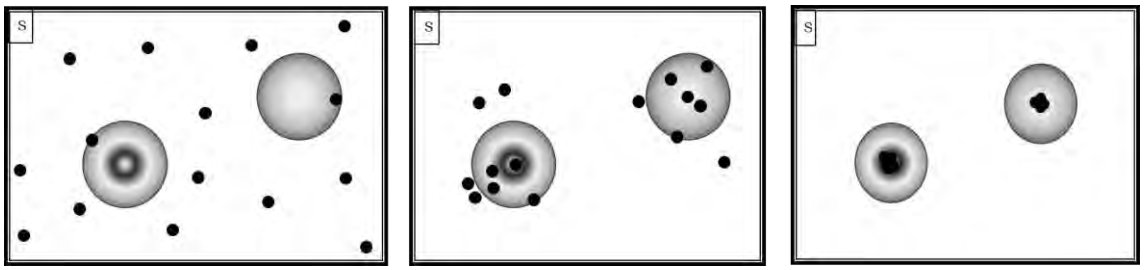


Figure 1. Illustration of fireflies' movement

3.1 Representation of firefly

A firefly represents one permutation solution of TSP as illustrated in Figure 2. In the representation, an element of an array represents a city and the index represents the order of a tour.

3	4	2	8	9	7	5	10	1	6	City
1	2	3	4	5	6	7	8	9	10	Order

Figure 2. Permutation representation of TSP solution

3.2 Light intensity

Light intensity is a value that represents the objective function of a problem. Since the objective function of TSP is to find a route with minimum distance, the light intensity is the invers of total distance produced by a firefly. A firefly which has less distance route, will have a brighter light intensity. Light intensity of a firefly x is calculated as Equation (3).

$$I(x) = \frac{1}{\text{totaldistance}(x)} \tag{3}$$

3.3 Distance

The distance between firefly-i and firefly-j can be defined as the number of different edges between them. The distance can be calculated by using Equation (4).

$$r = \frac{A}{N} \times 10 \tag{4}$$

where r is the distance between any two fireflies, A is the total number of different edges between two fireflies, and N is the number of cities. Equation (4) scales r in the interval [0,10] (Jati et al., 2013).

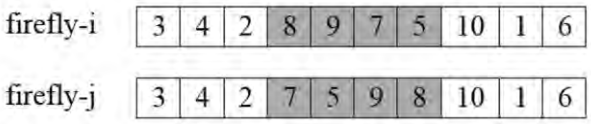


Figure 3. Permutation representation of TSP solution

In Figure 3, four edges 2-8, 8-9, 9-7, 5-10 in firefly-*i* do not exist in firefly-*j*. By using Equation (4), we get the distance between firefly-*i* and firefly-*j* is 4.

3.4 Attractiveness

The attractiveness of firefly-*j* seen by firefly-*i* $\beta(i,j)$ can be any monotonic decreasing function shown as Equation (5).

$$\beta(i,j) = \beta_0 e^{-\gamma r^2} \tag{5}$$

Where $\beta(i,j)$ is the attractiveness of a firefly when seen by other firefly at distance *r*, β_0 is the brightness (light intensity) of a brighter firefly, and γ is a fixed light absorption coefficient.

3.5 Light Absorption

Light absorption coefficient γ determine how fast the convergence of EDFA. It also determines the characteristics of the problems. If $\gamma \rightarrow 0$, then $\beta(i,j) = \beta_0$. Thus, the attractiveness of a firefly will not decrease when viewed by another. Whereas if $\gamma \rightarrow \infty$, then the attractiveness of a firefly will be close to zero. In this case, a firefly cannot be seen. Hence, the selection of γ plays an important role because it will affect the attractiveness of fireflies. In the EDFA, the coefficient γ is in the interval [0.01, 0.15] (Jati et al., 2013).

3.6 Movement

The movement of a firefly-*i* attracted to another more attractive firefly-*j* is determined by Equation (6)

$$x_i = random(1, A) \tag{6}$$

where x_i is the step that must be taken by firefly-*i* to move toward firefly-*j* and *A* is the total number of different edges between two fireflies i and *j*.

Since a firefly in EDFA has no direction to move and represents a permutation solution, it moves using inversion mutation. Thus, it does not deform the previous permutation. This movement makes existing solutions in the firefly are changed.

Each firefly will move using inversion mutation for *m* times. It means, each firefly has *m* new solutions. After *p* fireflies move and produce *pm* new solutions, then the *p* best fireflies will be selected as the new population.

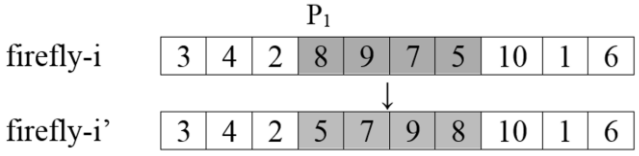


Figure 4. Inversion mutation with length movement (step) = 3

Figure 4 shows the inversion mutation of a firefly. Point P_1 is start point of the inversion mutation. If a firefly moves randomly, P_1 is determined randomly. If a firefly move towards another, P_1 is determined by the first different edges of both fireflies.

4. Ant Colony Optimization

Ant Colony Optimization (ACO) is an optimization algorithm that mimics the behaviorof ant colonies. ACO was first developed by Dorigo (Dorigo, et.al, 1996) to find the shortest path. In the ACO, a total of *m* ants cooperate and communicate using pheromones. In order to solve TSP, each artificial ant has the following characteristics:
(1) Ant chooses the city that will be visited based on the probability function called transition rules. The function depends on the cities distance and the amount of trail present on the connecting edge. (2) Ant is not allowed to visit the same city before a tour is completed. This will be controlled by the tabu list. (3) When an ant finish a tour, it lays a trail of pheromones on each edge (i, j) which have been visited.

Ants use substances called pheromones in communication among individuals. Ants would leave pheromones on the ground thus marking the path with pheromones trail. When there are other ants running randomly, the ants will be able to detect pheromones trail and decide a way that will be passed through the magnitude of probability. Then, the ants also left a trail of pheromones that magnifies the pheromone levels on the path. The more number of ants follow the trail, the more attractive the trail to be followed. Probability ants choose the path increase with the number of ants before choosing the path. Figure 5 represents the ants' movement.

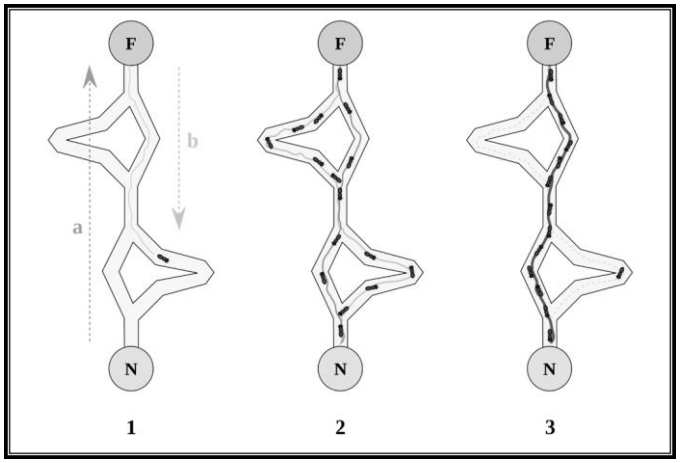


Figure 5. Illustration of ants' movement

4.1 Transition rule

Each ant is placed in a random departure city. Ants will visit one by one city using transition rule and produce a tour.

Given *N* is the set of cities that N = {1, 2,…, *n*} and *m* be the total number of ants. Probability for ant-*k* from the city-*i* to city-*j* is defined as Equation (7).

$$p_{ij} = \begin{cases} \dfrac{\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}(t)\right]^{\beta}}{\sum_{u \in U_k}\left[\tau_{iu}(t)\right]^{\alpha}\left[\eta_{iu}(t)\right]^{\beta}}, j \in U_k \\ 0, \text{otherwise} \end{cases} \quad (7)$$

where $\tau_{ij}(t)$ is the intensity of trail on edge (i, j) at time *t*, visibility η_{ij} is the inverse distance ($\eta_{ij} = 1/d_{ij}$), and U_k is the set of cities that have not been visited by ant-*k*.

The visited cities of each ant are saved in the tabu list. When a tour is completed, the tabu list is used to compute the ant's current solution. The tabu list is then emptied and the trail intensity is updated using global pheromone update rule.

4.2 Pheromone update

The pheromone update rule includes the evaporation on all edges and the addition of pheromones on the edges that are part of the tour. The shorter tour is produced, the more pheromones left by ants on the edges. It implies the edges with a lot of pheromones would be more desirable in the next tours.

After each ant produces a tour of *n* cities, the intensity of trail will be updated with the global pheromone update rules defined as Equation (8).

$$\tau_{ij}(t+n) = \rho\tau_{ij}(t) + \sum_{k=1}^{m}\Delta\tau_{ij}^{k} \quad (8)$$

where ρ is the coefficient such that $(1-\rho)$ represents the evaporation of trail between time *t* and *t* + *n*, and $\Delta\tau_{ij}^{k}$ is the quantity of pheromones laid by ant-*k* on the edge (i, j) is formulated by Equation (9).

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L_k}, \text{if ant-}k \text{ uses } (i, j) \\ 0, \text{otherwise} \end{cases} \quad (9)$$

where Q is a constant and L_k is the tour length of ant-*k*.

The shortest route found by the ants is saved and the tabu list is emptied to save the cities on the next tour. This process will be repeated until maximum condition.

5. Hybrid FA-ACO

To minimize the time of convergence which is due to the high number of the agents and iterations, we proposed a hybrid method with the combination of FA and ACO with a lower number of ants and fireflies as possible. In this method, FA plays as local search and ACO used to find the global solution. FA is implemented to find a local solution because the FA has fast convergence capability.

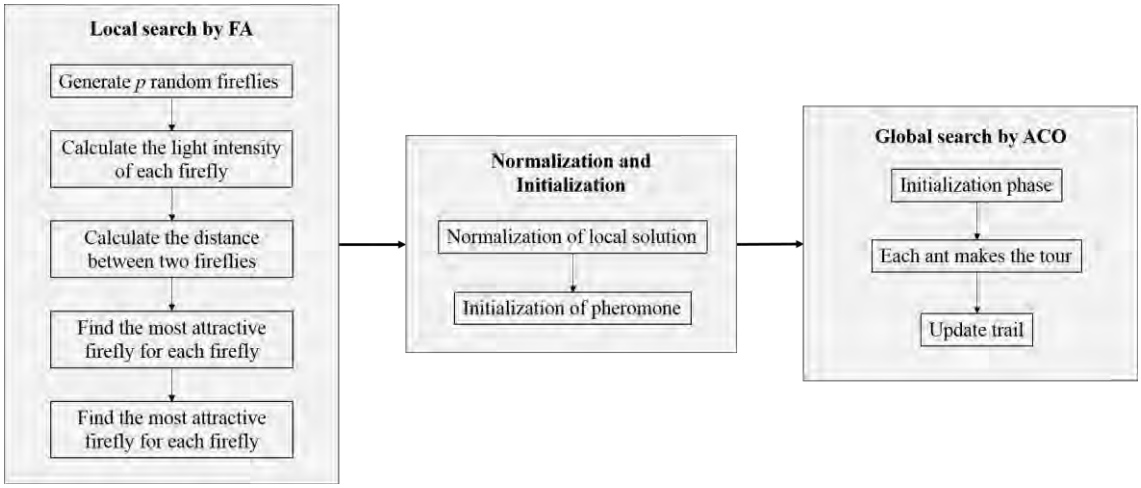


Figure 6. Phase of proposed method

Figure 6 shows the phase of the proposed method and explained as follows. Firstly, FA is implemented to find p local solutions of $pm + p$ solutions. The local solutions is obtained by generating p random firefly as first population. Then, calculate the attractiveness of each firefly when seen by another firefly. Before that, calculate the light intensity and the distance between two fireflies. Each firefly finds the most attractive firefly and moves toward. A firefly will move m times using inversion mutation. Finally, there are $pm + p$ solutions. Choose p best fireflies from $pm + p$ solutions as a new population for next iteration.

Code 1. Pseudocode hybrid FA-ACO for TSP

```
Input:
FA: p number of fireflies, light absorption γ, m moves.
ACO: n number of ants, parameter α, β, ρ coefficient, Q constant, Δτ_ij^k = 0

Begin
  % Local search by FA
  Generate p random fireflies.
  Calculate light intensity of p fireflies based on the objective function.
  Repeat
    temp = p firefly;
    Calculate the distance of two fireflies using Equation (4);
    Find attractiveness varies at distance r using Equation (5);
    for i = 1 to p do (p fireflies)
      Get most attractive firefly j, where i≠j
      if there is most attractive firefly j, move firefly i toward j for m
      times; add solution to temp;
      else move firefly i random for m times; add solution to temp;
      end
    end
    Evaluate new solution and update light intensity;
    Rank the fireflies in temp and select p best firelies.
  until (stopping condition is satisfied)

  % initialize pheromone for ACO based on the best solution found by FA
  Add pheromone on each edge of p firefly.

  % Global search by ACO
  Place n ants on n city
  While (stopping condition is not satisfied)
    for k = 1 to n do
      Place the starting town of ant-k on tabu_k(1)
```

```
    end
    repeat
    for k = 1 to n do
      Choose city-j with probability in Equation (7)
      Move ant-k to the city j
      Insert city j in tabu_k(s)
    end
    until tabu list is full (tabu_k(n))
    for k = 1 to n do
      Move the ant-k from tabu_k (n) to tabu_k(1)
      Compute the length L_k of the tour described by ant-k
      Calculate pheromone addition produced by ant-k
    end
    Update the shortest tour found
    Update trail set Δτ_ij^k = 0
    Empty all tabu list
  end
  Print the shortest tour
End
```

Secondly, in the end of FA's iteration, there will be *p* local solutions. From the *p* local solutions, do normalization by finding *q* different representations of firefly as candidate tours. Fireflies which have the same permutation solution will be counted as a single solution. Then, set initial trail of pheromone by adding the pheromone on the edges which are part of the *q* candidate tours. Edges that belong to best solution normalization will get the most pheromone addition. Edges on second best solution get less pheromone than the first one. Thus, the most frequently passed edges on candidate tour will get the most pheromone addition. This initial trail will be used to run ACO. By setting the initial trail, it will be minimize the agents and iterations for ACO. Third, ACO implemented to find the global solution. The output is the shortest tour. The pseudocode of the proposed method is shown as Code 1.

6. Experimental results

The experiment is examined for four TSP instances obtained from TSPLIB. The type of TSP instances in TSPLIB is based on Euclidian distances, wherein a TSP instance provides some cities with their coordinates. The number in the name of an instance represents the number of provided cities. For example, ulysses16 provides 16 cities with their coordinates.

The experiment is implemented using Matlab and run on computer with specification Processor Intel(R) Core(TM) i7 CPU 2.20 GHz and 4.00 GB RAM. Table 1 and Table 2 present the result of FA, ACO, and hybrid FA-ACO applied to the problems. The results reported are averaged after running the experiment 10 times.

Table 1. Comparison of best solution known hybrid FA-ACO with FA and ACO.

Problem	Best Solution	Best solution known		
		FA	ACO	Hybrid FA-ACO
ulysses16	6859	6870	6909	**6859**
oliver30	412	412	412	**412**
berlin52	7542	7718	7570	**7542**
pr76	108159	129550	117174	**110337**

Table 2. Comparison hybrid FA-ACO with FA and ACO. Results are averaged over 10 runs.

Problem	FA		ACO		Hybrid FA-ACO	
	Average solution	Average computation time	Average solution	Average computation time	Average solution	Average computation time
ulysses16	6981.4	**1.30057**	6925.5	5.8429	**6890**	3.61989
oliver30	434.1	**1.39531**	416.4	7.6767	**415**	3.98507
berlin52	8549.4	**8.4653**	7720.1	78.053	**7719.8**	23.8794
pr76	137561.3	**15.2506**	118951.9	126.709	**115339.5**	46.1076

In order to evaluate the performance of the proposed method, we compare the solution obtained by the method with best solution from dataset. From Table 1, hybrid FA-ACO obtains the nearest solution to the best solution for ulysses16, oliver30, berlin52, and pr76. For ulysses16, oliver30, berlin52, hybrid FA-ACO get the exact solution. It is noticed that hybrid FA-ACO yield better solution than FA and ACO for all problems.

Table 2 shows the average solution and average time converge after running the experiments 10 times. From Table 2, average solution by FA is higher than ACO. To the fact that FA is easily trapped into local optimum. Meanwhile, hybrid FA-ACO get the lowest average solution for all problems. Related to average computation time, hybrid FA-ACO runs faster than ACO, but slower than FA.

For oliver30 problem, the parameter settings are described as follows: light absorption γ = 0.05, α = 1, β = 5, ρ = 0.05, and Q = 100. For running FA, we used 7 fireflies with 7 movements and 700 iterations. For runnning ACO, 30 ants and 500 iterations are required. While for hybrid FA-ACO, the local search by FA needs 4 fireflies with 4 movements and 400 iterations, the global search by ACO requires 20 ants and 300 iterations.

Figure 7 shows the comparison of convergence time among three methods for oliver30 problem and more detailed shown as Figure 8. FA has the fastest time convergence but the worst solution while hybrid FA-ACO get the best solution and faster convergence time than ACO. It means that FA has fast speed to converge but easy trapped into local optimum. These results clearly showed that hybrid FA-ACO can find the solution much better without trapped into local optimum with shorter computation time.

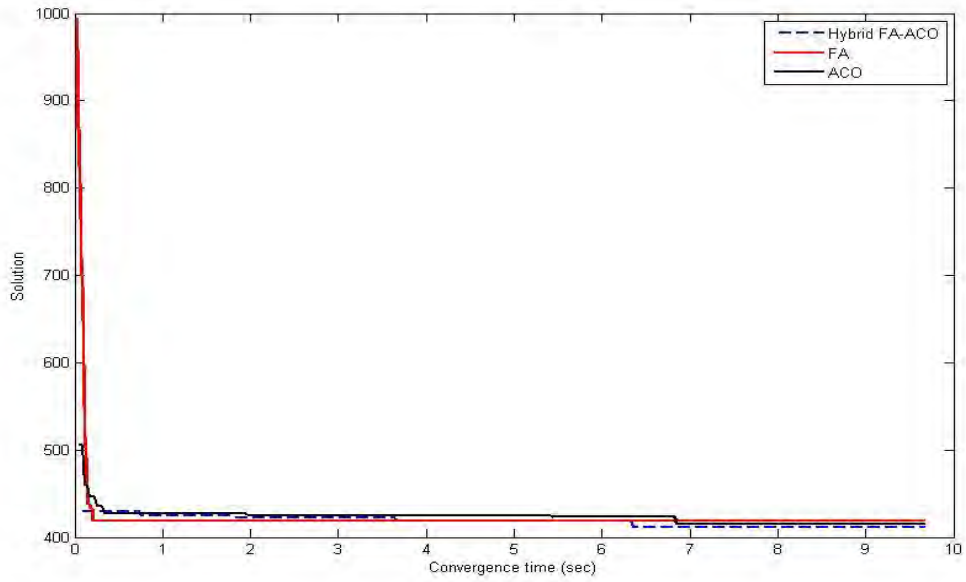


Figure 7. Comparison of convergence time for oliver30

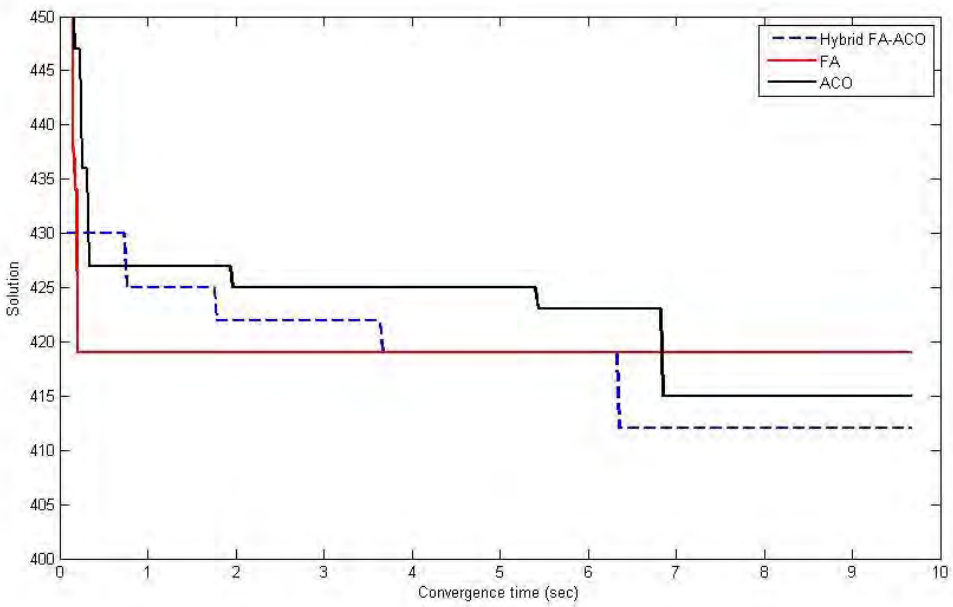


Figure 8. More detailed comparison of convergence time for oliver30

7. Conclusion

In this paper, we presented the hybrid FA-ACO method for solving the traveling salesman problem. The proposed method tries to combine FA for local search and ACO for global search. The local solution obtained by FA is used to initialize the pheromone for global search by ACO. The goal is to minimize the time of convergence with minimum number of the agents and iterations.

The experiment results showed that the hybrid FA-ACO is able to provide the better solution than FA and ACO. In our experiment, the proposed method can find the best solution without trapped into local optimum. In addition, it performs well with shorter computation time.

8. References

Braun, H. 1991. On Solving Travelling Salesman Problems by Genetic Algorithm. In: *1st Workshop, PPSN I Dortmund, FRG, October 1–3, 1990 Proceedings, vol. 496*, Springer-Verlag, Berlin, Heidelberg, Lecture Notes in Computer Science, 129-133.

Clerc, M. 2004. Discrete Particle Swarm Optimization, illustrated by Traveling Salesman Problem. *New Optimization Techniques in Engineering*, Springer-Verlag, Berlin, Heidelberg, Studies in Fuzziness and Soft Computing vol. 141, 219-239.

Dorigo, M., Maniezzo, V, and Colorni, A. 1996. The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on System, Man, and Cybernetics-Part B: Cybernetics*, 26(1): 29-41.

Hassan, M.R., Hasan, M.K., and Hashem, M.M.A. 2013. An Improved ACS Algorithm for the Solutions of Larger TSP Problems. In: *Proceedings of the ICEECE December 22-24, Dhaka, Bangladesh.*

Hlaing, Z.C.S.S. and Khine, M.A. December 2011. Solving Traveling Salesman Problem by Using Improved Ant Colony Optimization Algorithm. *International Journal of Information and Education Technology,* 1(5): 404-409.

Jati, G.K. and Suyanto. 2011. Evolutionary Discrete Firefly Algorithm for Travelling Salesman Problem. In: *Proceedings of Second International Conference ICAIS 2011, LNAI 6943*, Eds. Bouchachia, A., University of Klagenfurt, Klagenfurt, 393-403.

PENYELESAIAN MULTI-DEPOT MULTIPLE TRAVELING SALESMAN PROBLEM MENGGUNAKAN HYBRID FIREFLY ALGORITHM – ANT COLONY OPTIMIZATION

OLIEF ILMANDIRA RATU FARISI

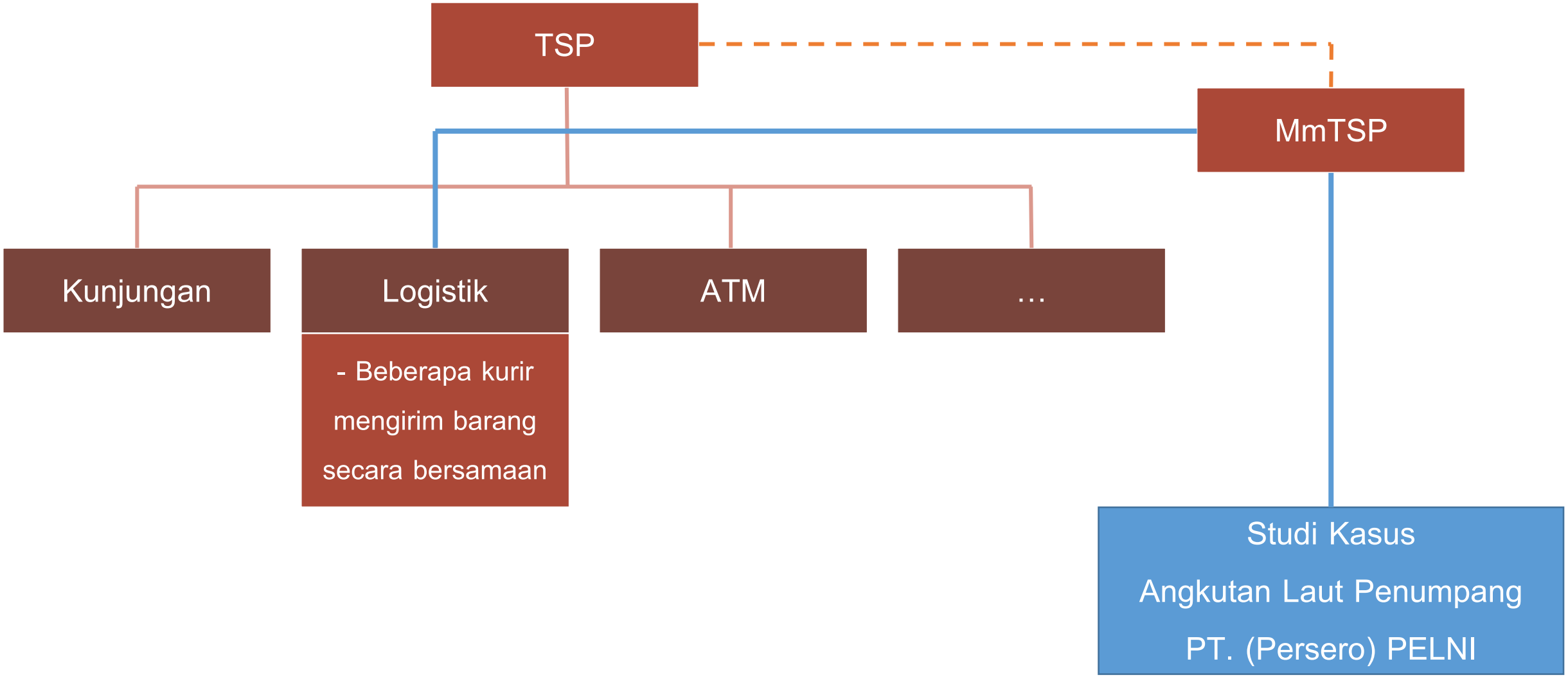
1213201010

Dosen Pembimbing:

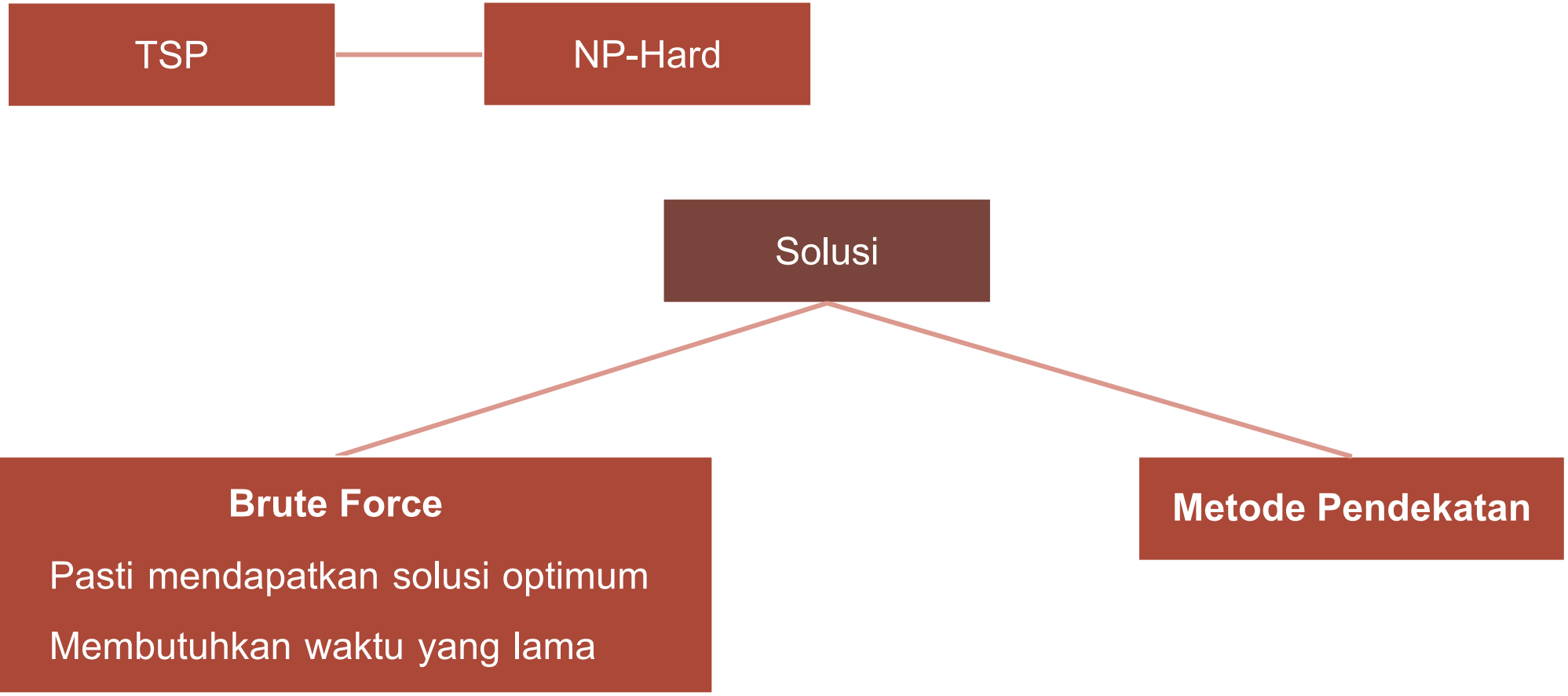
Dr. Budi Setiyono, S.Si, MT

Dr. Ir. R. Imbang Danandjojo. MT

LATAR BELAKANG



LATAR BELAKANG



LATAR BELAKANG

Ghafurian dan Javadian (2011) → ACO MmTSP

- ACO menghasilkan solusi mendekati solusi eksak yang didapat dengan Lingo 8.0
- Waktu komputasi ACO untuk mencapai solusi optimum jauh lebih cepat dibandingkan Lingo 8.0.

LATAR BELAKANG

Marco Dorigo, dkk (1996) → ACO TSP

- Mengadopsi cara kerja koloni semut dalam mencari makanan.
- Menggabungkan antara konsep proses autokatalis dan pencarian greedy, sehingga tidak mudah terjebak pada solusi lokal optimum.
- Semakin banyak objek pada TSP, semakin banyak semut yang diperlukan, semakin banyak pula iterasinya.

LATAR BELAKANG

Gilang Jati dan Suyanto (2011) → FA

- Mengadopsi cara kerja kunang-kunang. Kunang-kunang bergerak menuju kunang-kunang yang memiliki tingkat ketertarikan lebih besar.
- Setiap kunang-kunang merepresentasikan satu permutasi solusi.
- Dalam pergerakannya, kunang-kunang tidak memiliki arah, sehingga mudah terjebak solusi lokal optimum.

LATAR BELAKANG

ACO (Dorigo, dkk, 1996)

(+) Tidak mudah terjebak solusi lokal optimum

(-) Waktu konvergensinya lama

FA (Jati, G. dan Suyanto, 2011)

(+) Waktu konvergensinya cepat

(-) Mudah terjebak solusi lokal optimum

HYBRID FA-ACO

RUMUSAN MASALAH

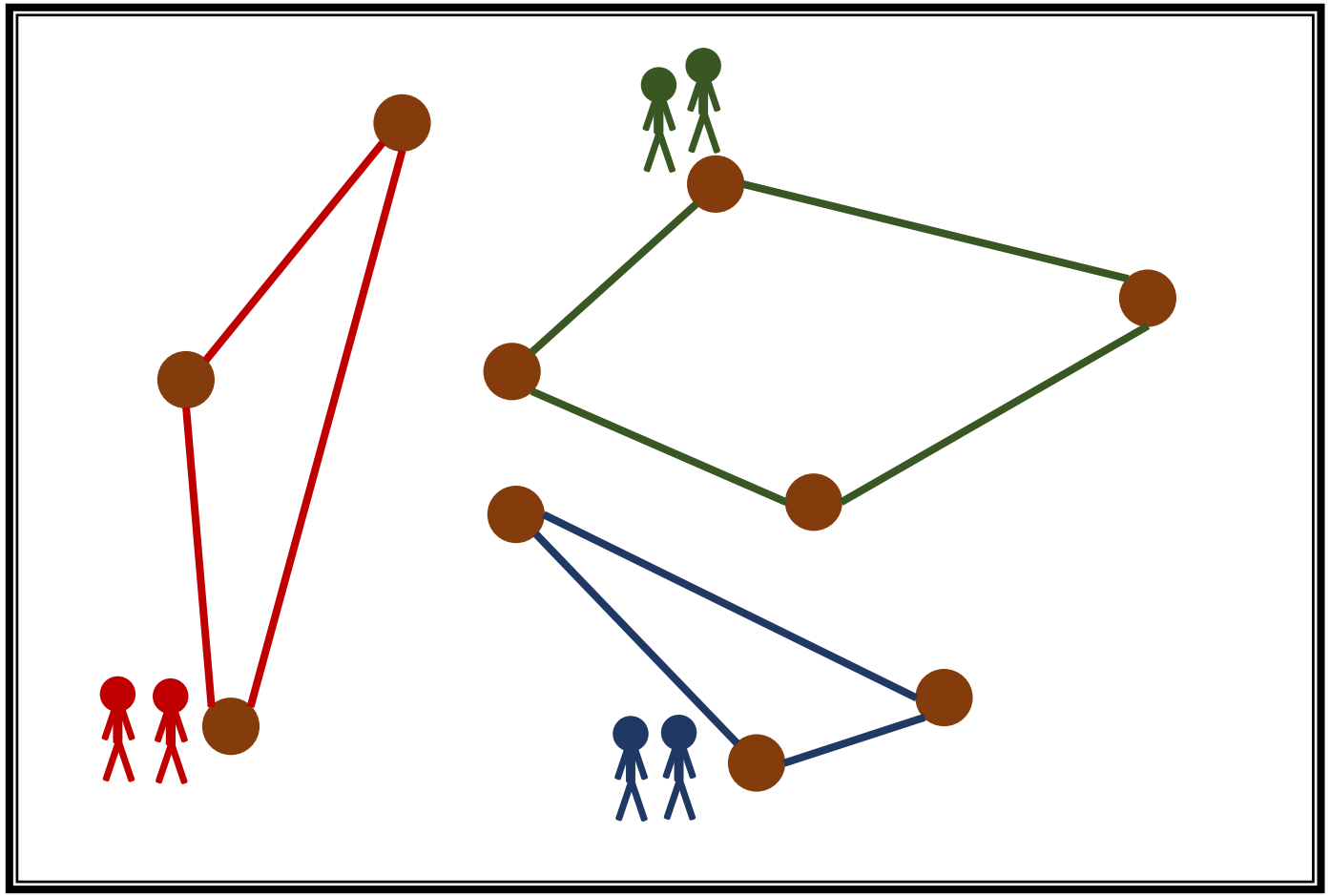
Bagaimana menyelesaikan MmTSP menggunakan *hybrid* FA-ACO?

Bagaimana performansi komputasi *hybrid* FA-ACO dibandingkan dengan ACO?

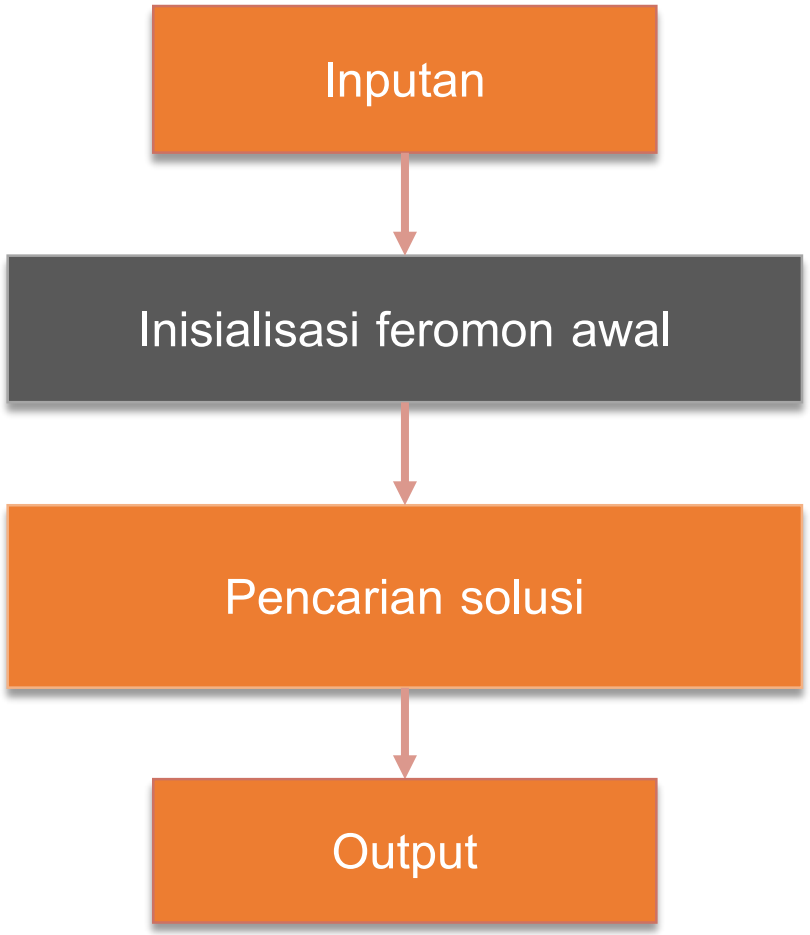
DASAR TEORI

Multi-Depot Multiple Traveling Salesman Problem (MmTSP)

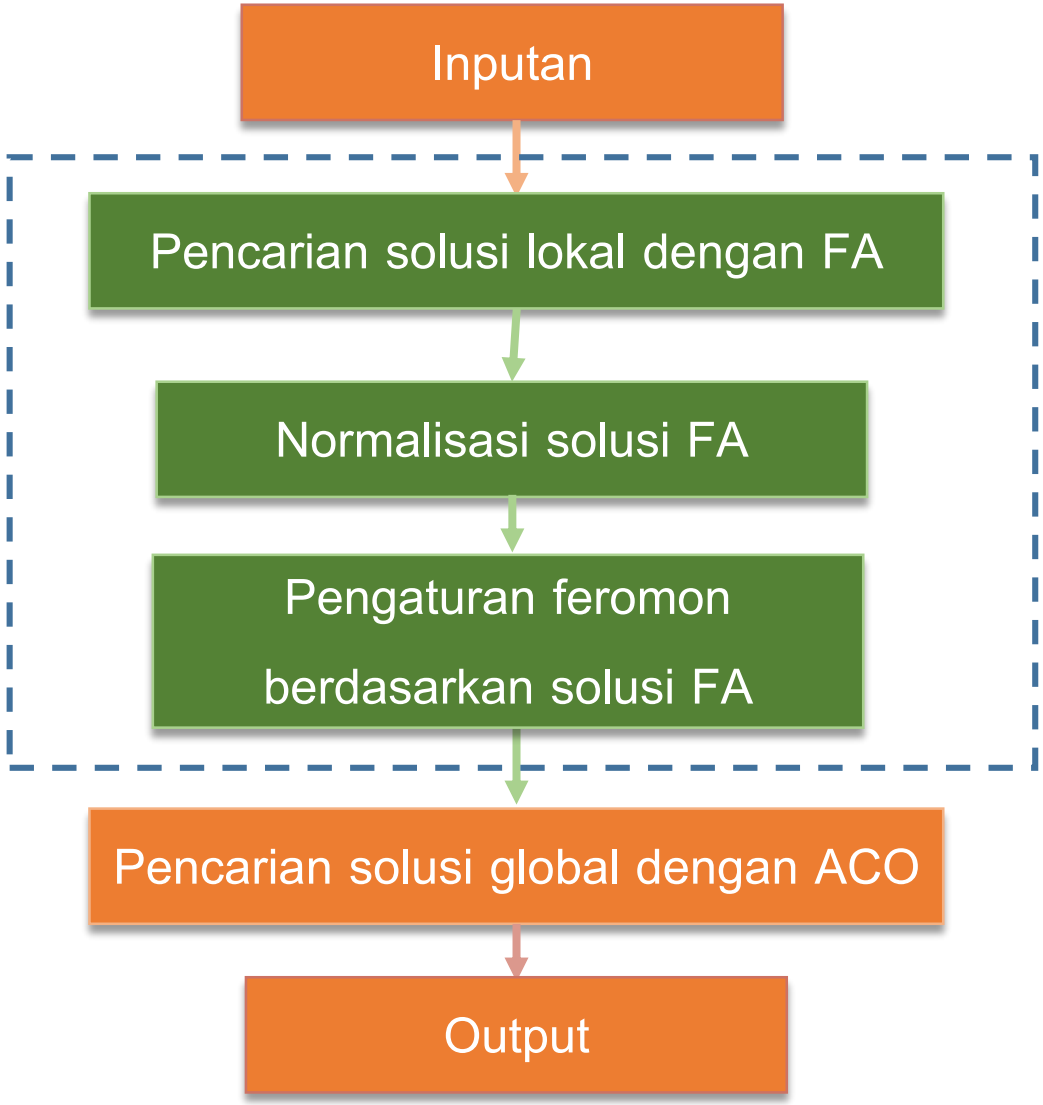
- Terdapat lebih dari satu orang salesman
- Beberapa salesman berangkat dari kota yang berbeda-beda yang disebut depot, dan harus kembali ke depot tersebut.
- Setiap salesman membentuk rutenya sendiri-sendiri
- Semua kota harus dilewati
- Satu kota hanya boleh dikunjungi oleh satu salesman
- Total bobot yang dilalui semua salesman harus minimum.



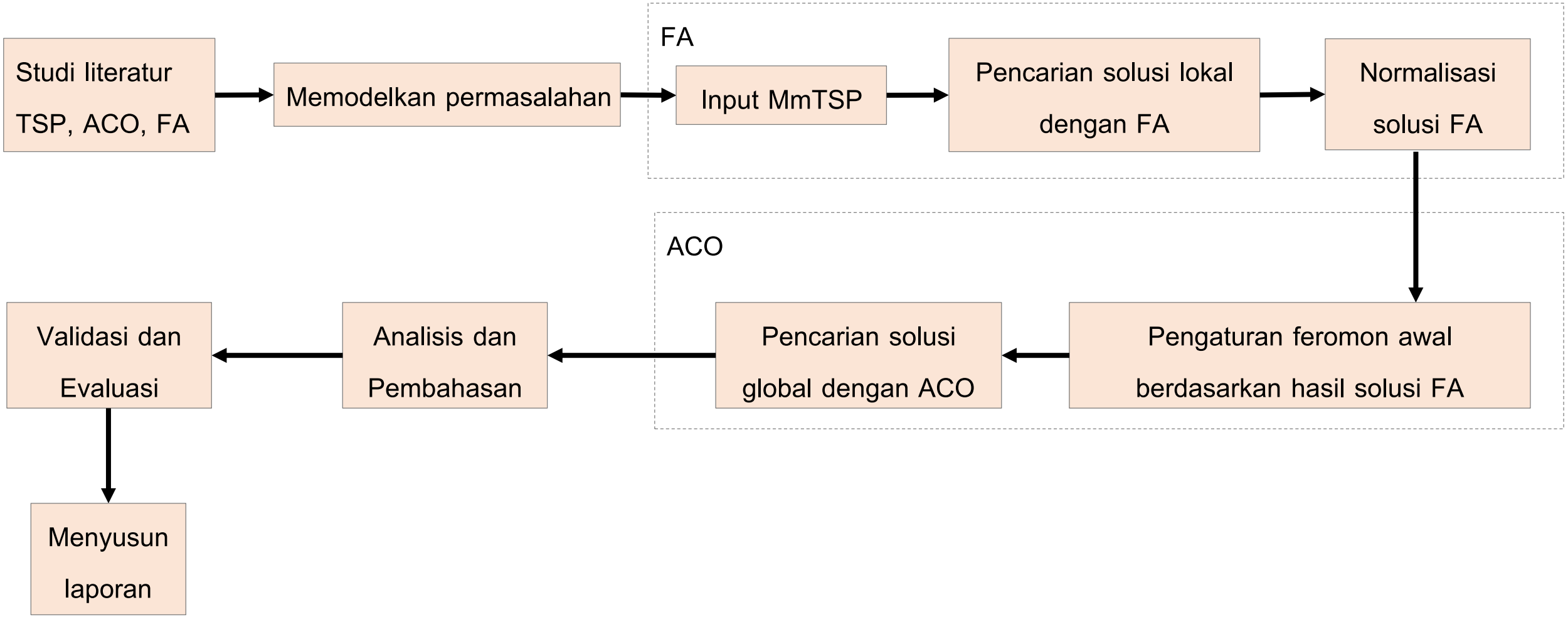
ACO



HYBRID FA-ACO



METODE PENELITIAN



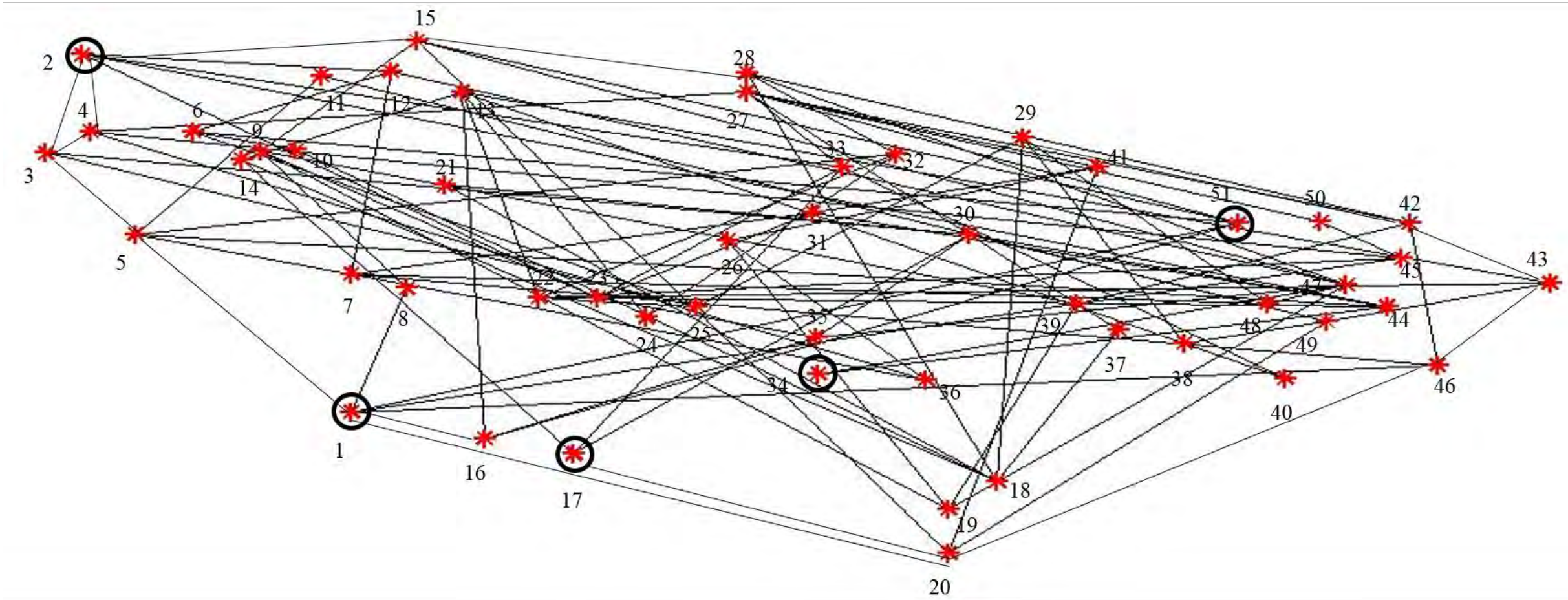
STUDI KASUS

- Data angkutan laut penumpang PT. Persero PELNI diperoleh dari disertasi Imbang (2014)
- Data berupa jarak antar 51 pelabuhan di Indonesia.
- Fungsi objektifnya adalah menentukan rute terpendek.
- Kendalanya adalah hanya terdapat satu kapal yang berangkat dari depot yang berbeda, dan setiap pelabuhan hanya dapat menampung satu kapal

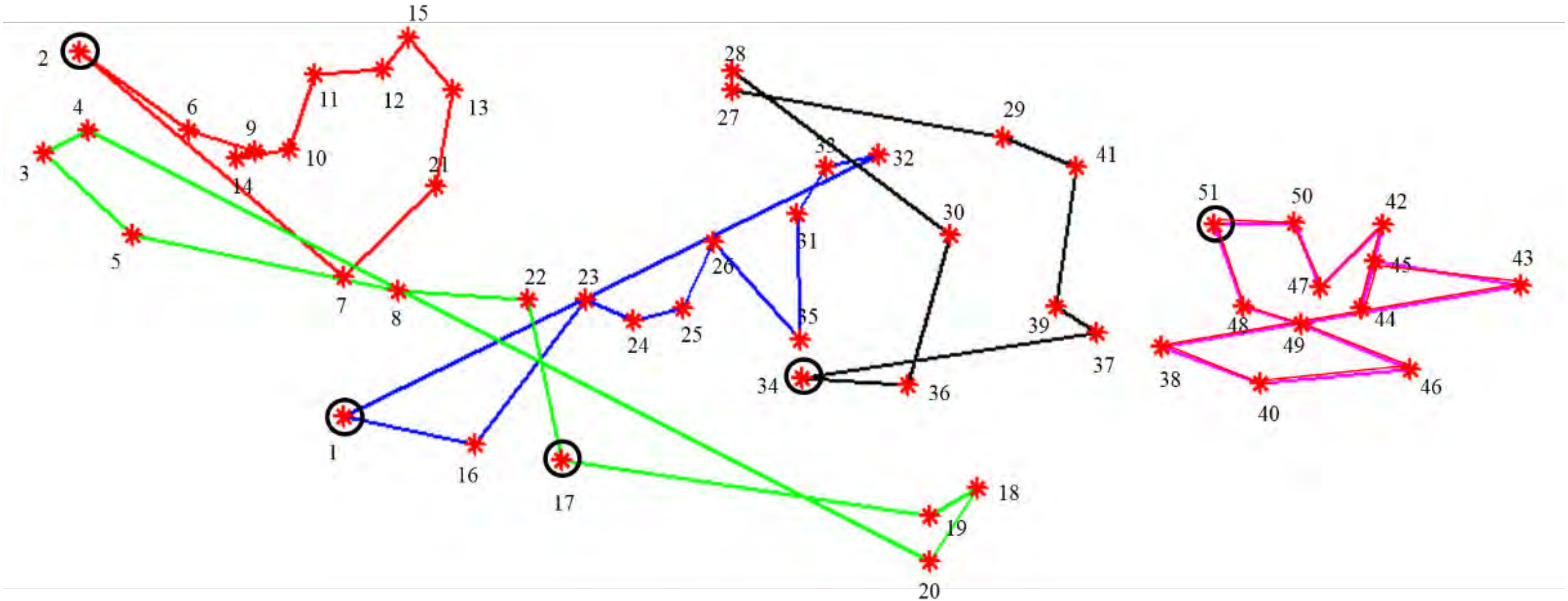
STUDI KASUS


Natuna
Letung
Midai
Serasan
Tarakan
Nunukan
Lokodidi
Toli-toli
Bitung
Ternate
Gunung Sitoli
Sibolga
Dumai
Batam
Kijang
Tanjung Balai Karimun
Pontianak
Banggai
Donggala
Manokwari
Biak
Teluk Bayur
Belinyu
Balikpapan
Jayapura
Wasior
Serui
Nabire
Namlea
Tanjung Pandan
Kumai
Sampit
Batu Licin
Banjarmasin
Parepare
Ambon
Fakfak
Kaimana
Banda
Tual
Tanjung Mas
Makassar
Bau-Bau
Timika
Kalabahi
Larantuka
Tenau

STUDI KASUS


15
2
28
4
6
11
12
13
27
29
9
33
32
41
3
10
21
14
51
50
42
30
5
31
26
43
45
22
23
7
8
35
39
48
44
24
25
49
37
38
46
34
36
40
1
16
17
18
19
20

STUDI KASUS


15
2
28
4
6
11
12
13
27
29
9
10
41
3
32
21
14
30
51
50
42
31
26
22
23
43
5
7
8
45
47
39
24
25
35
48
44
49
37
38
34
36
46
40
1
16
17
18
19
20

MODEL MmTSP

- Diberikan himpunan pelabuhan $N = \{1,2, \dots, 51\}$, matriks $c_{ij}, (i, j \in N)$ menyatakan jarak dari pelabuhan-i ke pelabuhan -j. N dipartisi menjadi $N = N' \cup D$, dengan $d \in \{1,2,17,34,51\}$ adalah anggota himpunan *depot* D. Terdapat 1_i kapal ditempatkan di masing-masing *depot*-i. Diberikan $N' = \{d+1, d+2, \dots, n\}$ adalah himpunan pelabuhan yang akan dikunjungi.
- Didefinisikan variabel biner x_{ijk} bernilai 1 jika kapal depot-k berangkat dari pelabuhan-i pelabuhan-j ke dalam turnya dan x_{ijk} bernilai 0 jika sisi (i, j) tidak dilewati. Diasumsikan terdapat batas bawah L dan batas atas U. Untuk sebarang kapal, u_i adalah banyaknya pelabuhan yang dilalui dari asalnya sampai pelabuhan-i. Sehingga, $1 \leq u_i \leq 11, \forall i \geq 2$. Jika $x_{ikk} = 1$, maka $8 \leq u_i \leq 11$ harus memenuhi

$$\min\left(\sum_{k\in D}\sum_{j\in N'}\left(c_{kj}x_{kjk} + c_{jk}x_{jkk}\right) + \sum_{k\in D}\sum_{i\in N'}\sum_{j\in N'} c_{ij}x_{ijk}\right)$$

MODEL MmTSP

dengan kendala

- Tepat 1_k kapal berangkat dari masing-masing *depot* $k \in D$.

$$\sum_{j \in N'} x_{kjk} = 1_k, k \in D$$

- Setiap pelabuhan dikunjungi tepat satu kali.

$$\sum_{k \in D} x_{kjk} + \sum_{k \in D} \sum_{i \in N'} x_{ijk} = 1, j \in N'$$

- Kontinuitas rute untuk pelabuhan yang dikunjungi.

$$x_{kjk} + \sum_{i \in N'} x_{ijk} - x_{jkk} - \sum_{i \in N'} x_{jik} = 0, k \in D, j \in N'$$

- Kontinuitas rute untuk *depot*.

$$\sum_{j \in N'} x_{kjk} - \sum_{j \in N'} x_{jkk} = 0, k \in D$$

MODEL MmTSP

- Batas atas tur.

$$u_i + 9\sum_{k\in D} x_{kik} + \sum_{k\in D} x_{ikk} \leq 10, i \in N'$$

- Batas bawah tur.

$$u_i + \sum_{k\in D} x_{kik} + (-6)\sum_{k\in D} x_{ikk} \geq 2, i \in N$$

- Tur tidak diperbolehkan hanya satu pelabuhan.

$$\sum_{k\in D} x_{kik} + \sum_{k\in D} x_{ikk} \leq 1, i \in N'$$

- Eliminasi sub tur. Sub tur adalah tur tanpa *depot* sebagai pelabuhan awal atau pelabuhan akhir yang mungkin terdapat pada solusi.

$$u_i - u_j + U\sum_{k\in D} x_{ijk} + 9\sum_{k\in D} x_{jik} \leq 10, i \neq j;\ i, j \in N$$

MODEL MmTSP

- Batas bawah L dan batas atas U harus memenuhi:

- L adalah bilangan bulat acak sedemikian hingga $2 \leq L \leq \frac{n-d}{d}$
- U adalah bilangan bulat acak sedemikian hingga $\frac{(n-d)}{m} \leq L \leq n-d$.
- m adalah bilangan bulat acak sedemikian hingga $d \leq m \leq \frac{n-d}{K}$.

Dimana n adalah banyaknya pelabuhan dan d adalah banyak depot.

PENCARIAN SOLUSI LOKAL

Langkah 1: Membentuk p representasi kunang-kunang secara random.

Kunang-kunang-1

1	38	35	21	29	10	16	8	3	28	9	1	2	11
6	48	43	22	25	36	30	4	2	17	13	31	23	32
15	12	20	44	5	17	34	41	49	33	47	46	42	24
26	50	27	34	51	40	37	7	18	14	39	45	19	51

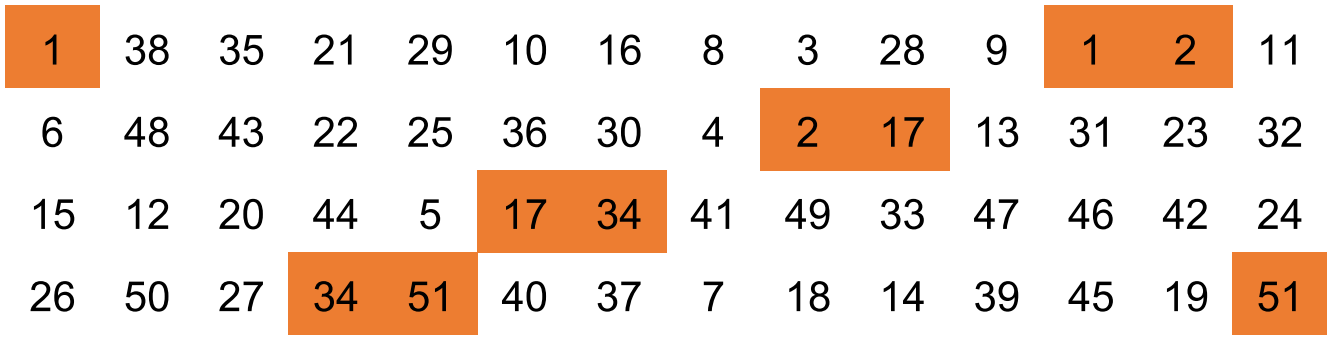
Kunang-kunang-2

1	38	35	21	29	10	16	8	3	28	9	1	2	11
6	48	43	20	5	12	44	13	15	2	17	25	22	41
49	36	31	24	30	33	42	17	34	4	47	46	23	32
26	50	27	34	51	40	37	7	18	14	39	45	19	51

PENCARIAN SOLUSI LOKAL

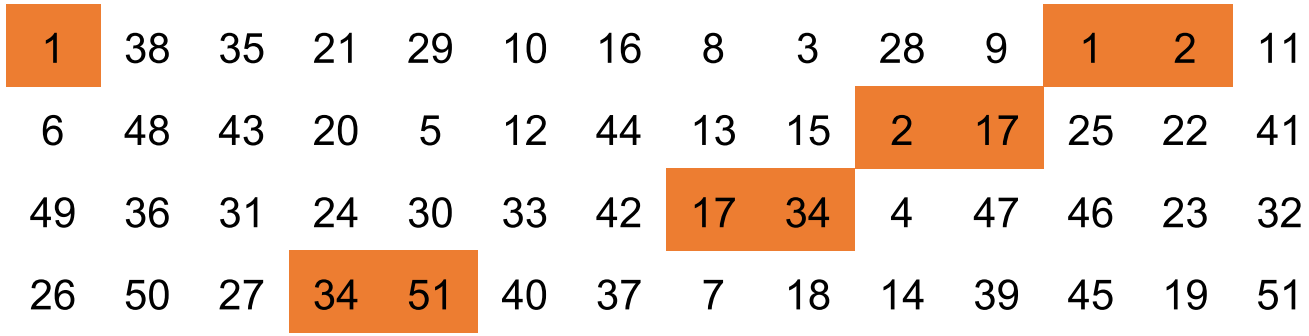
Langkah 2: Menghitung jarak rute yang dihasilkan oleh masing-masing kunang-kunang

Kunang-kunang-1



52219 km

Kunang-kunang-2



56165 km

PENCARIAN SOLUSI LOKAL

Langkah 3: Menghitung intensitas cahaya

Intensitas cahaya merupakan invers dari total jarak yang dihasilkan kunang-kunang.

Kunang-kunang 1: $\frac{1}{52219} = 0.00001915$

Kunang-kunang 2: $\frac{1}{56165} = 0.0000178$

PENCARIAN SOLUSI LOKAL

- ***Langkah 4: Mencari jarak antar kunang-kunang***

Jarak antar kunang-kunang dapat dihitung dari pelabuhan-pelabuhan yang dapat dikunjungi selain depot.

Urutan pelabuhan yang dikunjungi kunang-kunang 1

38	35	21	29	10	16	8	3	28	9	11	6	48	43	22	25
36	30	4	13	31	23	32	15	12	20	44	5	41	49	33	47
46	42	24	26	50	27	40	37	7	18	14	39	45	19		

Urutan pelabuhan yang dikunjungi kunang-kunang 2

38	35	21	29	10	16	8	3	28	9	11	6	48	43	20	5
12	44	13	15	33	41	22	36	31	49	24	30	25	42	4	47
46	23	32	26	50	27	40	37	7	18	14	39	45	19		

20 perbedaan sisi

Jarak =

$$\frac{20}{51} \times 10 = 3.9216$$

PENCARIAN SOLUSI LOKAL

- ***Langkah 5: Menghitung daya tarik kunang-kunang***

$$\beta(i,j) = \beta_0 e^{-\gamma r^2}$$

γ = koefisien penyerapa cahaya; r = jarak antar dua kunang-kunang

Daya tarik suatu kunang-kunang dengan dirinya sendiri sama dengan intensitas cahaya yang dimilikinya.

- Daya tarik kunang-kunang-1 terhadap kunang-kunang-2

$$\beta(2,1) = \beta_0 e^{-\gamma r^2} = I_1 e^{-0.005 \cdot 3.9216} = 0.00001915 e^{-0.005 \cdot 3.9216} = 0.00001886$$

- Daya tarik kunang-kunang-2 terhadap kunang-kunang-1

$$\beta(1,2) = \beta_0 e^{-\gamma r^2} = I_2 e^{-0.005 \cdot 3.9216} = 0.0000178 e^{-0.005 \cdot 3.9216} = 0.000017533$$

PENCARIAN SOLUSI LOKAL

- ***Langkah 6: Kunang-kunang mencari kunang-kunang lain dengan daya tarik besar***

Daya tarik kunang-kunang-1 terhadap dirinya sendiri $= 0.00001915$

Daya tarik kunang-kunang-2 terhadap kunang-kunang-1$= 0.000017533$

$0.000017533 < 0.00001915$, kunang-kunang 1 bergerak secara random

Daya tarik kunang-kunang-2 terhadap dirinya sendiri $= 0.0000178$

Daya tarik kunang-kunang-1 terhadap kunang-kunang 2 $= 0.00001886$

$0.00001886 > 0.0000178$, kunang-kunang 2 bergerak mendekati kunang-kunang 1

PENCARIAN SOLUSI LOKAL

- ***Langkah 7: Pergerakan kunang-kunang menggunakan mutasi invers***
- *Pergerakan = rand(1,20) = 16*

P1

38	35	21	29	10	16	8	3	28	9	11	6	48	43	20	5
12	44	13	15	33	41	22	36	31	49	24	30	25	42	4	47
46	23	32	26	50	27	40	37	7	18	14	39	45	19		

38	35	21	29	10	16	8	3	28	9	11	6	48	43	4	42
25	30	24	49	31	36	22	41	33	15	13	44	12	5	20	47
46	23	32	26	50	27	40	37	7	18	14	39	45	19		

PENCARIAN SOLUSI LOKAL

- ***Langkah 8: Menghitung jarak rute yang dihasilkan oleh kunang-kunang baru***

Kunang-kunang-3

1	38	35	21	29	49	41	5	44	20	1	2	12	15
32	23	31	13	4	30	36	2	17	25	22	43	48	6
11	9	28	3	8	17	34	16	10	33	47	46	42	24
26	50	34	51	27	40	37	7	18	14	39	45	19	51

54628 km

Kunang-kunang-4

1	38	35	21	29	10	16	8	3	28	9	1	2	11
6	48	43	4	42	25	30	24	2	17	49	31	36	22
41	33	15	13	44	12	17	34	5	20	47	46	23	32
26	50	34	51	27	40	37	7	18	14	39	45	19	51

58762 km

PENCARIAN SOLUSI LOKAL

- ***Langkah 9: Memilih populasi kunang-kunang untuk iterasi selanjutnya***

kunang-kunang-1 = 52219

kunang-kunang-2 = 56165

kunang-kunang-3 = 54628

kunang-kunang-4 = 58762

NORMALISASI DAN INISIALISASI FEROMON

- ***Langkah 1: Normalisasi solusi lokal FA***

	1	8	22	37	45	44	50	41	19	21	7	1	2	9
Kunang	10	13	12	15	6	14	16	18	2	17	35	33	29	30
kunang-1	36	24	23	26	17	34	5	3	4	20	39	32	31	28
	27	38	34	51	42	47	48	49	46	40	11	25	43	51

	1	8	22	37	45	44	50	41	19	21	7	1	2	9
Kunang	10	13	12	15	6	14	16	18	2	17	35	33	29	30
kunang-2	36	24	23	26	17	34	5	3	4	20	39	32	31	28
	27	38	34	51	42	47	48	49	46	40	11	25	43	51

Hasil normalisasi

1	8	22	37	45	44	50	41	19	21	7	1	2	9
10	13	12	15	6	14	16	18	2	17	35	33	29	30
36	24	23	26	17	34	5	3	4	20	39	32	31	28
27	38	34	51	42	47	48	49	46	40	11	25	43	51

NORMALISASI DAN INISIALISASI FEROMON

- ***Langkah 2: Inisialisasi feromon awal***

Tambahkan feromon pada sisi (i, j) yang merupakan bagian dari solusi normalisasi.

$$\Delta t_{i,j} = \frac{10}{q} \times (q - k + 1)$$
$$= \frac{10}{1} \times (1 - 1 + 1) = 10$$

	1	2	3	4	5	6	7	8	9		44	45	46	47	48	49	50	51
1	1	1	1	1	1	1	1	11	1	⋯	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	11	⋯	1	1	1	1	1	1	1	1
3	1	1	1	11	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
5	1	1	11	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
7	11	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
9	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
44	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	11	1
45	1	1	1	1	1	1	1	1	1	⋯	11	1	1	1	1	1	1	1
46	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
47	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	11	1	1	1
48	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	11	1	1
49	1	1	1	1	1	1	1	1	1	⋯	1	1	11	1	1	1	1	1
50	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
51	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1

PENCARIAN SOLUSI GLOBAL

- ***Langkah 1: Posisikan n semut pada salah satu depot***

Misalkan semut ditempatkan di depot -1

PENCARIAN SOLUSI GLOBAL

- ***Langkah 2: Menghitung visibilitas***

Visibilitas merupakan invers dari jarak

	1	2	3	4	⋯	50	51
1	∞	0.001159	0.001295	0.001309	⋯	0.000562	0.000634
2	0.001159	∞	0.001712	0.001629	⋯	0.000446	0.000445
3	0.001295	0.001712	∞	0.013333	⋯	0.000395	0.00043
4	0.001309	0.001629	0.013333	∞	⋯	0.000397	0.000431
	⋮	⋮	⋮	⋮	⋱	⋮	⋮
50	0.000562	0.000446	0.000395	0.000397	⋯	∞	0.00463
51	0.000634	0.000445	0.00043	0.000431	⋯	0.00463	∞

PENCARIAN SOLUSI GLOBAL

- ***Langkah 3: Menentukan pelabuhan yang akan dikunjungi selanjutnya***

Fungsi probabilitas

$$p_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{u \in U_k}[\tau_{iu}(t)]^{\alpha}[\eta_{iu}]^{\beta}}$$

Menentukan pelabuhan pertama yang akan dikunjungi dari depot 1 berdasarkan probabilitas berikut.

$$p_{13}^{1}(1) = \frac{[\tau_{13}(1)]^{1}[\eta_{13}]^{1}}{\sum_{u \in U_1}[\tau_{1u}(t)]^{1}[\eta_{1u}]^{1}} = \frac{1 \cdot 0.001295}{0.1023} = 0.0127$$

$$p_{14}^{1}(1) = \frac{[\tau_{14}(1)]^{1}[\eta_{14}]^{1}}{\sum_{u \in U_1}[\tau_{1u}(t)]^{1}[\eta_{1u}]^{1}} = \frac{1 \cdot 0.001309}{0.1023} = 0.0128$$

$$p_{15}^{1}(1) = \frac{[\tau_{15}(1)]^{1}[\eta_{15}]^{1}}{\sum_{u \in U_1}[\tau_{1u}(t)]^{1}[\eta_{1u}]^{1}} = \frac{1 \cdot 0.0018}{0.1023} = 0.0176$$

$$p_{16}^{1}(1) = \frac{[\tau_{16}(1)]^{1}[\eta_{16}]^{1}}{\sum_{u \in U_1}[\tau_{1u}(t)]^{1}[\eta_{1u}]^{1}} = \frac{1 \cdot 0.0015}{0.1023} = 0.0142$$

PENCARIAN SOLUSI GLOBAL

- ***Langkah 4: Menghitung jarak rute yang dihasilkan semut***

Rute yang dihasilkan:

1-5-3-6-8-22-37-45-44-50-41-1-2-7-9-10-13-12-15-19-21-14-16-2-17-18-35-33-29-30-36-24-23-26-17-34-4-20-39-32-31-28-27-38-34-51-42-47-48-49-46-40-11-25-43-51

Total jarak = 30830 km

PENCARIAN SOLUSI GLOBAL

- ***Langkah 5: Menghitung penambahan feromon pada setiap sisi (i, j) yang dilalui semut***

$$\Delta\tau_{ij}^{1} = \frac{Q}{L_1} = \frac{10000}{30830} = 0.3244$$

terdapat satu semut yang digunakan, maka

$$\Delta\tau_{ij} = \sum_{k=1}^{1} \Delta\tau_{ij}^{k} = 0.3244$$

	1	2	3	4	5	6	7	8	⋯	45	46	47	48	49	50	51
1	0	0	0	0	0.324	0	0	0	⋯	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0.324	0	⋯	0	0	0	0	0	0	0
3	0	0	0	0	0	0.324	0	0	⋯	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	⋯	0	0	0	0	0	0	0
5	0	0	0.324	0	0	0	0	0	⋯	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0.324	⋯	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	⋯	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0		⋯	0	0	0	0	0	0	0
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
45	0	0	0	0	0	0	0	0	⋯	0	0	0	0	0	0	0
46	0	0	0	0	0	0	0	0	⋯	0	0	0	0	0	0	0
47	0	0	0	0	0	0	0	0	⋯	0	0	0	0.324	0	0	0
48	0	0	0	0	0	0	0	0	⋯	0	0	0	0	0.324	0	0
49	0	0	0	0	0	0	0	0	⋯	0	0.324	0	0	0	0	0
50	0	0	0	0	0	0	0	0	⋯	0	0	0	0	0	0	0
51	0	0	0	0	0	0	0	0	⋯	0	0	0	0	0	0	0

PENCARIAN SOLUSI GLOBAL

- ***Langkah 6: Update trail***

$$\tau_{ij}(t+n) = (1-0.5)\tau_{ij} \times +\Delta\tau_{ij} =$$

0.5	0.5	0.5	0.5	0.824	0.5	0.5	5.5	⋯	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.824	0.5	⋯	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	5.5	0.5	0.824	0.5	0.5	⋯	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	⋯	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	5.824	0.5	0.5	0.5	0.5	0.5	⋯	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.824	⋯	0.5	0.5	0.5	0.5	0.5	0.5
5.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	⋯	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	⋯	0.5	0.5	0.5	0.5	0.5	0.5
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	⋯	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	⋯	0.5	0.5	5.824	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	⋯	0.5	0.5	0.5	5.824	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	⋯	5.824	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	⋯	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	⋯	0.5	0.5	0.5	0.5	0.5	0.5

UJI VALIDITAS

- Empiris : membandingkan perhitungan manual dengan hasil program.
- Sub-proses.

Hal-hal yang dibandingkan:

- Populasi kunang-kunang untuk iterasi selanjutnya
- Hasil normalisasi solusi
- Inisialisasi feromon awal
- Probabilitas empat pelabuhan pertama yang dikunjungi dari depot-1
- Update trail

UJI VALIDITAS

Kunang-kunang-1 dengan total jarak 52219

1	38	35	21	29	10	16	8	3	28	9	1	2	11
6	48	43	22	25	36	30	4	2	17	13	31	23	32
15	12	20	44	5	17	34	41	49	33	47	46	42	24
26	50	27	34	51	40	37	7	18	14	39	45	19	51

Kunang-kunang-2 dengan total jarak 54628

1	38	35	21	29	49	41	5	44	20	1	2	12	15
32	23	31	13	4	30	36	2	17	25	22	43	48	6
11	9	28	3	8	17	34	16	10	33	47	46	42	24
26	50	34	51	27	40	37	7	18	14	39	45	19	51

```
of =

    52219    56165    54628    58762

k =

Columns 1 through 14

     1    38    35    21    29    10    16     8     3    28     9     1     2    11
     1    38    35    21    29    49    41     5    44    20     1     2    12    15

Columns 15 through 28

     6    48    43    22    25    36    30     4     2    17    13    31    23    32
    32    23    31    13     4    30    36     2    17    25    22    43    48     6

Columns 29 through 42

    15    12    20    44     5    17    34    41    49    33    47    46    42    24
    11     9    28     3     8    17    34    16    10    33    47    46    42    24

Columns 43 through 56

    26    50    27    34    51    40    37     7    18    14    39    45    19    51
    26    50    34    51    27    40    37     7    18    14    39    45    19    51
```

UJI VALIDITAS

Hasil normalisasi solusi

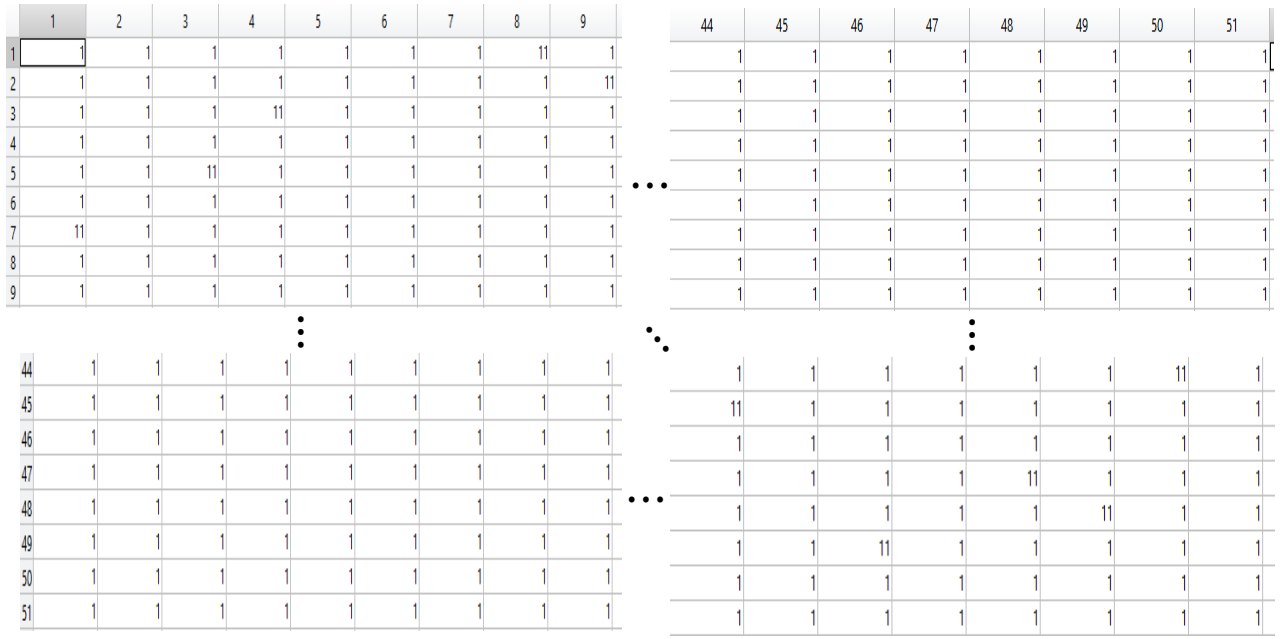
1	8	22	37	45	44	50	41	19	21	7	1	2	9
10	13	12	15	6	14	16	18	2	17	35	33	29	30
36	24	23	26	17	34	5	3	4	20	39	32	31	28
27	38	34	51	42	47	48	49	46	40	11	25	43	51

```
norm_solusi =

  Columns 1 through 14

     1     8    22    37    45    44    50    41    19    21     7     1     2     9

  Columns 15 through 28

    10    13    12    15     6    14    16    18     2    17    35    33    29    30

  Columns 29 through 42

    36    24    23    26    17    34     5     3     4    20    39    32    31    28

  Columns 43 through 56

    27    38    34    51    42    47    48    49    46    40    11    25    43    51
```

UJI VALIDITAS

Inisialisasi feromon

	1	2	3	4	5	6	7	8	9		44	45	46	47	48	49	50	51
1	1	1	1	1	1	1	1	11	1	⋯	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	11	⋯	1	1	1	1	1	1	1	1
3	1	1	1	11	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
5	1	1	11	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
7	11	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
9	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
44	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	11	1
45	1	1	1	1	1	1	1	1	1	⋯	11	1	1	1	1	1	1	1
46	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
47	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	11	1	1	1
48	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	11	1	1
49	1	1	1	1	1	1	1	1	1	⋯	1	1	11	1	1	1	1	1
50	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1
51	1	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	1



UJI VALIDITAS

Perhitungan manual probabilitas untuk menentukan pelabuhan yang selanjutnya disinggahi.

- $p_{13}^{1}(1) = 0.0127$
- $p_{14}^{1}(1) = 0.0128$
- $p_{15}^{1}(1) = 0.0176$
- $p_{16}^{1}(1) = 0.0142$

```
prob =

  Columns 1 through 8

    0.0127    0.0128    0.0176    0.0142    0.0183    0.4843    0.0145    0.0153

  Columns 9 through 16

    0.0095    0.0111    0.0122    0.0134    0.0131    0.0416    0.0089    0.0097

  Columns 17 through 24

    0.0088    0.0233    0.0269    0.0210    0.0194    0.0151    0.0128    0.0078

  Columns 25 through 32

    0.0088    0.0068    0.0085    0.0101    0.0080    0.0086    0.0125    0.0102

  Columns 33 through 40

    0.0073    0.0067    0.0079    0.0059    0.0066    0.0052    0.0045    0.0056

  Columns 41 through 46

    0.0052    0.0050    0.0052    0.0062    0.0055    0.0055
```

UJI VALIDITAS

Update trail

0.5	0.5	0.5	0.5	0.824	0.5	0.5	5.5	⋯	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.824	0.5	⋯	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	5.5	0.5	0.824	0.5	0.5	⋯	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	⋯	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	5.824	0.5	0.5	0.5	0.5	0.5	⋯	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.824	⋯	0.5	0.5	0.5	0.5	0.5	0.5
5.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	⋯	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	⋯	0.5	0.5	0.5	0.5	0.5	0.5
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	⋯	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	⋯	0.5	0.5	5.824	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	⋯	0.5	0.5	0.5	5.824	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	⋯	5.824	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	⋯	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	⋯	0.5	0.5	0.5	0.5	0.5	0.5

	1	2	3	4	5	6	7	8	⋯	46	47	48	49	50	51
1	0.5000	0.5000	0.5000	0.5000	0.8244	0.5000	0.5000	5.5000	⋯	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
2	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.8244	0.5000	⋯	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
3	0.5000	0.5000	0.5000	5.5000	0.5000	0.8244	0.5000	0.5000	⋯	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
4	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	⋯	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
5	0.5000	0.5000	5.8244	0.5000	0.5000	0.5000	0.5000	0.5000	⋯	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
6	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.8244	⋯	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
7	5.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	⋯	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
8	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	⋯	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
⋮									⋱					⋮	
46	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	⋯	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
47	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	⋯	0.5000	0.5000	5.8244	0.5000	0.5000	0.5000
48	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	⋯	0.5000	0.5000	0.5000	5.8244	0.5000	0.5000
49	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	⋯	5.8244	0.5000	0.5000	0.5000	0.5000	0.5000
50	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	⋯	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
51	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	⋯	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000

UJI COBA DAN ANALISIS

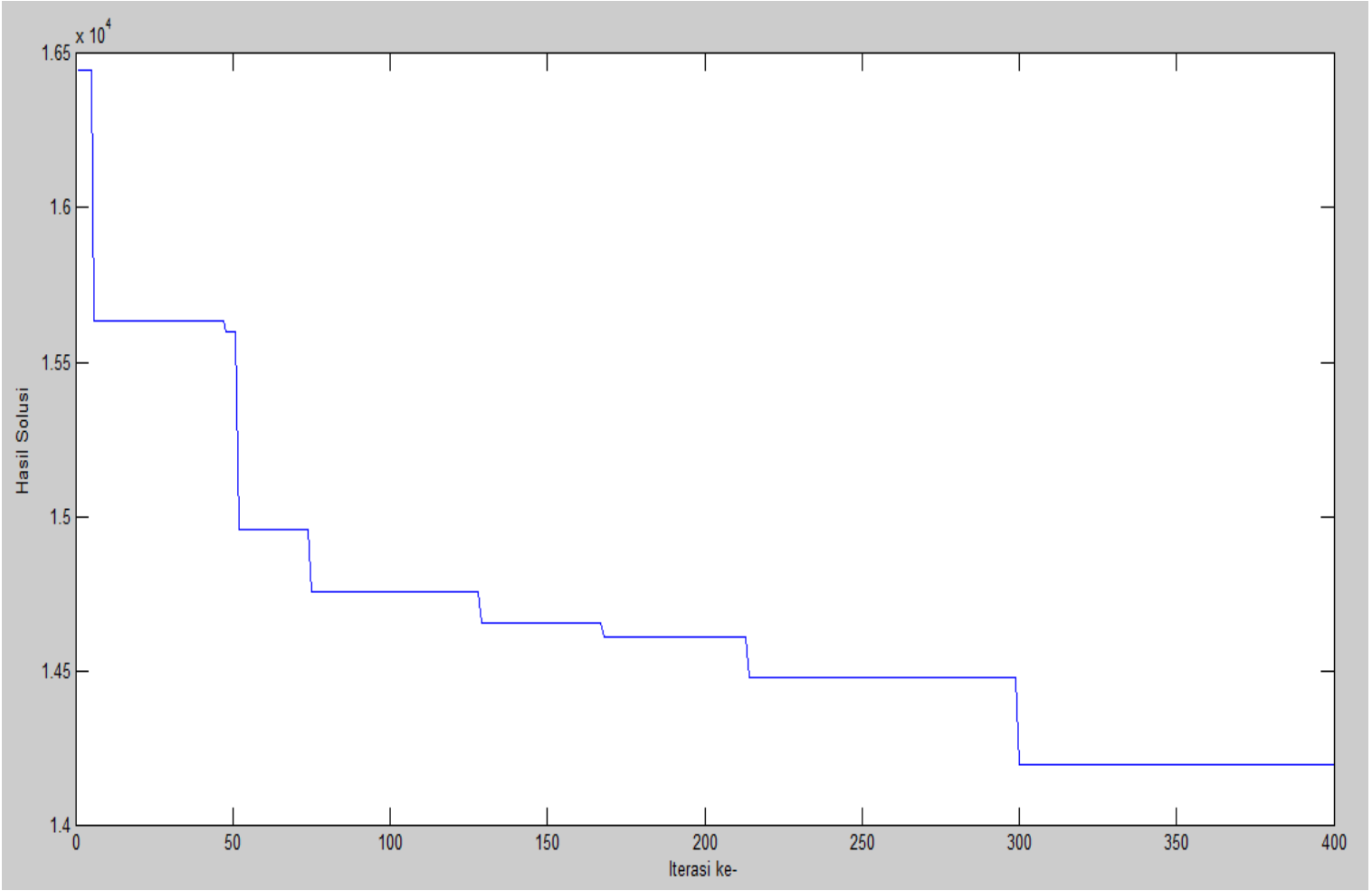
- Penentuan parameter ACO
- Penentuan parameter hybrid FA-ACO
- Perbandingan hasil

PENENTUAN BANYAK ITERASI (ACO)

$\alpha = 1, \beta = 5, \rho = 0.5, Q = 100$

dan semut sebanyak n (Dorigo, 1996)

Iterasi minimal yang diperlukan ACO untuk mencapai konvergensi adalah 300



PENENTUAN PARAMETER α DAN β (ACO)

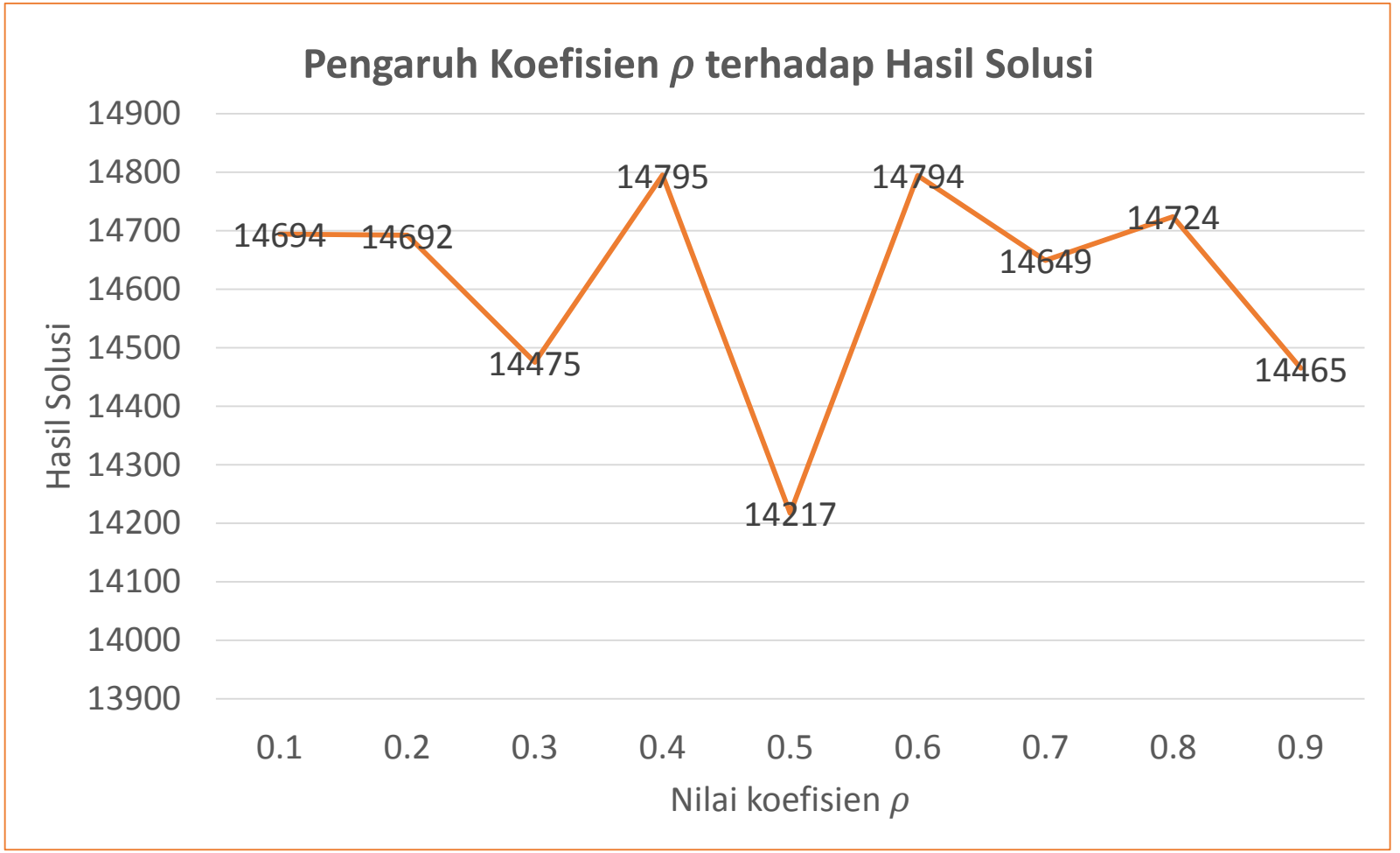
Nilai yang akan diujicobakan adalah $\alpha \in \{0.5,1,2\}$ dan $\beta \in \{1,2,5\}$ (Dorigo, 1996). Pada uji coba ini, pengaturan *default* yang digunakan antara lain $\rho = 0.5, Q = 100$, semut sebanyak 51 dan banyak iterasi adalah 300.

Hasil terbaik diperoleh jika $\alpha = 1$ dan $\beta = 5$

α	β	Hasil	Waktu
0.5	1.0	21038	81.4154
	2.0	17169	80.0412
	5.0	14659	83.1577
1.0	1.0	19088	80.0881
	2.0	16072	79.0493
	5.0	14231	81.7190
2.0	1.0	18872	79.8144
	2.0	16532	77.6645
	5.0	15361	81.1222

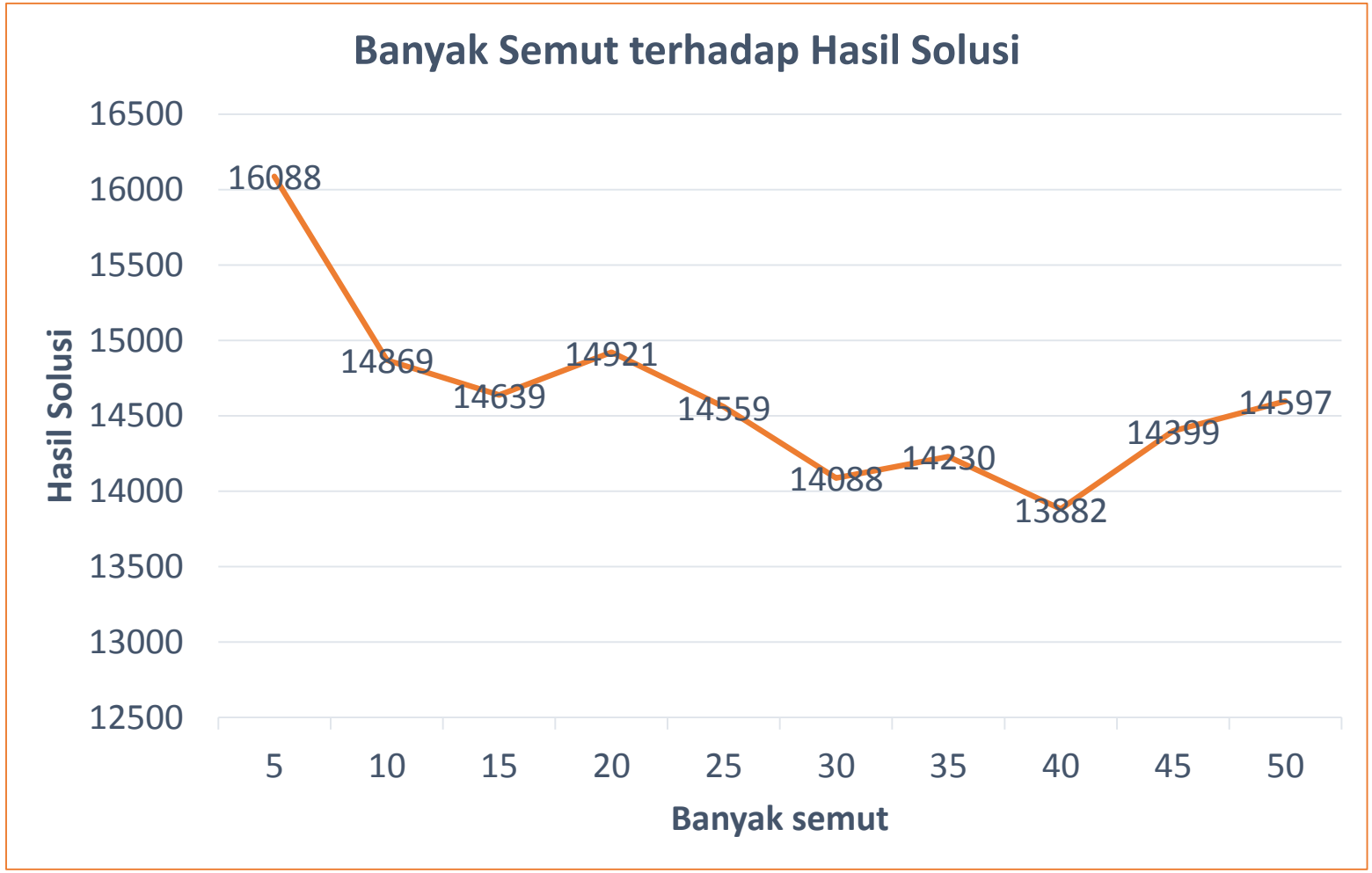
PENENTUAN PARAMETER ρ(ACO)

ρ	Hasil	Waktu
0.1	14694	82.37993
0.2	14692	82.47564
0.3	14475	82.2798
0.4	14795	82.13375
0.5	14217	82.33667
0.6	14794	82.56143
0.7	14649	82.74688
0.8	14724	82.22003
0.9	14465	82.59745



PENENTUAN BANYAK SEMUT (ACO)

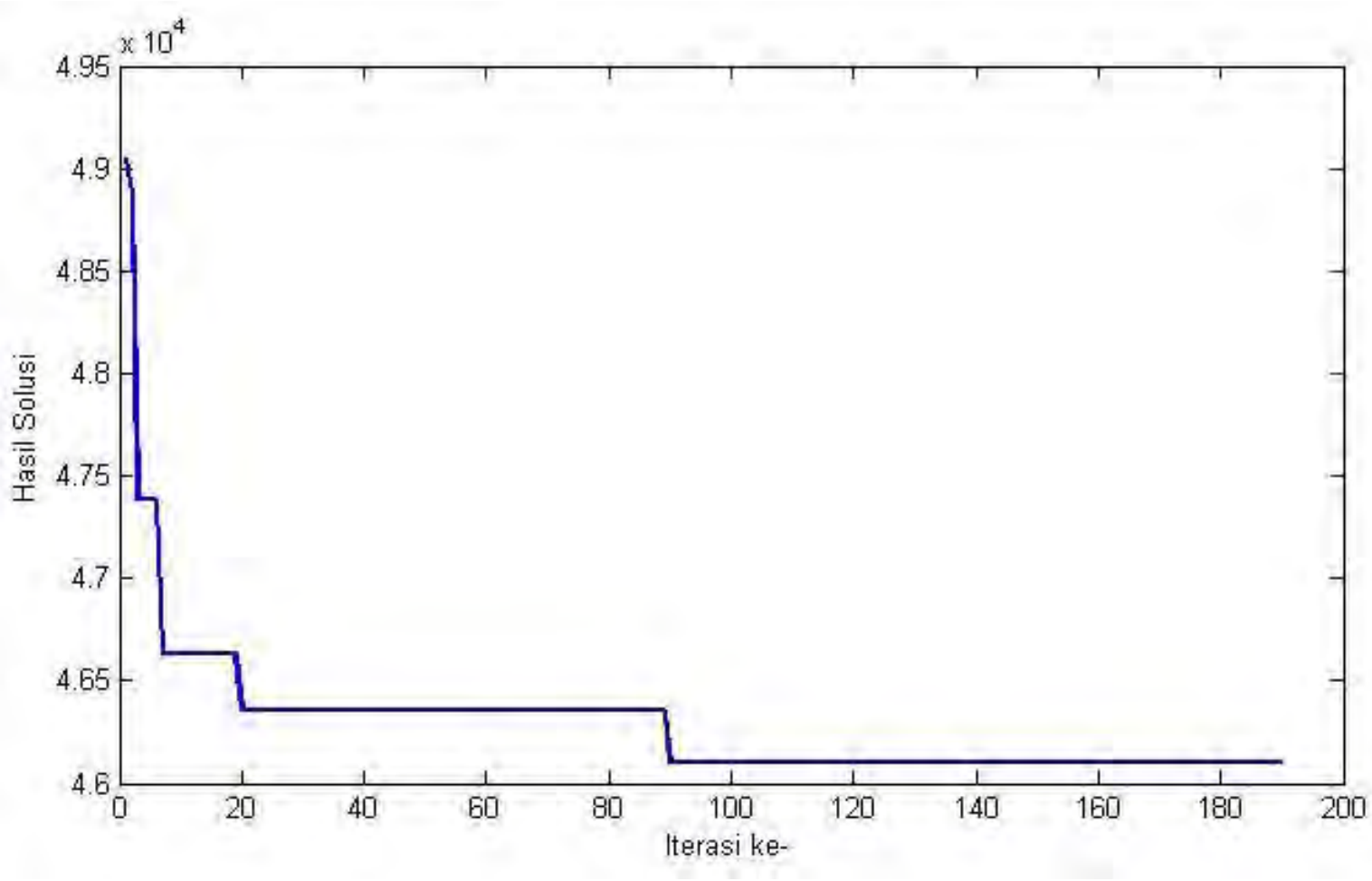
Banyak semut	Hasil	Waktu
5	16088	8.169784
10	14869	16.40132
15	14639	24.41994
20	14921	32.88892
25	14559	41.32483
30	14088	50.2965
35	14230	57.64817
40	13882	62.98049
45	14399	73.93611
50	14597	82.45433



PENENTUAN BANYAK ITERASI FA

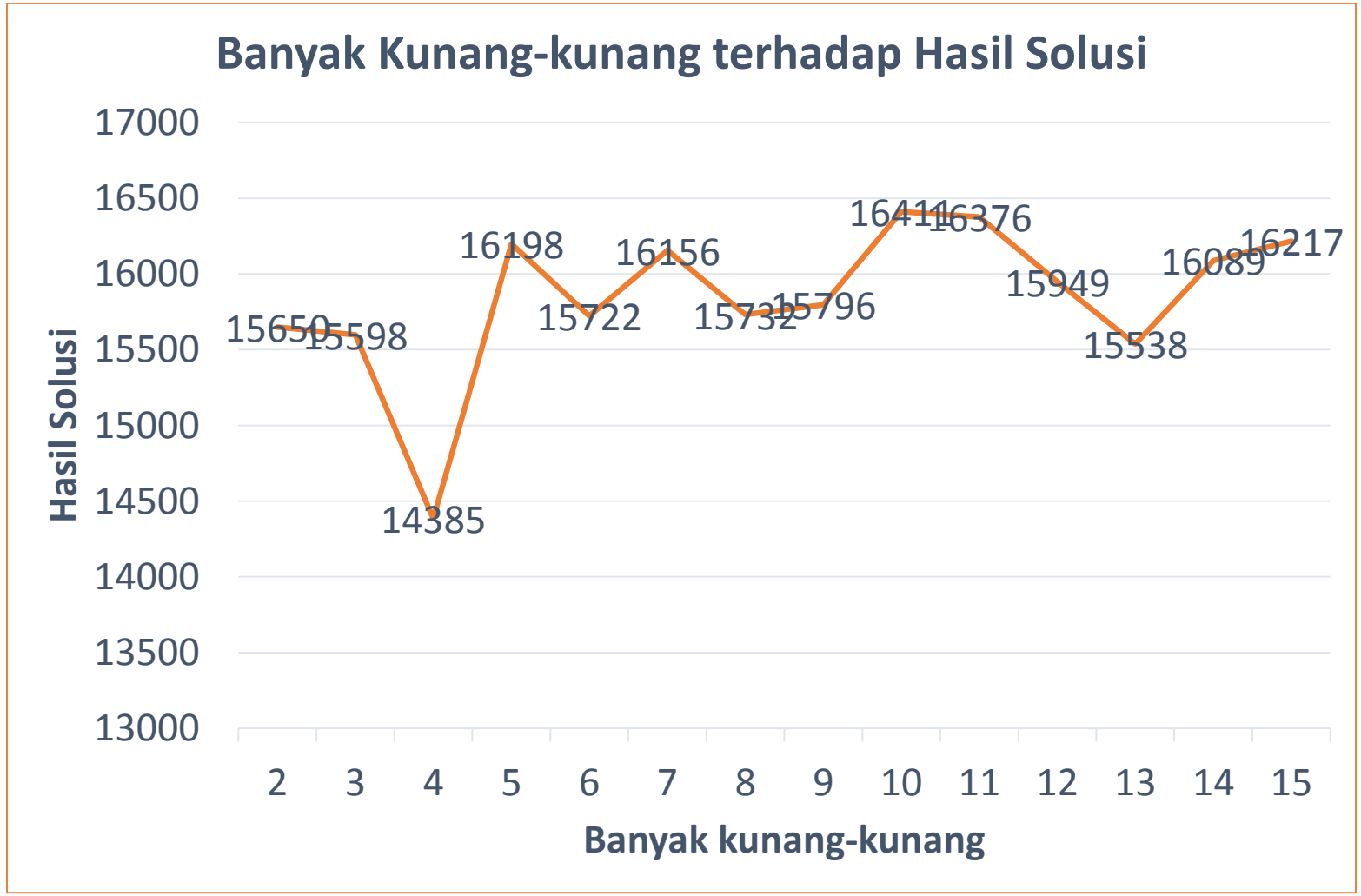
Parameter yang digunakan antara lain $\gamma = 0.07$ dengan 15 kunang-kunang dan 11 pergerakan untuk tiap kunang-kunang.

Konvergensi dicapai pada saat iterasi ke-100



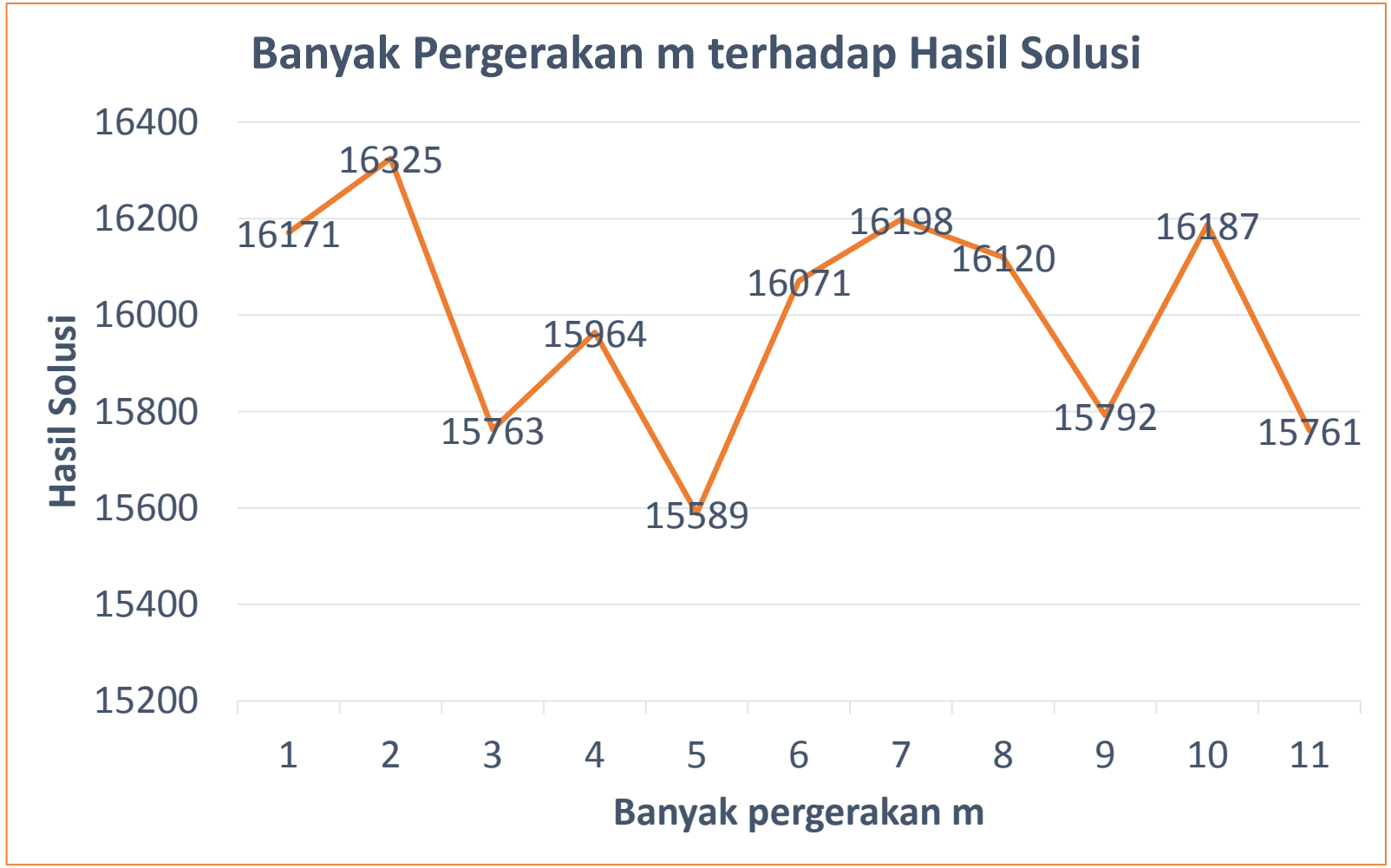
PENENTUAN BANYAK KUNANG-KUNANG

Kunang-kunang	Hasil	Waktu
2	15650	94.96176
3	15598	101.0158
4	14385	107.2189
5	16198	112.8496
6	15722	119.6184
7	16156	125.7431
8	15732	132.3029
9	15796	137.7738
10	16411	144.1912
11	16376	150.0479
12	15949	156.675
13	15538	162.2745
14	16089	167.6822
15	16217	173.826



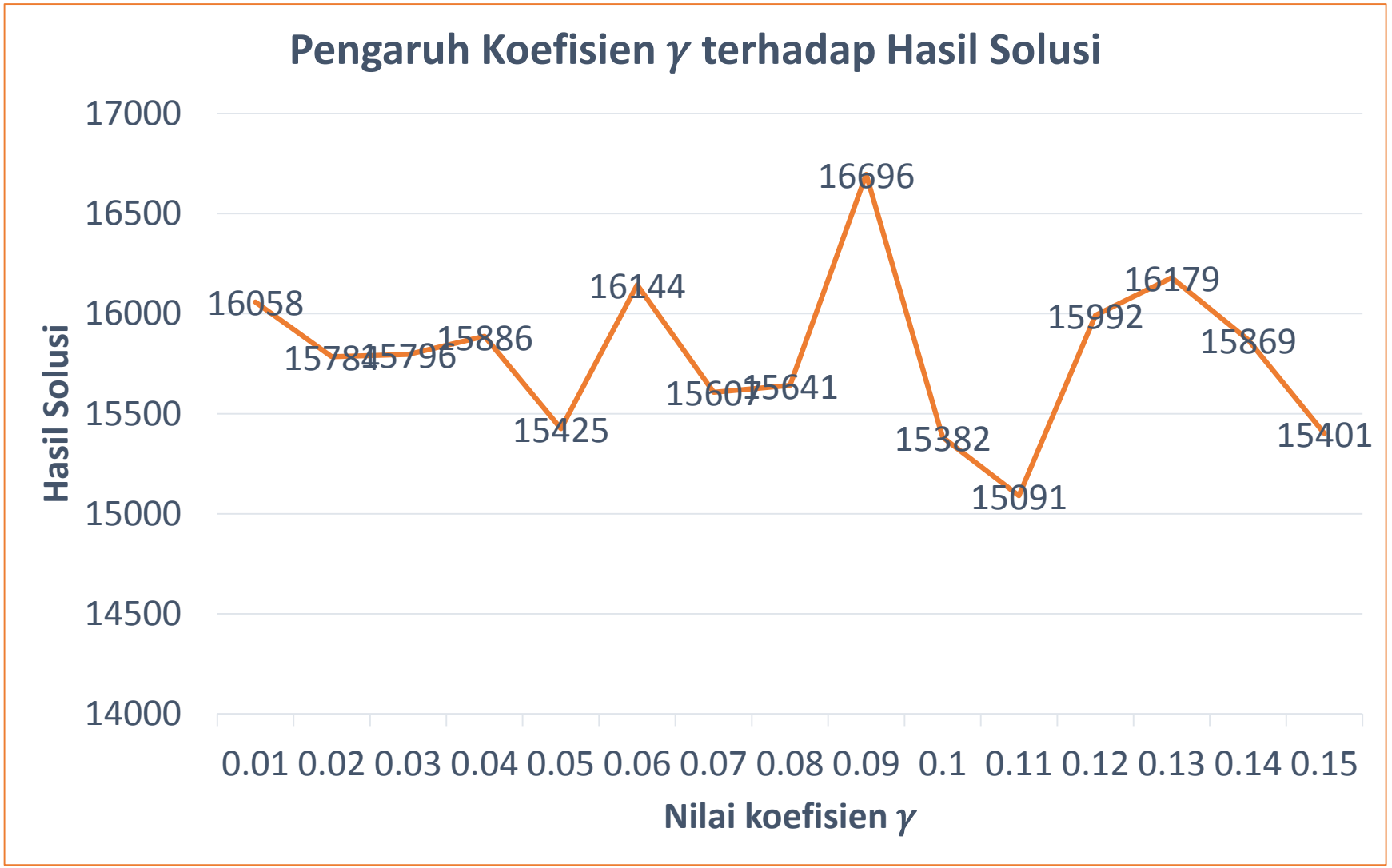
PENENTUAN PERGERAKAN

Banyak pergerakan	Hasil	Waktu
1	16171	87.38962
2	16325	89.61266
3	15763	91.3517
4	15964	93.58011
5	15589	95.88416
6	16071	97.91237
7	16198	100.1068
8	16120	101.2708
9	15792	103.419
10	16187	105.8723
11	15761	107.7854



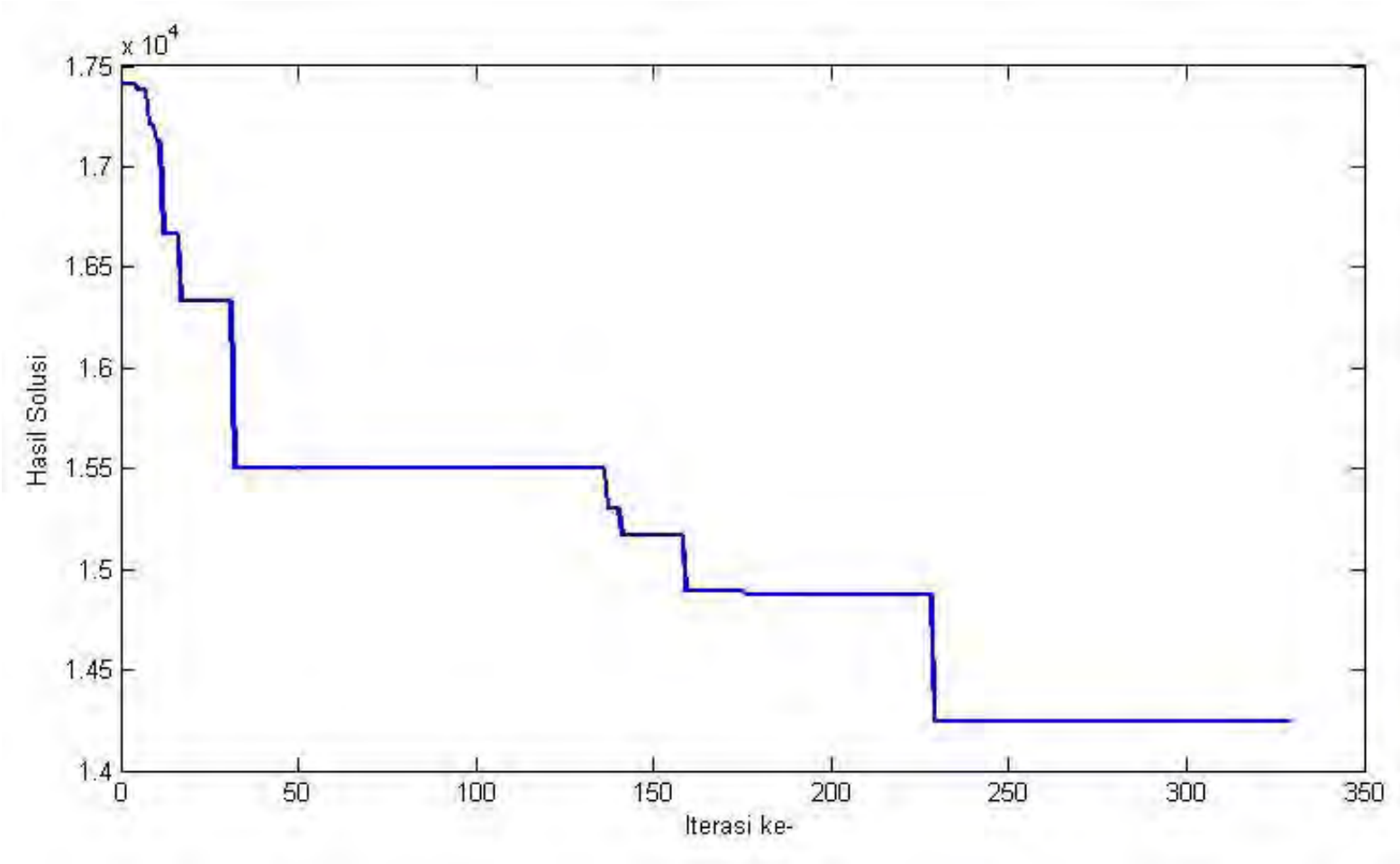
PENENTUAN PARAMETER γ

γ	Hasil	Waktu
0.01	16058	96.07631
0.02	15784	96.34295
0.03	15796	95.94018
0.04	15886	96.08202
0.05	15425	95.62379
0.06	16144	96.31628
0.07	15607	96.66601
0.08	15641	96.53021
0.09	16696	95.3499
0.10	15382	96.1339
0.11	15091	97.71056
0.12	15992	96.88888
0.13	16179	96.78503
0.14	15869	96.59213
0.15	15401	96.76168



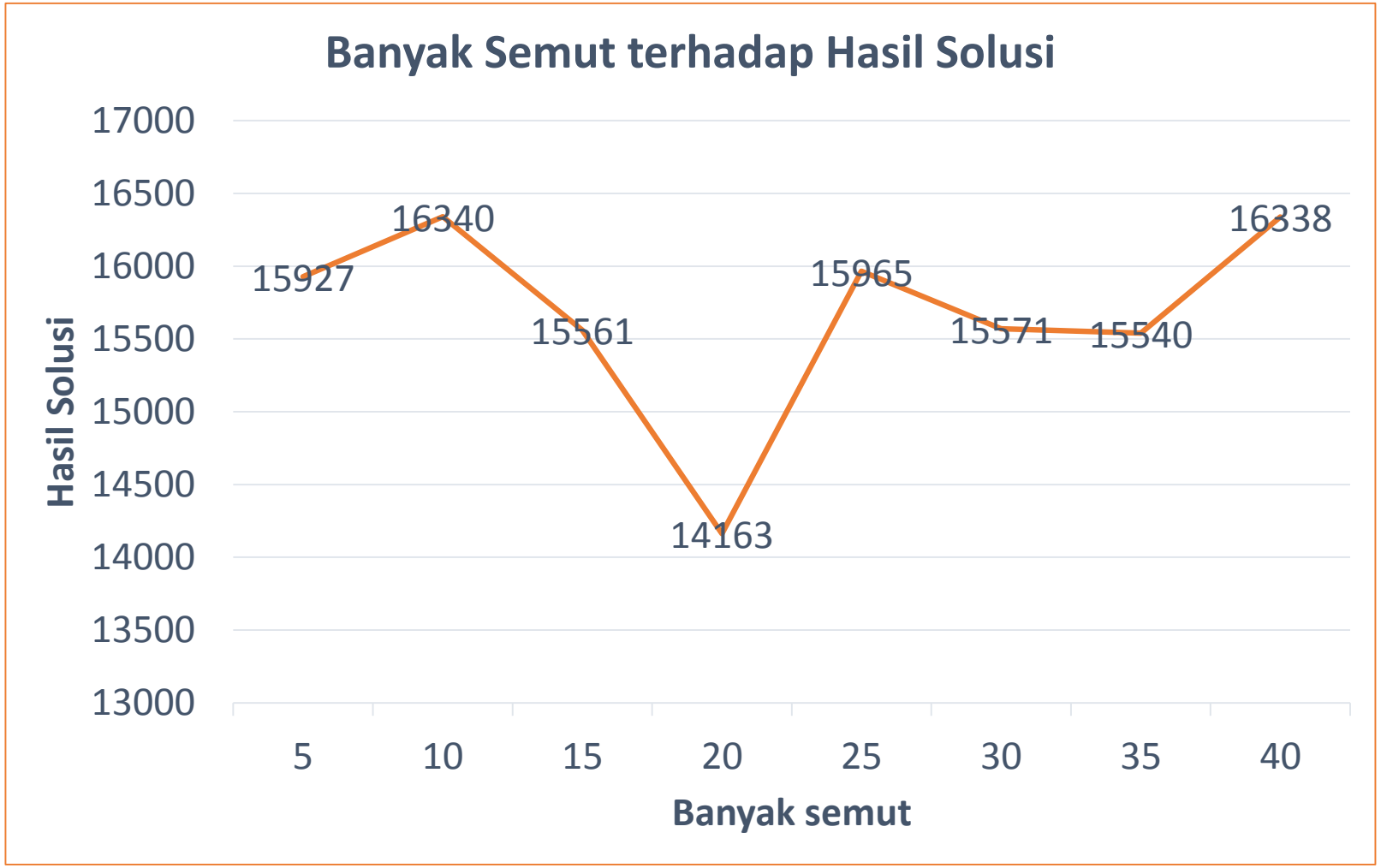
PENENTUAN BANYAK ITERASI ACO

Iterasi minimal yang diperlukan pencarian solusi global ACO untuk mencapai konvergensi adaah 229.



PENENTUAN BANYAK SEMUT

Banyak semut	Hasil	Waktu
5	15927	20.3944
10	16340	28.21828
15	15561	36.42925
20	14163	44.30809
25	15965	52.59493
30	15571	60.1798
35	15540	68.16099
40	16338	76.49842



PERBANDINGAN METODE

ACO :

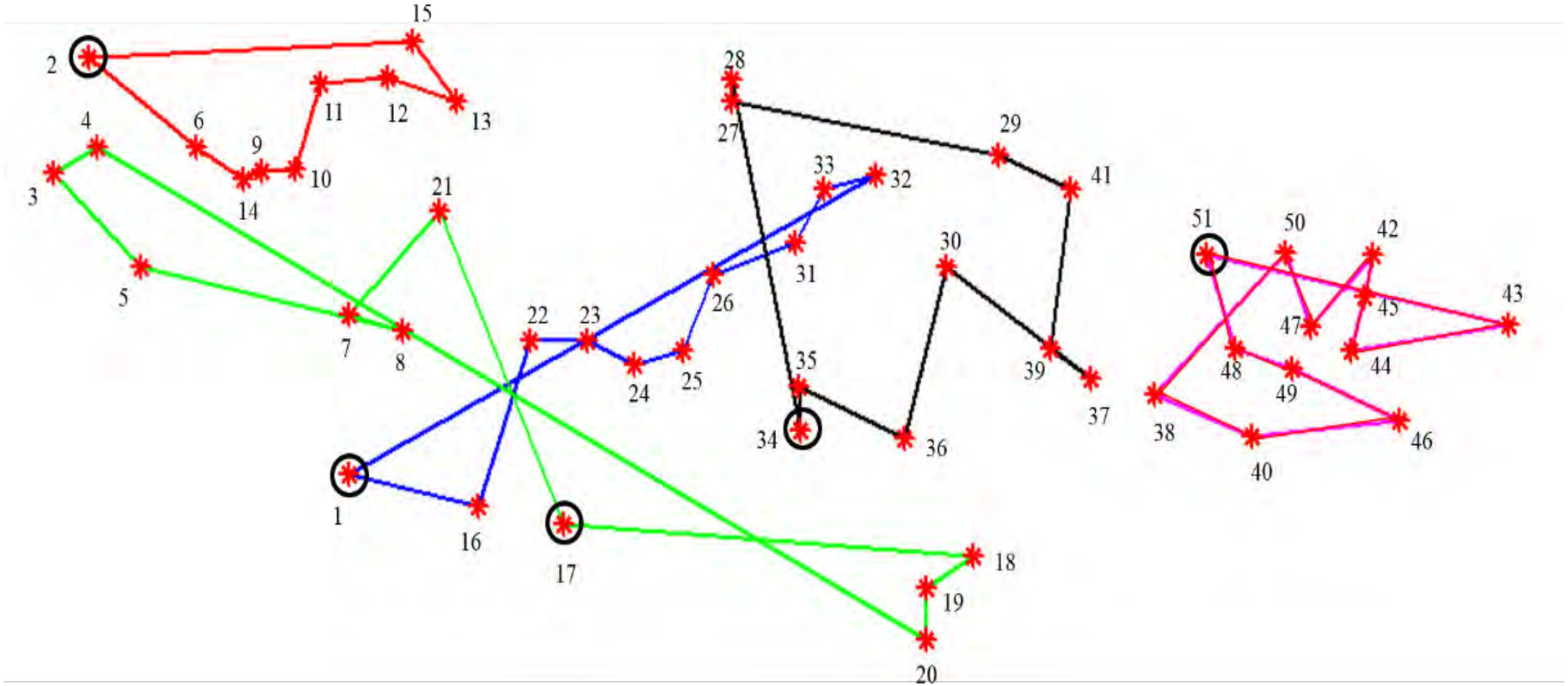
$\alpha = 1, \beta = 5, \rho = 0.5, Q = 100$, 40 semut, dan 300 iterasi

Hybrid FA-ACO :

Parameter $\gamma = 0.01$, kunang-kunang sebanyak 4 dan 5 pergerakan untuk tiap kunang-kunang,100 iterasi FA, $\alpha = 1, \beta = 5, \rho = 0.5, Q = 100$, 20 semut, dan 229 iterasi ACO

Metode	Solusi terbaik yang didapat	Rata-rata solusi	Rata-rata waktu komputasi
Hybrid FA-ACO	13828	14400.28	44.42685
ACO	13882	14463.24	60.77814

HASIL RUTE OPTIMUM



HASIL RUTE OPTIMUM

- Kapal keberangkatan dari Tanjung Priok memiliki rute Tanjung Priok, Tanjung Mas, Kumai, Sampit, Banjarmasin, Balikpapan, Donggala, Toli-toli, Lokodidi, dan Tanjung Priok dengan jarak tempuh 2915 km.
- Kapal keberangkatan dari Belawan memiliki rute Belawan, Dumai, Tanjung Balai Karimun, Batam, Kijang, Letung, Midai, Serasan, Natuna, dan Belawan dengan jarak tempuh 1758 km.
- Kapal keberangkatan dari Tanjung Perak memiliki rute Tanjung Perak, Pontianak, Belinyu, Tanjung Pandan, Teluk Bayur, Gunung Sitoli, Sibolga, Ternate, Larantuka, Kalabahi, dan Tanjung Perak dengan jarak tempuh 2957 km.
- Kapal keberangkatan dari Makassar memiliki rute Makassar, Pare-Pare, Bau-Bau, Banggai, Ambon, Namlea, Ternate, Bitung, Tarakan, Nunukan, dan Makassar dengan jarak tempuh 2679 km.
- Kapal keberangkatan dari Sorong memiliki rute Sorong, Fakfak, Kaimana, Timika, Tual, Banda, Manokwari, Wasior, Nabire, Serai, Biak, Jayapura, dan Sorong dengan jarak tempuh 3519 km.

PERBANDINGAN KONVERGENSI

Hybrid FA-ACO

Hasil terbaik : 13828 km

Waktu komputasi : 33.28346 detik

Waktu konvergensi :

16.86102 detik

ACO

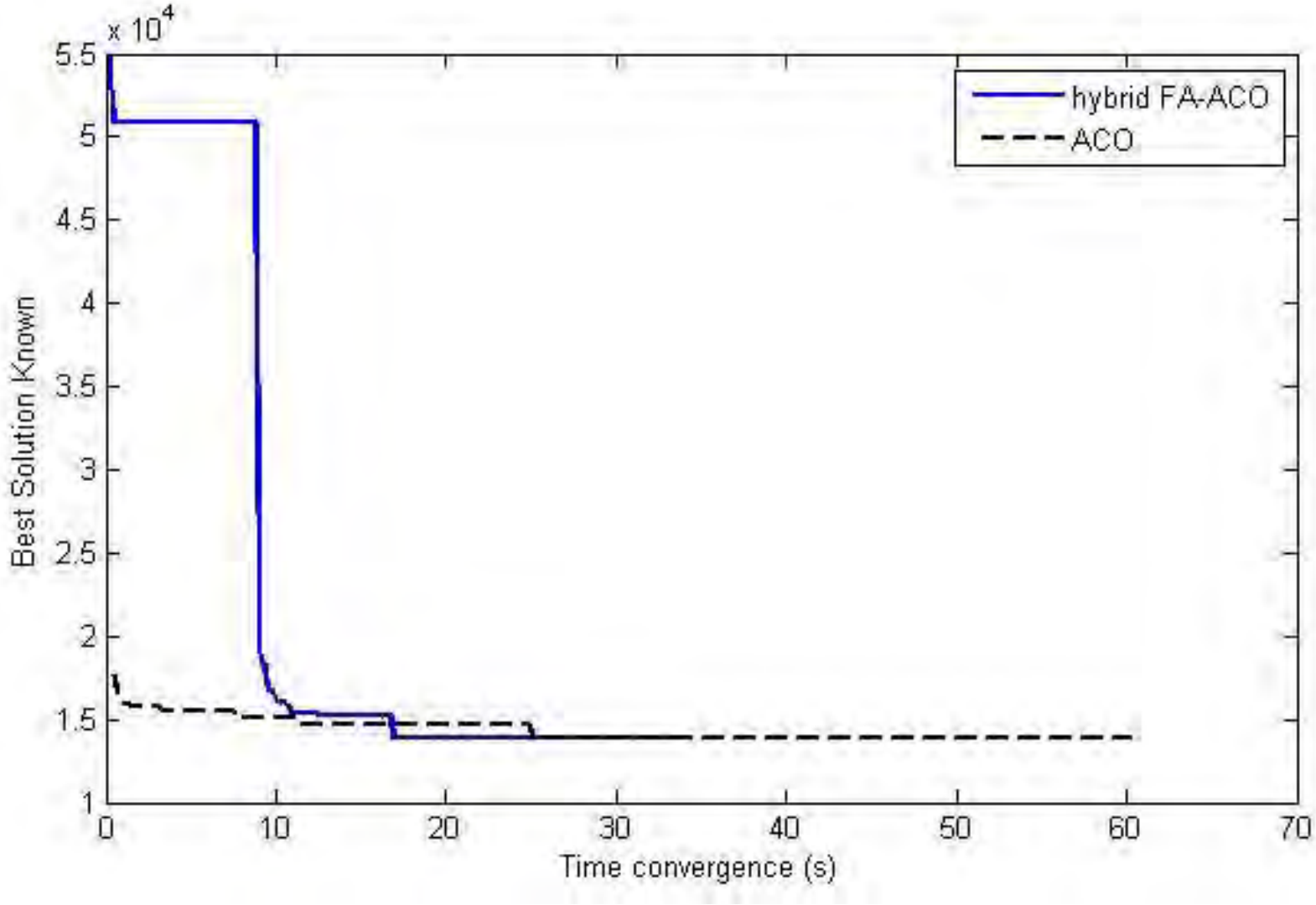
Hasil terbaik : 13828 km

Waktu komputasi : 60.77814

detik

Waktu konvergensi :

25.07218 detik



KESIMPULAN

ACO :

$\alpha = 1, \beta = 5, \rho = 0.5, Q = 100$, 40 semut, dan 300 iterasi

Hybrid FA-ACO :

Parameter $\gamma = 0.01$, kunang-kunang sebanyak 4 dan 5 pergerakan untuk tiap kunang-kunang,100 iterasi FA, $\alpha = 1, \beta = 5, \rho = 0.5, Q = 100$, 20 semut, dan 229 iterasi ACO

ACO

Hasil terbaik : 13882 km

Rata-rata solusi: 14463,24 km

Rata-rata waktu komputasi : 60,77814 detik

Hybrid FA-ACO

Hasil terbaik : 13828 km

Rata-rata solusi: 14400,28 km

Rata-rata waktu komputasi : 44,42685 detik

ACO

Hasil terbaik : 13828 km

Waktu komputasi : 60,77814 detik

Waktu konvergensi : 25,07218 detik

Hybrid FA-ACO

Hasil terbaik : 13828 km

Waktu komputasi : 33,28346 detik

Waktu konvergensi : 16,86102 detik

Metode *hybrid* FA-ACO lebih baik dalam mencapai solusi optimum dengan rata-rata waktu komputasi 26,90% dan waktu konvergensi 32,75% lebih cepat dibandingkan metode ACO