PENYELESAIAN MULTI-DEPOT MULTIPLE TRAVELING SALESMAN PROBLEM MENGGUNAKAN K-MEANS DAN ANT COLONY OPTIMIZATION

Olief Ilmandira Ratu Farisi[1)] dan Gulpi Qorik Oktagalu Pratamasunu[2)]
[1, 2)]Jurusan Teknik Informatika, Sekolah Tinggi Teknologi Nurul Jadid
PP. Nurul Jadid Paiton Probolinggo 67291
e-mail: olief.ilmandira@gmail.com[1)], gulpi.qorik@gmail.com[2)]

ABSTRAK

Multi-Depot Multiple Traveling Salesman Problem (MmTSP) merupakan masalah pencarian rute terpendek oleh beberapa salesman yang berangkat dari kota yang berbeda-beda, disebut depot, dan kembali ke depotnya masing-masing dengan setiap kota harus dikunjungi tepat satu kali. ACO merupakan algoritma yang didesain untuk menyelesaikan TSP. Untuk menyelesaikan MmTSP, pada penelitian ini diusulkan metode K-Means ACO. K-Means digunakan untuk mencari pembagian kota yang optimal. Pembagian ini dilakukan sesuai dengan banyak depot pada permasalahan. Hasil setiap cluster ini menjadi kota-kota yang akan dikunjungi oleh setiap salesman. Setiap cluster hasil dari K-Means dicari rute terpendeknya menggunakan ACO. Gabungan hasil rute terpendek dari setiap cluster tersebut menjadi penyelesaian MmTSP. Hasil penelitian menunjukkan bahwa metode K-Means ACO dapat mencari rute yang mendekati optimal dengan waktu yang singkat.

***Kata Kunci**: Ant Colony Optimization, K-Means, Multi-Depot Multiple Traveling Salesman Problem*

ABSTRACT

Multi-Depot Multiple Traveling Salesman Problem (MmTSP) is 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. ACO is an algorithm designed for solving TSP. In this paper, we proposed a method, K-Means ACO, for solving MmTSP. K-Means is used to find the optimal cluster of cities. The clustering based on the number of depot in the problem. The result of each cluster will be the visited cities for each salesman. We used ACO to find the shortest route for each cluster of K-Means. All of the shortest routes from each cluster is the solution of the MmTSP. The experimental result showed that K-Means ACO can find the solution much better with shorter computation time.

***Keywords**: Ant Colony Optimization, K-Means, Multi-Depot Multiple Traveling Salesman Problem*

I. PENDAHULUAN

TRAVELING Salesman Problem merupakan salah satu masalah optimasi kombinatorial. Prinsip dari TSP adalah bagaimana seorang *salesman* dapat mengunjungi seluruh kota pada suatu daerah tepat satu kali dan harus kembali ke kota awal keberangkatan dengan jarak seminimal mungkin. Beberapa permasalahan yang dapat diselesaikan dengan TSP adalah permasalahan perencanaan kunjungan wisata, logistik, *integrated circuit* (IC), 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.

Ant Colony Optimization (ACO) merupakan suatu algoritma untuk menyelesaikan TSP yang diperkenalkan pertama kali oleh Marco Dorigo, dkk [1]. Metode ini mencoba menirukan tingkah laku koloni semut dalam mencari rute terpendek dari sarang menuju ke tempat makanan. ACO memodelkan setiap semut berjalan mengunjungi setiap kota dan meninggalkan zat feromon di rute yang dilewatinya.

Semut lain akan mengikuti rute dengan feromon yang paling banyak. Dengan cara ini rute terpendek dari permasalahan TSP dapat ditemukan dengan metode ACO.

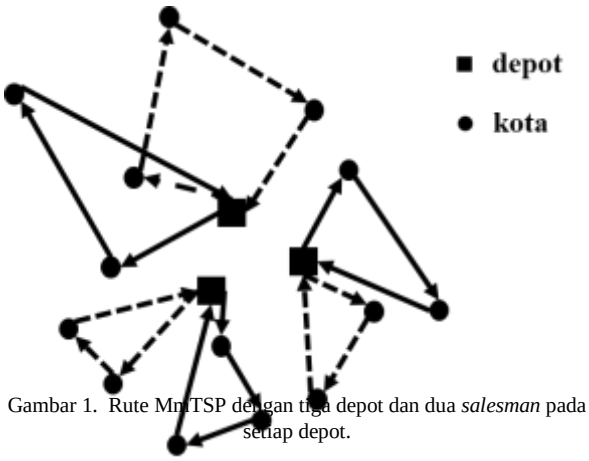
ACO telah banyak digunakan untuk menyelesaikan masalah-masalah TSP dengan hasil yang mendekati optimal. Tetapi metode ACO tidak bisa digunakan untuk menyelesaikan permasalahan TSP yang lebih kompleks seperti mTSP dan MmTSP. ACO harus didesain secara spesifik untuk setiap masalah [2]. Untuk mencari desain ACO yang efisien cukup sulit. Sehingga, tidak jarang ACO dikombinasikan dengan algoritma meta-heuristik lainnya. Selain itu, banyaknya parameter yang digunakan di ACO, membuat lebih sulit lagi untuk mencari parameter yang cocok dalam setiap permasalahan.

Jinjie dan Dingwei [3] dalam penelitiannya memodifikasi ACO untuk menyelesaikan masalah mTSP. Pada penelitian tersebut representasi rute mTSP dimodifikasi menjadi rute TSP karena setiap *salesman* berangkat dari kota yang sama. Sayangnya, metode ini tidak bisa diterapkan untuk MmTSP karena setiap *salesman* berangkat dari depot yang berbeda-beda. Oleh karena itu, dibutuhkan suatu metode untuk menemukan rute yang optimal pada permasalahan MmTSP.

Pada paper ini diusulkan penggunaan metode K-Means ACO untuk menemukan rute optimal dari suatu permasalahan MmTSP. Pada MmTSP, rute yang diambil setiap *salesman* tidak berhubungan secara langsung dengan rute *salesman* lain karena setiap *salesman* berangkat dari kota yang berbeda. Sehingga, rute setiap *salesman* dapat ditentukan menggunakan ACO. K-Means digunakan untuk menemukan pembagian *cluster* yang optimal yang harus ditempuh masing-masing *salesman* berdasarkan jarak antar kota. K-Means memastikan bahwa setiap *salesman* benar-benar mengunjungi kota-kota terdekat dari depot masing-masing. Pembagian *cluster* ini akan mengurangi kemungkinan solusi yang harus dilalui metode ACO sehingga waktu komputasi yang dibutuhkan menjadi lebih cepat karena pencarian rute terpendek setiap depot dilakukan secara terpisah. Dengan metode ini diharapkan rute optimal untuk *salesman* dapat ditemukan dengan meminimalkan jumlah jarak yang harus ditempuh seluruh *salesman*.

II. MULTI-DEPOT MULTIPLE TRAVELING *SALESMAN* PROBLEM

Traveling Salesman Problem (TSP) merupakan permasalahan optimasi kombinatorial dimana terdapat seorang *salesman* yang ingin mengunjungi seluruh kota pada suatu daerah tepat satu kali dan harus kembali ke kota awal keberangkatan. Penyelesaian dari TSP adalah rute dengan bobot minimum. Jika terdapat lebih dari satu orang *salesman*, permasalahan ini disebut dengan *Multiple Traveling Salesman Problem* (mTSP). Pada mTSP, beberapa *salesman* ditempatkan pada satu kota keberangkatan yang sama 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. Permasalahan 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. Satu depot bisa terdapat lebih dari satu orang *salesman*. Gambar. 1. merepresentasikan rute MmTSP dengan tiga depot dan dua *salesman* pada masing-masing depot.



Gambar 1. Rute MmTSP dengan tiga depot dan dua *salesman* pada setiap depot.

MmTSP dapat dirumuskan sebagai berikut [4]. Diberikan himpunan tak kosong N yang merupakan himpunan kota dan matriks jarak $c_{ij}, (i, j \in N)$ yang merupakan jarak dari kota-i ke kota-j. Himpunan N dipartisi menjadi $N = D \cup N'$ dimana himpunan depot D adalah d kota pertama dari N dan $N' = \{d+1, d+2, ..., n\}$ adalah himpunan kota kecuali kota yang menjadi depot. Dianggap terdapat m_i *salesman* ditempatkan di depot i pada awalnya. Sehingga, banyak *salesman* untuk semua depot adalah m.

Didefinisikan variabel biner x_{ijk} sama dengan 1 jika *salesman* berangkat dari kota ke-k, melewati (i, j) dan x_{ijk} sama dengan 0, jika tidak melewati (i, j). Untuk sebarang *salesman*, u_i adalah banyaknya kota yang dikunjungi oleh *salesman* dari awal sampai kota-i. Diberikan pula konstanta L dan U yang merupakan banyak minimum dan maksimum kota yang boleh dikunjungi oleh seorang *salesman*, secara berturut-turut; dengan demikian, $1 \le u_i \le U, \forall i > 2$. Jika $x_{ikk} = 1$, maka $L \le u_i \le U$ harus memenuhi

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

dengan kendala

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

$$\sum_{k\in D} x_{kjk} + \sum_{k\in D}\sum_{i\in N'} x_{ijk} = 1, j \in N' \quad (2)$$

$$x_{kjk} + \sum_{i\in N'} x_{ijk} - x_{jkk} - \sum_{i\in N'} x_{jik} = 0, k \in D, j \in N' \quad (3)$$

$$\sum_{j\in N'} x_{kjk} - \sum_{j\in N'} x_{jkk} = 0, k \in D \quad (4)$$

$$u_i + (U-2)\sum_{k\in D} x_{kik} + \sum_{k\in D} x_{ikk} \le U-1, i \in N' \quad (5)$$

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

$$\sum_{k\in D} x_{kik} + \sum_{k\in D} x_{ikk} \le 1, i \in N' \quad (7)$$

$$u_i - u_j + U\sum_{k\in D} x_{ijk} + (U-2)\sum_{k\in D} x_{jik} \le U-1, \quad i \ne j; i, j \in N' \quad (8)$$

Dari formulasi tersebut, kendala (1) memastikan bahwa ada tepat m_k *salesman* berangkat dari setiap depot $k \in D$. Kendala (2) memastikan bahwa setiap kota dikunjungi tepat satu kali. Kendala (3) merepresentasikan kontinuitas rute untuk kota. Kendala (4) merepresentasikan kontinuitas rute untuk depot. Kendala (5) dan (6) berturut-turut memastikan batas tas dan bawah untuk setiap tur. Kedua kendala ini memaksa u_i harus sama dengan 1, jika *i* adalah kota pertama dalam tur. Kendala (7) memastikan tidak ada tur dengan hanya satu kota saja. Kendala (8) disebut *subtour elimination constraints* (SEC) yang melarang tur tanpa depot sebagai kota awal atau kota akhir yang mungkin terdapat pada solusi.

III. K-MEANS

K-Means adalah salah satu algoritma *clustering* yang digunakan untuk membagi data menjadi beberapa bagian berdasarkan kriteria tertentu. Algoritma yang diusulkan oleh J.B.MacQueen [5] ini dapat mencari pembagian partisi yang mendekati optimal dengan jumlah pembagian partisi yang sudah ditentukan terlebih dahulu. Algoritma K-Means akan mencoba mengelompokkan data dengan memaksimalkan similaritas setiap data dalam suatu *cluster* dan meminimalkan similaritas antar *cluster*. Algoritma ini termasuk dalam kategori *unsupervised* dan biasanya digunakan untuk mengolah data pada penelitian *data mining* dan *pattern recognition*.

Algoritma K-Means sangat populer karena kemudahan dan kesederhanaan dalam implementasinya. Dasar dari algoritma ini adalah dengan meminimalkan *cluster performance index*, *square-error*, dan *error criterion* dalam pembagian paritisi suatu data. Untuk mencari solusi optimal, algoritma K-Means mencoba menemukan *K cluster* yang memenuhi kriteria yang telah ditentukan. Jumlah *cluster* yang harus ditentukan manual oleh peneliti menjadi satu kekurangan dari algoritma *clustering* ini.

Prosedur untuk menjalankan algoritma K-Means adalah sebagai berikut.

1. Penentuan banyak *cluster*, *K*, dan titik-titik awalan sebanyak *K* yang akan digunakan sebagai *centroid* masing-masing *cluster* awal.
2. Setiap titik akan ditugaskan menjadi anggota *centroid* dengan jarak yang terdekat, dan akhirnya membentuk suatu *cluster*.
3. Setelah semua titik mempunyai *cluster* masing-masing, *centroid* baru dipilih berdasarkan jarak terdekat suatu titik ke pusat masing-masing *cluster*.
4. Lakukan langkah 2 dan 3 sampai semua *centroid* tidak berubah atau *stopping condition* telah terpenuhi.

IV. 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.

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.

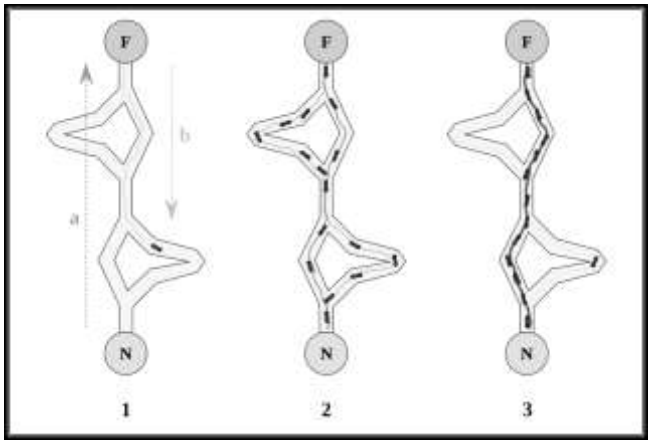
1. Semut buatan memiliki memori.
2. Semut buatan tidak sepenuhnya buta.
3. Semut buatan hidup di lingkungan dimana waktu adalah sesuatu yang diskrit.

Semut menggunakan substansi yang disebut feromon dalam komunikasi antara sesama individu. 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. Probabilitas semut memilih lintasan bertambah sesuai dengan banyaknya semut sebelumnya yang memilih lintasan tersebut. semakin banyak semut yang mengikuti jejak, semakin menarik jejak tersebut untuk diikuti. Gambar 2. merepresentasikan pergerakan semut pada algoritma ACO.

A. Aturan Transisi

Setiap semut ditempatkan di titik sebagai kota awal keberangkatan, yang berbeda dan acak. Semua semut akan mengunjungi satu per satu kota secara berulang kali menggunakan aturan transisi sehingga akan menghasilkan suatu tur.

Diberikan *N* adalah himpunan kota-kota yaitu $N = \{1, 2, ..., n\}$ dan *m* adalah banyak semut yang



Gambar 2. Ilustrasi pergerakan semut pada algoritma ACO

disebar. Probabilitas semut-*k* dari kota-*i* memilih kota-*j* didefinisikan sebagai

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

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

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.

B. Aturan Pembaruan Feromon

Aturan pembaruan feromon 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.

Setelah setiap semut menghasilkan satu tur, yaitu melewati *n* kota, intensitas *trail* akan diperbarui dengan aturan pembaruan feromon global yang didefinisikan sebagai

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

dimana ρ adalah koefisien dengan $(1-\rho)$ merepresentasikan penguapan feromon antara waktu *t* dan $t+n$ dan $\Delta\tau_{ij}^{k}$ adalah kuantitas feromon yang dihasilkan oleh semut-*k* pada sisi (i, j) yang diformulasikan

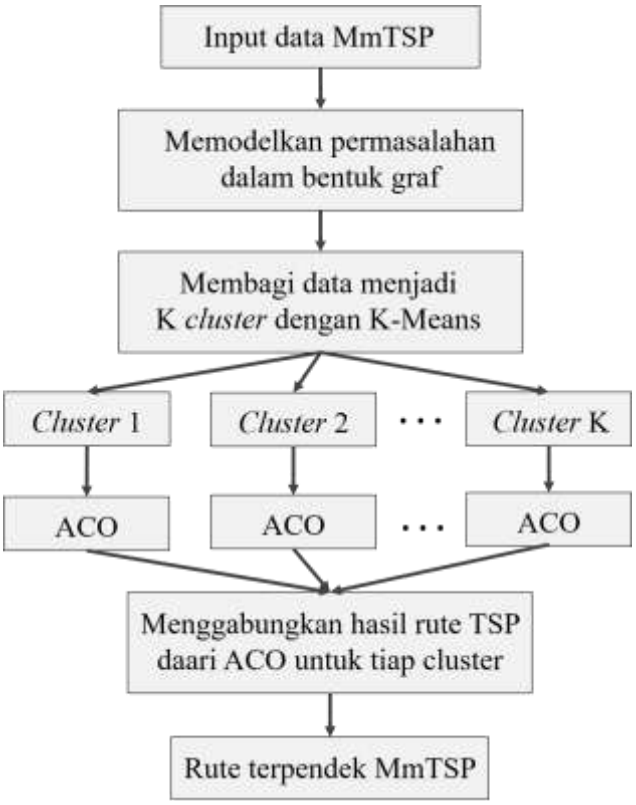
$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L_k}, \text{jika semut-}k \text{ melewati } (i, j) \\ 0, \text{lainnya} \end{cases} \quad (2.19)$$

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

Rute terpendek yang dihasilkan oleh setiap 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.

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 τij(t)
dan Δτij = 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 Δτij = 0
   Kosongkan kembali tabu list
end
Output: tur terpendek
```



Gambar 8. Tahapan metode K-Means ACO untuk menyelesaikan MmTSP

V. METODE YANG DIUSULKAN

Pada penelitian ini diusulkan metode K-Means ACO untuk menyelesaikan MmTSP. K-Means digunakan untuk mencari *cluster* optimal pada MmTSP yang telah dimodelkan dalam bentuk graf. Kota-kota pada MmTSP di-*cluster* menjadi K *cluster* sesuai dengan banyak depot yang ditentukan. Dengan cara seperti ini, akan didapat kota-kota yang jaraknya dekat dengan depot. Setiap *cluster* akan berisi kota-kota yang akan dikunjungi oleh setiap *salesman* pada cluster atau depot tersebut. Kemudian, pada setiap *cluster* akan dicari rute terpendeknya menggunakan ACO. Rute yang dibentuk pada setiap *cluster* merupakan masalah TSP. Sehingga, MmTSP diubah menjadi K rute TSP dengan K-Means. Gabungan rute terpendek hasil dari ACO pada setiap *cluster* merupakan solusi dari rute terpendek MmTSP. Tahapan K-Means ACO untuk menyelesaikan MmTSP ditunjukkan seperti pada Gambar 3.

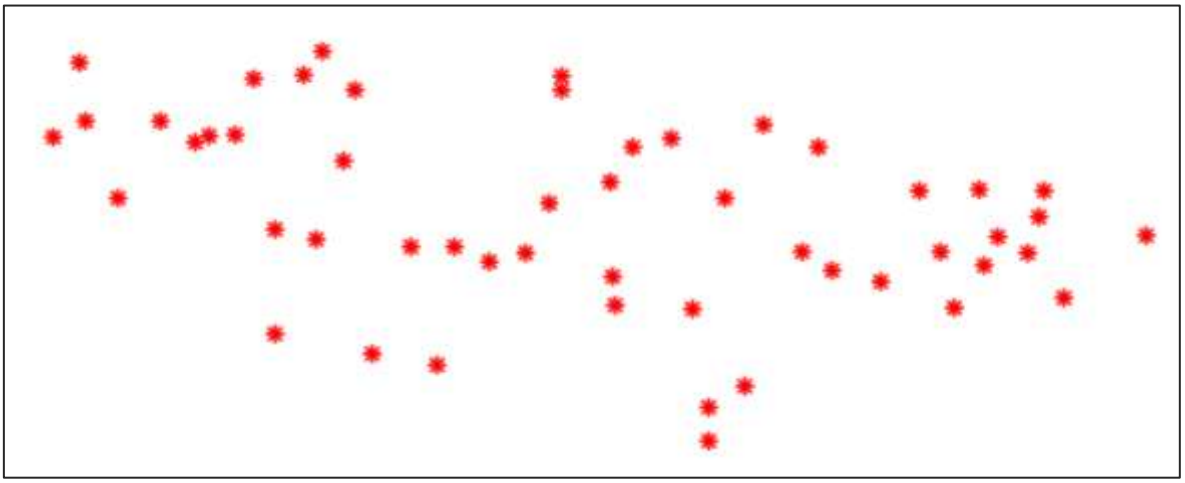
VI. HASIL EKSPERIMEN DAN PEMBAHASAN

Penelitian dilakukan menggunakan Matlab dengan spesifikasi komputer Processor Intel(R) Core(TM) i7 CPU 2.20 GHz dan 4.00 GB RAM. Data yang digunakan pada penelitian ini adalah *dataset* yang diambil dari penelitian mengenai penentuan rute angkutan laut penumpang yang dilakukan oleh Farisi [6]. Data tersebut adalah data yang dipublikasikan oleh Kementerian Perhubungan. Terdapat lima puluh satu pelabuhan beserta waktu tempuh masing-masing pelabuhan ke pelabuhan lainnya yang dapat dikunjungi. Dari lima puluh satu pelabuhan tersebut, lima pelabuhan dipilih sebagai depot. Kendala yang terdapat pada permasalahan ini adalah hanya terdapat satu kapal pada masing-masing depot dan setiap pelabuhan hanya dapat menampung satu kapal. Peta persebaran lima puluh satu pelabuhan tersebut ditunjukkan oleh Gambar Peta persebaran pada Gambar 4 dapat dibuat model graf berdasarkan posisi pelabuhan seperti pada Gambar 5.

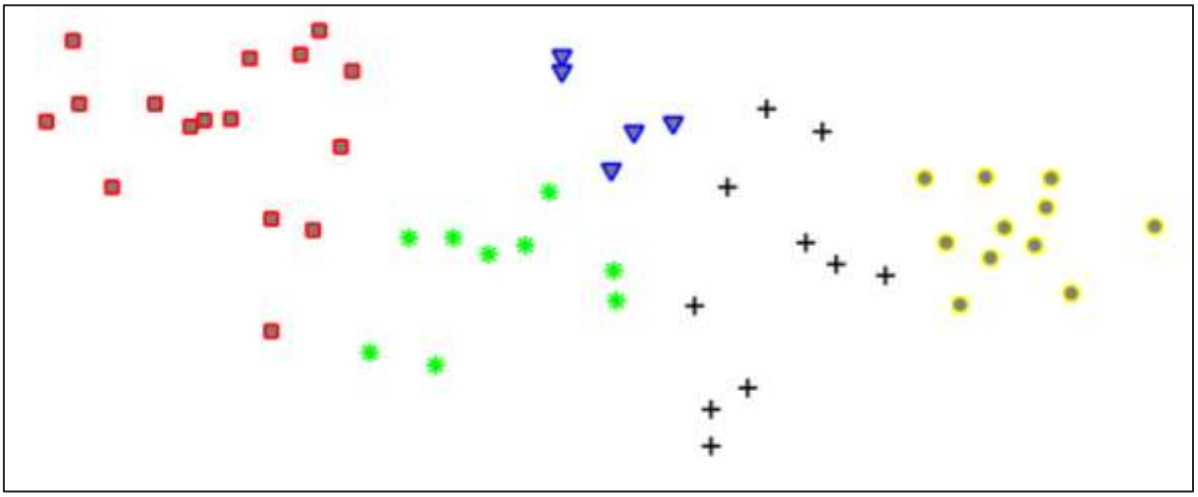
Graf pada Gambar 5 di*cluster* menjadi lima cluster sesuai dengan banyak depot pada permasalahan. Hasil peng*cluster*an dengan K-Means ditunjukkan seperti pada Gambar 6. Kemudian, dicari rute terpendek setiap *cluster* menggunakan ACO.

Pada eksperimen ini, parameter yang digunakan antara lain $\alpha = 1$, $\beta = 5$, $\rho = 0,5, Q = 100$, semut sebanyak jumlah kota [1]. Iterasi yang digunakan sebanyak 500. Dari 50 percobaan yang dilakukan didapat hasil terbaik K-Means ACO adalah 12443 km dengan waktu komputasinya 5,1957 detik. Rata-rata solusinya dari metode yang diusulkan adalah 13549 km dan rata-rata waktu komputasinya 5,2013 detik. Rute yang dihasilkan oleh metode K-Means ACO ditunjukkan pada Gambar 7. Gambar 8 menunjukkan rute angkutan laut penumpang hasil dari K-Means ACO.

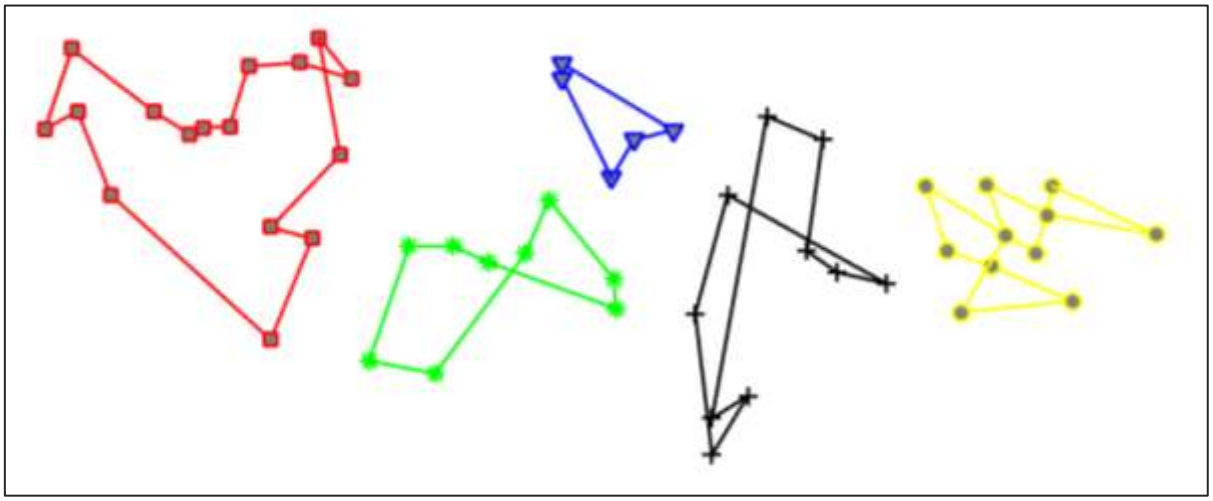
Gambar 4. Persebaran 51 Pelabuhan di Indonesia



Gambar 5. Model graf posisi 51 pelabuhan



Gambar 6. Hasil *cluster* menggunakan K-Means dengan 5 depot



Gambar 7. Hasil rute terpendek menggunakan ACO untuk setiap *cluster*

Gambar 8. Penerapan rute MmTSP hasil rute terpendek ACO pada peta pelabuhan Indonesia

Gambar 9. Rute pada depot Tanjung Perak

Gambar 10. Rute pada depot Sorong

TABEL I
PERBANDINGAN METODE ACO DENGAN K-MEANS ACO PADA 50 KALI PERCOBAAN

Perbandingan	ACO	K-Means ACO
Solusi terbaik (km)	13882	12443
Waktu komputasi dalam mencapai solusi terbaik (detik)	60,2617	5,1957
Rata-rata solusi (km)	14463,24	13549
Rata-rata waktu komputasi (detik)	60,77814	5,2013

Pada kenyataan di lapangan sesuai dengan penelitian [6], pelabuhan-pelabuhan yang menjadi depot antara lain Tanjung Priok, Belawan, Tanjung Perak, Makassar, dan Sorong. Namun, karena metode K-Means ACO mencari *cluster* berdasarkan posisi terdekat, maka penentuan depot dicari secara otomatis. Beberapa depot hasil dari K-Means ACO yang sama dengan realitasnya antara lain Tanjung Priok, Tanjung Perak, dan Sorong. Pada Gambar 8, terlihat bahwa terdapat satu rute salesman yang memuat dua depot antara depot Tanjung Perak dan depot Makassar. Namun, hasil yang didapat K-Means ACO lebih optimal daripada rute dengan depot yang ditentukan secara manual. Selain itu, rute yang didapat menggunakan K-Means ACO jika diterapkan pada lapangan, tidak sesuai dengan realitas yang ada. Pada rute pulau Papua yang ditunjukkan oleh Gambar 9, terdapat loncatan rute dari Tual menuju Nabire. Hal ini terjadi karena pencarian ACO berdasarkan pada jarak antar pelabuhan tanpa memperhatikan lokasi pelabuhan tersebut. Sehingga, hasil dari K-Means ACO ini merupakan hasil optimal berdasarkan jarak pelabuhan yang ada.

Untuk menguji performansi dari metode yang diusulkan, solusi dan waktu komputasi dari K-Means ACO dibandingkan dengan ACO yang dimodifikasi untuk MmTSP pada penelitian [6]. Hasil terbaik yang didapat pada metode ACO untuk 50 kali percobaan adalah 13882 km dengan waktu komputasi 60,2617 detik. Rata-rata solusi dari 50 percobaan untuk metode ACO adalah 14463,24 km dengan rata-rata waktu komputasi 60,77814 detik. Tabel I menunjukkan perbandingan metode K-Means ACO dengan ACO untuk menyelesaikan MmTSP.

Dari Tabel I terlihat bahwa metode K-Means ACO mencapai solusi optimal lebih baik dibandingkan metode ACO dengan waktu komputasi 91,37% lebih cepat. Dari 50 percobaan, metode K-Means ACO mendapatkan rata-rata solusi lebih baik dibandingkan ACO. Dari segi waktu komputasi K-Means ACO lebih cepat 91,44% dibandingkan metode ACO. Hal ini disebabkan pada metode K-Means ACO, rute MmTSP diubah menjadi K-*cluster* rute TSP.

ACO yang dimodifikasi untuk menyelesaikan MmTSP berjalan secara langsung untuk mencari rute dari lima puluh satu pelabuhan. Sedangkan pada metode K-Means ACO, lima puluh satu pelabuhan terlebih dahulu dibagi menjadi lima *cluster* dengan menggunakan K-Means. ACO berjalan untuk tiap *cluster* dengan banyak pelabuhan antara lima sampai enam belas pelabuhan pada setiap *cluster*. Karena banyak pelabuhannya semakin sedikit, maka banyaknya kemungkinan rute yang dapat dicari juga semakin sedikit. Hal ini menyebabkan waktu komputasi yang dibutuhkan untuk menemukan rute terpendek menjadi lebih cepat daripada pencarian rute terpendek dari lima puluh satu pelabuhan secara langsung. Sehingga, ketika K-Means ACO digunakan untuk mencari rute MmTSP akan lebih cepat dibandingkan dengan metode ACO yang dimodifikasi.

VII. KESIMPULAN

Pada penelitian ini diusulkan metode K-Means ACO untuk menyelesaikan *Multi-Depot Multiple Traveling Salesman Problem* (MmTSP). Metode yang diusulkan mengkombinasikan K-Means untuk *mengcluster* kota-kota yang akan dilalui tiap *salesman* dengan ACO untuk mencari rute tiap *cluster*. Tujuannya adalah untuk mengubah permasalahan MmTSP menjadi K rute TSP. Dengan cara demikian, akan meminimalkan waktu komputasi dalam menyelesaikan masalah MmTSP.

Hasil eksperimen menunjukkan bahwa metode K-Means ACO menghasilkan solusi yang lebih baik dengan waktu komputasi 91,44% lebih cepat dibandingkan metode ACO yang dimodifikasi untuk MmTSP. Penentuan depot dari hasil rute yang didapat metode K-Means ACO berbeda dengan kenyataan di lapangan. Namun, total rute yang didapat K-Means ACO lebih kecil dibandingkan dengan rute realitasnya. Sehingga, hasil ini dapat dijadikan rekomendasi penentuan depot pada rute angkutan laut penumpang agar mendapatkan hasil yang optimal. Penelitian lebih lanjut juga diperlukan agar hasil dari penelitian ini dapat diterapkan pada rute angkutan laut penumpang di Indonesia.

DAFTAR PUSTAKA

[1] M. Dorigo, V. Maniezzo, dan A. Colorni, "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, 1996.

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

[3] P. Jinjie dan W. Dingwei, "An Ant Colony Optimization Algorithm for Multiple Travelling Salesman Problem," dalam *Proc. ICICIC*, 2006, hal. 210-213.

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

[5] K. Kim dan H. Ahn, "A recommender system using GA K-means clustering in an online shopping market," Expert Systems with Applications, vol. 34, hal. 1200-1209, Februari 2008.

[6] O.I.R. Farisi, "Penyelesaian *Multi-Depot Multiple Traveling Salesman Problem* Menggunakan *Hybrid Firefly Algorithm – Ant Colony Optimi-*

zation," tesis magister, Jurusan Matematika, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia, 2015.