# Modifikasi ACO untuk Penentuan Rute Terpendek ke Kabupaten/Kota di Jawa

Ahmad Jufri, Sunaryo, dan Purnomo Budi Santoso

*Abstract* – **This research focused on modification ACO algorithm. The purpose of this research was to obtain better performance by modifying the algorithm of Ant Colony Optimization (ACO), particularly in terms of computational speed while maintaining the quality of the solution. Modifications made in this research is the calculation of the probability of the next city to be visit, the ant trail intensity calculations, and modification number of ants to follow the size of the problem.**

**From the results of this research it can be concluded that by modifying the probability of the next city to be visit and the intensity of the ant trail can maintain the quality of the resulting solution with a percentage of 99.8% while the number of ants that used 35% of the size of the problem. In terms of memory usage, modification ACO algorithm is more efficient than the original ACO algorithm with an average of 7%. In addition, the time required to generate the shortest path on average three times faster than the original ACO.**

***Keywords – ACO, ACO Modified, Shortest Path***

*Abstrak* – **Penelitian ini berfokus pada modifikasi ACO untuk penentuan rute terpendek ke Kabupaten/Kota. di Jawa Tujuan penelitian ini adalah untuk mendapatkan kinerja yang lebih baik dengan memodifikasi Algoritma Ant Colony Optimization (ACO), khususnya dalam hal kecepatan komputasi dengan tetap menjaga kualitas solusi. Modifikasi yang dilakukan dalam penelitian ini adalah perhitungan probabilitas pemilihan Kota berikutnya yang akan dikunjungi, perhitungan intensitas jejak Semut, dan modifikasi jumlah Semut mengikuti ukuran permasalahan.**

**Dari hasil pengujian dapat disimpulkan bahwa dengan memodifikasi probabilitas pemilihan Kota berikutnya dan intensitas jejak Semut dapat menjaga kualitas solusi yang dihasilkan dengan prosentase 99,8% walaupun jumlah Semut yang digunakan 35% dari ukuran masalah. Dalam hal pemakaian memori, Algoritma Modifikasi ACO secara rata-rata lebih efisien dari Algoritma ACO dengan prosentase 7%. Selain itu waktu yang dibutuhkan untuk menghasilkan rute terpendek rata-rata 3 kali lebih cepat daripada ACO asli.**

***Kata Kunci - ACO, Modifikasi ACO, Rute Terpendek***

Ahmad Jufri adalah mahasiswa Program Magister Teknik Elektro Universitas Brawijaya, Malang, Indonesia (phone : 085785070371; email: ahmad.stikma@yahoo.com)

Sunaryo adalah Ketua Program Studi Geofisika, Jurusan Fisika, FMIPA Universitas Brawijaya, Malang, Indonesia, phone: 08123354285, email: sunaryo.geofis.ub@gmail.com, sunaryo@ub.ac.id

Purnomo Budi Santoso adalah Kepala Laboratorium Komputer Teknik Industri Universitas Brawijaya, Malang, Indonesia (phone: 081216709809 ; email: pbsabn@ub.ac.id)

## I. PENDAHULUAN

KABUPATEN didefinisikan sebagai daerah swatantra Tingkat II yang dikepalai oleh Bupati, setingkat dengan Kotamadya, dan merupakan bagian langsung dari Provinsi yang terdiri atas beberapa Kecamatan. Sedangkan Kota didefinisikan sebagai daerah pemukiman yang terdiri atas bangunan rumah yang merupakan kesatuan tempat tinggal dari berbagai lapisan masyarakat [1]. Menurut buku induk kode data dan wilayah Tahun 2013, jumlah Kabupaten di Indonesia sebanyak 399, sedangkan jumlah Kota di Indonesia berjumlah 98 Kota [2]. Provinsi Jawa Barat sendiri terdiri dari 17 Kabupaten dan 9 Kota. Untuk Jawa Tengah terdiri dari 29 Kabupaten dan 6 Kota. Sedangkan Jawa Timur memiliki 29 Kabupaten dan 9 Kota. Setiap Kabupaten/Kota dihubungkan oleh jalur, baik udara, laut maupun darat. Untuk Kabupaten/Kota di Jawa sebagian besar dihubungkan dengan jalur darat dengan jarak yang berbeda-beda.

Dengan banyaknya Kabupaten/Kota, bukan hal yang mudah untuk mencari dan memilih rute yang terbaik atau optimal dari satu Kabupaten/Kota ke Kabupate/Kota yang lain. Permasalahan pencarian rute terpendek dikenal juga dengan istilah *Traveling Salesman Problem* (TSP). Sampai saat ini telah banyak dikembangkan metode untuk memecahkan masalah TSP, baik menggunakan pendekatan deterministik maupun probabilistik. Untuk permasalahan dengan kandidat solusi yang banyak pendekatan probabilistik lebih sesuai. *Ant Colony Optimization* (ACO) merupakan salah satu Algoritma probabilistik yang terbukti mampu memberikan solusi yang optimal untuk permasalahan TSP.

Dari hasil simulasi yang dilakukan [3] menunjukkan bahwa algoritma ACO menghasilkan solusi terbaik rata-rata 24,9% lebih baik daripada Algoritma Genetik. Tetapi dari segi waktu komputasi rata-rata Algoritma Genetik 34% masih lebih baik daripada Algoritma ACO. Dari penelitian [4] dengan membandingkan Algoritma Genetik, ACO, dan *Simulated Anneling* (SA) disimpulkan bahwa Algoritma ACO menghasilkan menghasilkan kualitas solusi 4.2% lebih baik daripada yang lain, tetapi dalam sisi waktu komputasi Algoritma ACO 1,7 kali lebih lambat daripada Algoritma Genetik.

Dari permasalahan diatas maka pada penelitian ini akan dilakukan modifikasi pada Algoritma ACO yang terdiri dari modifikasi jumlah Semut secara otomatis mengikuti jumlah titik/kot, modifikasi perhitungan

intensitas jejak Semut dan perhitungan probabilitas kota yang akan dikunjungi. Hal ini dilakukan untuk mendapatkan kinerja yang lebih baik khususnya dalam hal waktu komputasi dengan tetap menjaga kualitas solusi yang dihasilkan.

## II. TINJAUAN PUSTAKA

Dalam memodifikasi Algoritma ACO perlu dilakukan kajian terhadap pustaka yang berhubungan dengan rute terpendek dan Algoritma ACO. Hal ini agar peneliti memiliki landasan yang kuat dalam melakukan proses modifikasi Algoritma ACO.

### *A. Rute Terpendek*

Menurut [1] rute didefinisikan sebagai jarak atau arah yang harus diturut, ditempuh, atau dilalui. Dari pengertian tersebut maka rute terpendek bisa didefinisikan sebagai jarak atau arah terpendek yang harus ditempuh atau dilalui dari satu titik (sebagai sumber) ke titik lainnya (sebagai tujuan).

### *B. Algoritma ACO*

Algoritma *Ant Colony Optimization* (ACO) merupakan Algoritma yang diadopsi dari perilaku Semut [5]. Berikut langkah Algoritma ACO:

1. Inisialisasi harga parameter algoritma yang terdiri dari:
   a. Jumlah Kota (n) beserta koordinatnya (x dan y) atau jarak antar Kota ($d_{ij}$).
   b. Tetapan siklus Semut (Q).
   c. Tetapan pengendalian intensitas jejak Semut ($\alpha$), dimana $\alpha \geq 0$.
   d. Tetapan pengendali visibilitas ($\beta$).
   e. Jumlah Semut (m).
   f. Tetapan penguapan jejak Semut ($\rho$), dimana $0<\rho<1$.
   g. Jumlah siklus maksimum (NCmaks).
   h. Intensitas jejak Semut antar Kota ($\tau_{ij}$).
2. Mengisi titik pertama ke dalam *tabu list*.
3. Penyusunan rute kunjungan setiap Semut ke setiap titik yang dilewati. Untuk menentukan probabilitas titik untuk dikunjungi digunakan rumus berikut:

$$p_{ij}^{k} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum_{k' \in \{N-tabu_k\}} [\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}, & \text{utk j} \in \{\text{N - tabu}_k\} \\ 0, & \text{utk} \in \text{j lainnya} \end{cases} \quad (1)$$

4. Perhitungan panjang rute untuk setiap Semut.
   a. Perhitungan panjang rute. Perhitungan ini dilakukan berdasarkan *tabu*$_k$ masing-masing dengan persamaan berikut :

$$L_k = d_{tabu_k(n),tabu_k(1)} + \sum_{s=1}^{n-1} d_{tabu_k(s),tabu_k(s+1)} \quad (2)$$

   dengan $d_{ij}$ adalah jarak antara Kota i ke Kota j yang dihitung berdasarkan persamaan:

$$d_{ij} = \sqrt{(x_i - y_i)^2 + (y_i - y_j)^2} \quad (3)$$

   b. Pencarian rute terpendek lokal (tiap siklus) & global (semua siklus).
   c. Perhitungan intensitas jejak Semut antar titik dengan rumus:

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \quad (4)$$

   Dengan $\Delta\tau_{ij}$ adalah perubahan harga intensitas jejak kaki Semut antar Kota yang dihitung dengan rumus:

$$\Delta\tau_{ij} = \begin{cases} \dfrac{Q}{L_k}, & \text{utk (i, j)} \in \text{Kota asal dan Kota tujuan dlm tabu}_k \\ 0, & \text{utk (i, j) lainnya} \end{cases} \quad (5)$$

5. Perhitungan harga intensitas jejak Semut antar titik untuk siklus berikutnya dengan rumus:

$$\tau_{ij} = \rho.\tau_{ij} + \Delta\tau_{ij} \quad (6)$$

   Untuk siklus berikutnya nilai ini akan direset menjadi nol (0).
6. Mengosongkan tabu list.
7. Mengulangi langkah 2 sampai siklus maksimum.

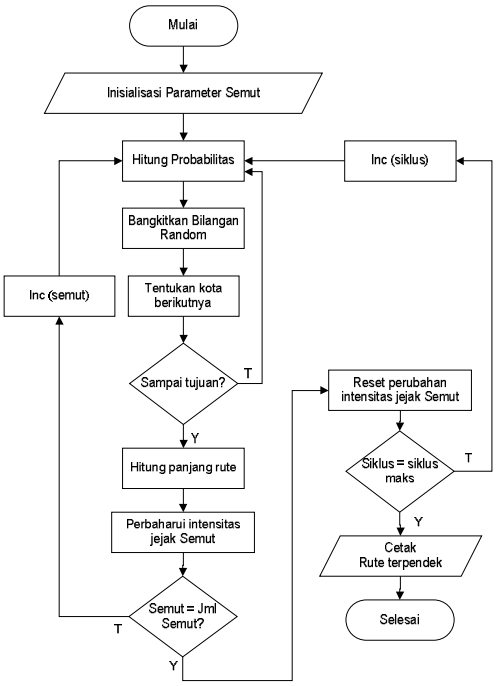
## III. METODOLOGI PENELITIAN

Modifikasi ACO untuk menentukan rute terpendek diawali dengan penelusuran terhadap implementasi algoritma ACO pada penelitian terdahulu diikuti dengan pengujian awal algoritma ACO pada kasus TSP untuk mengetahui pengaruh parameter-parameter masukan terhadap kinerja dan solusi yang dihasilkan oleh algoritma ACO, kemudian dilanjutkan dengan memodifikasi parameter yang memiliki pengaruh signifikan terhadap kinerja algoritma ACO serta solusi yang dihasilkan. Dari kedua Algoritma ini kemudian di uji kembali dan dibandingkan. Kemudian hasil pengujian di analisis menggunakan pendekatan posteriori (*poteriori testing*) untuk mengetahui kinerja dari kedua algoritma tersebut. Dan yang terakhir diambil kesimpulan.

### *A. Pengaruh parameter ACO terhadap kualitas solusi*

Setiap parameter masukan yang ada di dalam suatu algoritma memiliki pengaruh terhadap keluaran algoritma tersebut. Algoritma ACO memiliki beberapa parameter masukan seperti $\alpha$, $\beta$, Q, $d_{ij}$, $\rho$, $\tau_{ij}$, $NC_{maks}$, m dan n. Setiap parameter memiliki pengaruh yang berbeda-beda. Dari penelitian terdahulu yang dilakukan oleh [6] dapat diketahui bahwa kualitas solusi yang dihasilkan oleh algoritma ACO bergantung pada jumlah Semut. Penelitian ini diperkuat dengan hasil penelitian yang dilakukan oleh [7] yang menyatakan bahwa ACO menghasilkan solusi yang paling optimal ketika jumlah Semut sama dengan jumlah Kota dan nilai $\alpha$ harus mendekati 1. Hal ini dapat dilihat pada diagram alir perhitungan rute terpendek dengan algotima ACO pada Gambar 1. Dari gambar tersebut terlihat bahwa untuk mendapatkan rute terpendek Algoritma ACO harus melakukan proses yang berulang-ulang. Ada tiga perulangan yang terjadi selama mencari rute terpendek, yaitu pertama probabilitas pemilihan Kota berikutnya sampai ditemukannya titik/Kota tujuan. Kedua, proses

pencarian rute terpendek yang dilakukan oleh Semut pertama sampai semut terakhir. Dan ketiga, pencarian rute terpendek yang dilakukan oleh setiap Semut mulai dari siklus pertama sampai dengan siklus yang terakhir. Jika dalam pengujian berpijak pada penelitian yang dilakukan oleh [10], dengan jumlah titik/Kota=20 maka proses perulangan yang terjadi akibat jumlah Semut dan jumlah siklus paling tidak berjumlah 400 perulangan. Dengan banyaknya perulangan tersebut maka waktu yang diperlukan untuk mendapatkan rute terpendek akan semakin lama. Sedangkan dari segi pemakaian memori yang digunakan selain bergantung dari jumlah Semut dan jumlah siklus, juga bergantung jumlah titik/Kota yang dikunjungi selama proses pencarian trute terpendek. Berikut diagram alir pencarian rute terpendek menggunakan algoritma ACO.

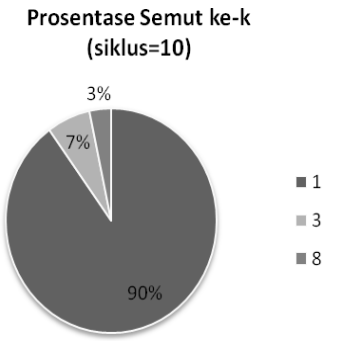



Gambar 1 Diagram alir Algoritma ACO

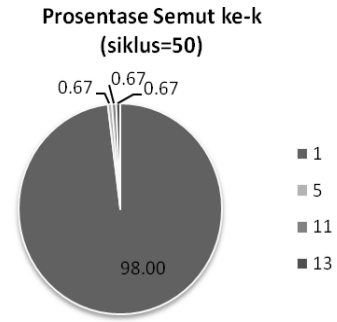
### *B. Pengujian awal ACO*

Pengujian awal di dalam penelitian ini dilakukan dengan tujuan untuk mengetahui parameter-parameter yang berpengaruh terhadap kinerja algoritma Semut, khususnya jumlah Semut dan siklus terhadap solusi yang dihasilkan. Pengujian awal terhadap algoritma ACO untuk permasalahan TSP menggunakan parameter sebagai berikut: $\alpha = 1$, $\beta = 5$, $Q = 100$, $\rho = 0.5$, $\tau_{ij} = 0.1$, $m = 25$, dan $n = 30$. Pengujian ini dilakukan dengan menggunakan data matrik jarak Kabupaten/Kota di Jawa Timur dan dilakukan sebanyak 3 kali untuk jumlah siklus 10, 50, dan 100. Dari hasil pengujian kemudian akan di rata-rata dan diambil prosentasenya.

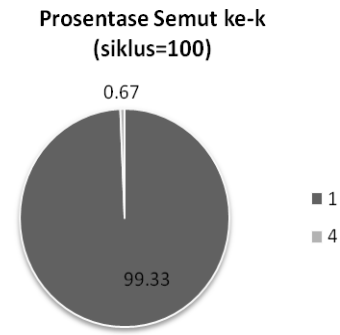
Dari hasil pengujian awal terlihat bahwa rute terpendek lokal lebih banyak dihasilkan oleh Semut nomor 1, dengan prosentase diatas 90%. Dari sini terlihat bahwa Semut nomor 1 lebih mendominasi terhadap perolehan rute terpendek lokal (optimum lokal). Nilai terkecil dari optimum lokal yang diperoleh dari mulai siklus pertama sampai dengan siklus terakhir akan diambil sebagai rute terpendek global (optimum global). Berikut gambar prosentase hasil pengujian Algoritma ACO.




Gambar 2 Prosentase Semut penghasil rute terpendek dengan siklus = 10




Gambar 3 Prosentase Semut penghasil rute terpendek dengan siklus = 50




Gambar 4 Prosentase Semut penghasil rute terpendek dengan siklus = 100

### *C. Modifikasi ACO*

Dari masalah diatas maka pada penelitian ini dilakukan modifikasi Algoritma ACO dengan berdasar pada pengujian awal dan penelitian yang dilakukan oleh [8]. Dari penelitian yang dilakukan oleh [8] dengan memodifikasi Algoritma ACO yang digabungkan dengan Insert, Swap, dan 2-opt yang disebut sebagai NMACO dan digunakan untuk menyelesaikan masalah *Multiple Traveling Salesman Problem* (MTSP) menunjukkan bahwa Algoritma NMACO berhasil

meningkatkan efisiensi dari Algoritma ACO. Modifikasi yang dilakukan pada penelitian ini meliputi:

1. Memodifikasi jumlah Semut yang melakukan perjalanan untuk mencari rute terpendek, dimana jumlah Semut akan secara otomatis disesuaikan dengan ukuran masalah (jumlah titik atau Kota). Masukan jumlah Semut pada *Modified* ACO dalam penelitian ini akan diset otomatis dengan ketentuan sebagai berikut:
   a. Jika jml_Kota >= 25 maka jml_Semut = 25.
   b. Jika jml_Kota > 25 maka jml_Semut=0.35*jml_Kota.

   Tujuan modifikasi jumlah Semut disini untuk mengurangi jumlah perulangan sehingga dapat mengurangi waktu komputasi. Akan tetapi pengurangan jumlah Semut juga dapat mengurangi kualitas solusi yang dihasilkan apalagi jika ukuran masalahnya bertambah.
2. Memodifikasi perhitungan intensitas jejak Semut yang ditinggalkan. Pada Modifikasi ACO akumulasi intensitas jejak intensitas jejak Semut antar titik/Kota dihitung dengan rumus:

$$\tau_{ij}(t+1) = (1-\rho).\tau_{ij}(t) + \Delta\tau_{ij}^{cbs}(t) + \Delta\tau_{ij}^{bks}(t) \quad (7)$$

   Dimana,

   $\Delta\tau_{ij}^{cbs}(t)$ = perubahan intensitas jejak Semut pada Semut yang memperoleh solusi optimum lokal pada iterasi saat ini.

   $\Delta\tau_{ij}^{bks}(t)$ = perubahan intensitas jejak Semut pada Semut yang memperoleh solusi optimum lokal mulai dari iterasi awal sampai iterasi saat ini.
3. Memodifikasi proses perhitungan probabilitas pemilihan Kota berikutnya yang akan dikunjungi. Pada Modifikasi ACO probabilitas pemilihan Kota berikutnya yang akan dikunjungi dihitung berdasarkan rumus:

$$p_{ij}^{k} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}.[k_{ij}]^{\lambda}}{\sum_{k' \in \{N-tabu_k\}} [\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}.[k_{ij}]^{\lambda}}, & \text{utk } j \in \{N - tabu_k\} \\ 0, & \text{utk } j \text{ lainnya} \end{cases} \quad (8)$$

   Dimana,

   $k_{ij}^{\lambda}$ = jarak kombinasi antara dua titik i dan j yang dihitung dengan rumus:

$$k_{ij}^{\lambda} = \alpha.d_{i1} + \beta.d_{1j} - \lambda.d_{ij} \quad (9)$$

   $\lambda$ = parameter kontrol yang dimasukkan oleh pengguna.

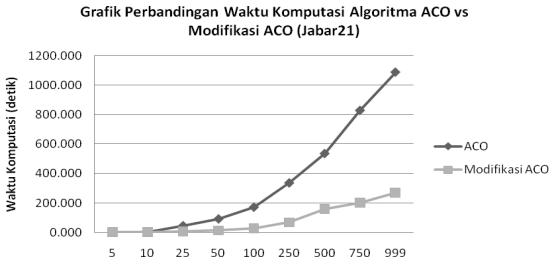
   Modifikasi pada perhitungan intensitas jejak Semut dan probabilitas pemilihan Kota berikutnya diadopsi dari penelitian yang dilakukan oleh [8]. Tujuan dari modifikasi pada perhitungan intensitas jejak Semut (no. 2) dan probabilitas pemilihan Kota berikutnya (no. 3) adalah untuk mempertahankan solusi yang dihasilkan oleh Algoritma Modifikasi ACO agar tetap optimal walaupun jumlah Semut yang melakukan perjalanan untuk memperoleh rute terpendek dikurangi.

## IV. HASIL UJI COBA

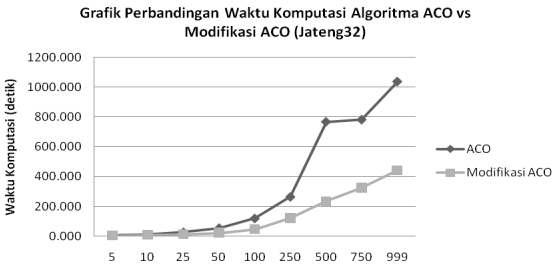
Pengujian dilakukan menggunakan dataset yang dibuat berdasarkan data jarak antar Kabupaten/Kota di Propinsi Jawa Barat, Jawa Tengah, dan Jawa Timur. Dataset yang digunakan berbentuk matrik jarak antar titik/ Kota. Parameter yang digunakan dalam penelitian ini adalah sebagai berikut: $\alpha = 1$, $\beta = 5$, $Q = 100$, $\rho = 0.5$, $\tau_{ij} = 0.1$, dan n mengikuti jumlah Kabupaten/Kota yang ada di tiap Provinsi. Untuk jumlah Semut pada Algoritma ACO menggunakan nilai 25. Sedangkan pada Algoritma Modifikasi ACO mengikuti jumlah Kota (ukuran masalah). Jika jumlah Kabupaten/Kota kurang dari atau sama dengan 25 maka jumlah Semut diisi otomatis 25, dan jika jumlah Kabupaten/Kota lebih dari 25 maka jumlah Semut akan diisi otomatis 35% dari jumlah Kabupaten/Kota.

### *A. Hasil Analisa Perbandingan Waktu Komputasi Algoritma ACO dan Modifikasi ACO*

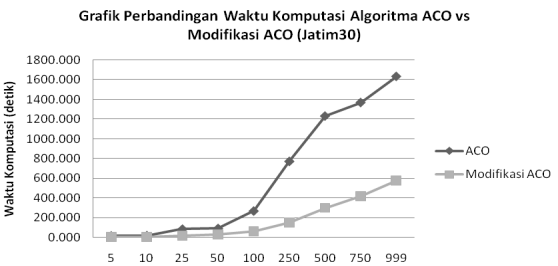
Berdasarkan hasil pengujian yang dilakukan terhadap Algoritma ACO dan Modifikasi ACO untuk pencarian rute terpendek di Kabupaten/Kota di Jawa Barat, Jawa Tengah, dan Jawa Timur diperoleh hasil sebagai berikut:




Gambar 5 Perbandingan Waktu Komputasi ACO dan Modifikasi ACO (Jabar21)




Gambar 6 Perbandingan Waktu Komputasi ACO dan Modifikasi ACO (Jateng32)



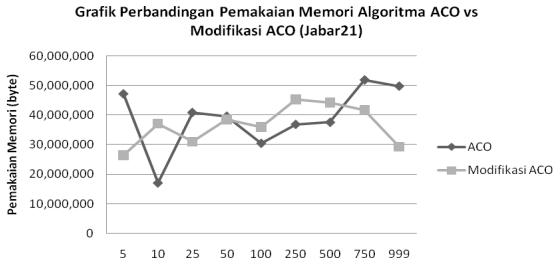

Gambar 7 Perbandingan Waktu Komputasi ACO dan Modifikasi ACO (Jatim30)

Dari Gambar 5, 6, dan 7 terlihat bahwa waktu yang diperlukan Algoritma Modifikasi ACO untuk

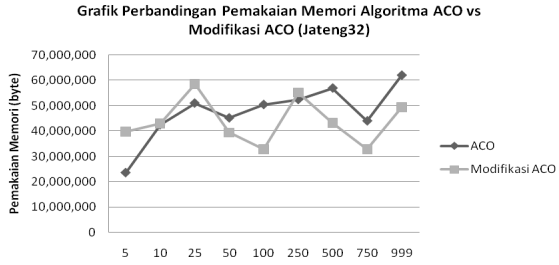
menghasilkan rute terpendek lebih baik daripada Algoritma ACO. Semakin besar siklusnya, semakin besar pula perbedaannya. Peningkatan kecepatan akibat dari pengurangan jumlah Semut yang melakukan pencarian rute terpendek lokal untuk setiap siklusnya. Dengan mengurangi jumlah Semut menjadi 35% dari jumlah Kota maka waktu yang dibutuhkan untuk memperoleh rute terpendek lokal setiap siklusnya menjadi kurang lebih sepertiga dari waktu yang diperlukan oleh ACO asli untuk mendapatkan rute terpendek atau dengan kata lain kecepatannya meningkat sekitar tiga kali lipat dari ACO asli. Tetapi pengurangan jumlah Semut juga bukan satu-satunya faktor yang menyebabkan waktu komputasi algoritma Modifikasi ACO lebih baik dari ACO.

### *B. Hasil Analisa Perbandingan Pemakaian Memori Algoritma ACO dan Modifikasi ACO*

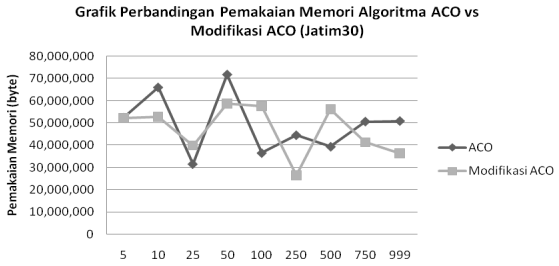
Dari pengujian yang dilakukan terhadap Algoritma ACO dan Modifikasi ACO diperoleh perbandingan hasil pemakaian memori untuk mendapatkan rute terbaik sebagai berikut:




Gambar 8 Perbandingan Pemakaian Memori ACO dan Modifikasi ACO (Jabar21)




Gambar 9 Perbandingan Pemakaian Memori ACO dan Modifikasi ACO (Jateng32)

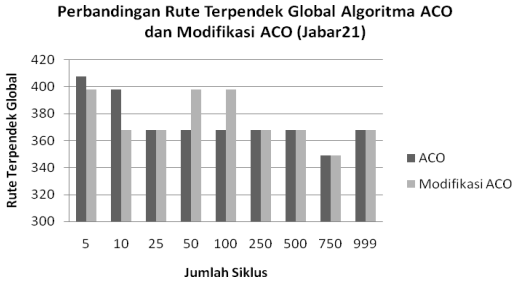



Gambar 10 Perbandingan Pemakaian Memori ACO dan Modifikasi ACO (Jatim30)

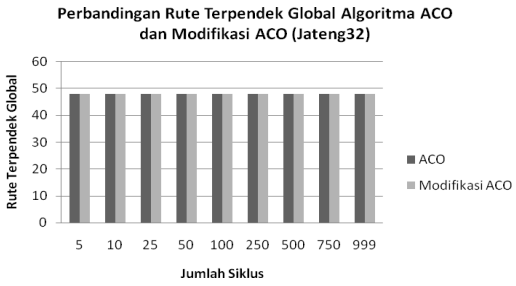
Dari ketiga Gambar 8, 9, dan 10 hasil pengujian diatas terlihat bahwa pemakaian memori oleh kedua Algoritma sama-sama mengalami fluktuasi. Hal ini dikarenakan untuk pemilihan titik/Kota menggunakan probabilitas yang kemudian dibandingkan dengan bilangan random yang dibangkitkan, sehingga kesempatan setiap titik/Kota yang akan dikunjungi adalah sama. Namun secara umum Algoritma Modifikasi ACO menggunakan memori rata-rata yang lebih rendah 7% daripada Algoritma ACO. Dalam hal ini tidak ada perbedaan yang signifikan karena walaupun jumlah Semut yang melakukan perjalan pada Modifikasi ACO dikurangi menjadi 35% dari jumlah titik/Kota (ukuran masalah). Perbedaan penggunaan memori antara Algoritma ACO dan Modifikasi ACO tidak terlalu besar karena variabel yang digunakan untuk proses perhitungan untuk mendapatkan rute terpendek pada Algoritma Modifikasi ACO lebih banyak daripada yang digunakan pada Algoritma ACO. Jadi dari sisi perulangan jumlah Semut mengurangi pemakaian memori Algoritma Modifikasi ACO, tetapi dari sisi proses perhitungan untuk mendapatkan rute terpendek Algoritma Modifikasi ACO memakai ruang memori yang lebih besar daripada Algoritma ACO.

### *C. Hasil Analisa Perbandingan Rute Terpendek yang dihasilkan oleh Algoritma ACO dan Modifikasi ACO*

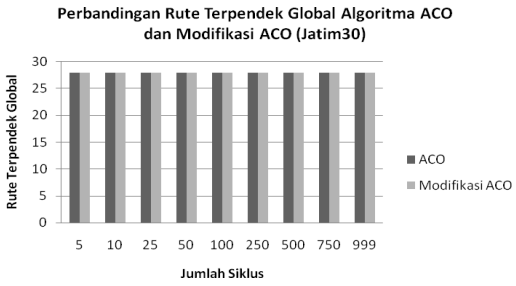
Berikut hasil perbandingan rute terpendek yang dihasilkan oleh Algoritma ACO dan Modifikasi ACO.




Gambar 11 Perbandingan Rute Terpendek ACO dan Modifikasi ACO (Jabar21)




Gambar 12 Perbandingan Rute Terpendek ACO dan Modifikasi ACO (Jateng32)




Gambar 13 Perbandingan Rute Terpendek ACO dan Modifikasi ACO (Jatim30)

Dari Gambar 11 terlihat bahwa kualitas solusi yang dihasilkan oleh Algoritma Modifikasi ACO dalam menghasilkan rute terpendek hampir sama dengan kualitas solusi yang dihasilkan oleh Algoritma ACO (masih ada perbedaan). Pada kasus Jabar21 (Gambar 11) kualitas solusi yang dihasilkan Modifikasi ACO mendekati solusi yang dihasilkan Algoritma ACO dengan prosentase 99,41%. Sedangkan pada kasus Jateng32 dan Jatim30 (Gambar 12 dan 13) solusi yang dihasilkan sama persis. Secara rata-rata kualitas solusi yang dihasilkan oleh Algoritma Modifikasi ACO untuk permasalahan pencarian rute terpendek di Jawa Barat, Jawa Tengah dan Jawa Timur 99,88% mendekati solusi yang dihasilkan ACO yang asli.

## V. KESIMPULAN

Dari Berdasarkan data-data hasil pengujian terhadap implementasi Algoritma ACO dan Modifikasi ACO untuk menyelesaikan permasalahan rute terpendek dengan menggunakan Dataset Jabar21, Jateng32, dan Jatim30 dapat ditarik kesimpulan sebagai berikut:

1. Modifikasi Algoritma ACO dengan memberikan nilai secara otomatis terhadap parameter jumlah Semut sebesar 35% dari ukuran masalah terbukti mampu meningkatkan kecepatan pencarian rute terpendek sebesar 3 kali dari kecepatan Algoritma ACO yang asli. Sedangkan dari sisi pemakaian memori Modifikasi ACO terbukti lebih efisien dengan rata-rata pemakaian memori 93% dari Algoritma ACO yang asli.
2. Modifikasi pada perhitungan intensitas jejak Semut antar titik dan probabilitas pemilihan Kota berikutnya dengan mempertimbangkan intensitas jejak Semut terbaik siklus sebelumnya dan internsitas jejak Semut selama siklus berjalan (awal sampai saat ini) terbukti mampu mempertahankan kualitas solusi yang dihasilkan sebesar 99,8% untuk permasalahan jarak terpendek di Jawa Barat, Jawa Tengah, dan Jawa Timur kendati jumlah Semut dikurangi.

Pada penelitian berikutnya diharapkan Algoritma Modifikasi ACO dapat di *hybrid* dengan metode lainnya, khususnya yang dapat mengoptimalkan kualitas pencarian rute terpendek lokal.

## REFERENSI


[1] Kamus Besar Bahasa Indonesia: http://kbbi.web.id. Diakses tanggal 12 Mei 2014.

[2] Buku Induk Kode Data dan Wilayah: http://www.kemendagri.go.id. Diakses tanggal 25 November 2014.

[3] Zukhri Z. dan Paputungan V. *A Hybrid Optimization Algorithm Based on Genetic Algorithm and Ant Colony Optimization*. International Journal of Artificial Intelligence & Applications (IJAIA), Vol. 4 , No. 5, September 2013.

[4] J. L. Pasquier, dkk. *A Comparative Study of three Metaheuristics applied to the Traveling Salesman Problem*. Sixth Mexican International Conference on Artificial Intelligence, Special Session. 2007.

[5] Dorigo, M., dan Stutzle, T. *The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances*.

[6] I. Brezina Jr. dan Z. Čičková. *Solving the Traveling Salesman Problem Using the Ant Colony Optimization*. Management Information Systems, Vol. 6, No. 4, 2011, pp. 010-014.

[7] M. Aggarwal dan Saroj. *Compute Traveling Salesman Problem Using Ant Colony Optimization*. International Journal of Computing & Business Research. 2012.

[8] M. Yousefikhoshbakht, F. Didehvar, dan F. Rahmati. *Modification of the Ant Colony Optimization for Solving the Multiple Traveling Salesman Problem*. Romanian Journal of Information Science and Technology. Vol. 16, No. 1, 2013, pp. 65-80.