EDUKOM 1 (2) (2014)

Edu Komputika Journal

http://journal.unnes.ac.id/sju/index.php/edukom

IMPLEMENTASI METODE *ANT COLONY* UNTUK *TRAVELING SALESMAN PROBLEM* MENGGUNAKAN *GOOGLE MAPS* PADA KOTA-KOTA DI JAWA TENGAH

Awang Harizka✉ dan Feddy Setio Pribadi

Jurusan Teknik Elektro, Fakultas Teknik, Universitas Negeri Semarang, Indonesia

Info Artikel

Sejarah Artikel:
Diterima Oktober 2014
Disetujui Oktober 2014
Dipublikasikan Oktober 2014

Keywords:
Ant Colony Optimization; Traveling Salesman Problem; Google Maps; Local Search 2-opt; Greedy.

Abstrak

Tujuan penelitian ini adalah mengimplementasikan metode Ant Colony untuk Traveling Salesman Problem (TSP) dengan memanfaatkan Google Maps studi kasus pada kota-kota di Jawa Tengah. Algoritma Ant System yang merupakan salah satu algoritma Ant Colony Optimization (ACO) digunakan untuk membangun sebuah rute yang optimal untuk Traveling Salesman Problem pada Google Maps. Dari hasil pengujian untuk inputan kota kurang dari 10, algoritma Ant System mampu memberikan hasil yang optimal. Sedangkan untuk inputan kota lebih dari 10, algoritma Ant System tidak mampu memberikan hasil yang optimal. Oleh karena itu untuk menambah kinerja algoritma Ant System ditambahkan algoritma Local Search 2opt. Perbandingan hasil perjalanan yang dihasilkan kedua algoritma Ant System sebelum dan setelah ditambah dengan algoritma Local Search 2-opt juga dibandingkan dengan algoritma Greedy. Hasil pengujian menunjukan algoritma Ant System Local Search mempunyai bobot jarak dan waktu tempuh yang paling kecil dari dua algoritma yang lain, serta jalur Traveling Salesman Problem yang dibangun optimal. Pada hasil waktu komputasi dalam perhitungan jalur Traveling Salesman Problem, algoritma Greedy membutuhkan waktu komputasi yang paling sedikit dibandingkan dengan algoritma Ant System dan Ant System Local Search.

Abstract

The purpose of this study was to implemented Ant Colony method to solve the Traveling Salesman Problem (TSP) using Google Maps on cities of Central Java. Ant System algorithm is one of the Ant Colony Optimization (ACO) algorithm to determine the optimum route on Google Maps and solves Traveling Salesman Problem. Based on results for 10 cities input, the Ant System algorithm can solves optimum route, the otherwise where input is more than 10 cities it can't solve optimum route. Therefore to improve performance of Ant System algorithm, the Local Search 2-opt algorithm added to Ant System algorithm. The comparison of the results of route optimum construction both algorithm, that shows the Ant System algorithm with Local Search 2-opt have the best minimum cost tour. And the best results of computation time was produced by Greedy algorithm.

✉ Alamat korespondensi:
Gedung E6 Lantai 2 FT Unnes
Kampus Sekaran, Gunungpati, Semarang, 50229
E-mail: aharizka.tugas@gmail.com

ISSN 2252-6811

PENDAHULUAN

Google Maps merupakan sebuah layanan peta virtual gratis dari Google yang dapat diakses melalui http://maps.google.com. Google Maps tidak hanya menawarkan peta virtual dan citra satelit dari seluruh penjuru dunia melainkan juga perencanaan rute perjalanan atau petunjuk arah dari satu kota awal ke kota tujuan disertai dengan informasi yang diperlukan, misalnya saja jarak tempuh, waktu tempuh dan opsi rute perjalanan.

Informasi yang ditunjukkan pada Google Maps akan sangat membantu bagi penggunanya yang ingin melakukan sebuah perjalanan. Layanan ini sangat mudah digunakan untuk melakukan perencanaan perjalanan dari kota awal yang diinginkan menuju ke kota tujuan, tetapi dalam Google Maps tidak memberikan perencanaan perjalanan untuk mengunjungi banyak kota dengan hasil yang optimal. Para penggunalah yang harus dengan bijak untuk memasukkan kota awal ke kota tujuan dan kembali lagi ke kota awal secara terurut pada kotak input yang ada pada layanan Google Maps. Petunjuk arah yang ditampilkan oleh Google Maps hanya akan sesuai urutan saja.

Permasalahan perjalanan untuk mengunjungi banyak kota dan setiap kota tepat satu kali dikunjungi, dimulai dan diakhiri pada kota yang sama disebut sebagai Masalah Penjaja Keliling atau *Traveling Salesman Problem* (Jong Jek Siang, 2011:253).

Terdapat beragam teori algoritma yang banyak diaplikasikan terhadap permasalahan *Traveling Salesman Problem* seperti, algoritma Brute Force, algoritma Greedy, algoritma Genetic, dan algoritma Ant System. Berdasarkan hasil penelitian yang dilakukan oleh M. Dorigo dan Thomas Stutzle membuktikan bahwa metode Ant Colony mampu menyelesaikan *Traveling Salesman Problem* (Dorigo dan Stutzle, 2004: 40).

Ant Colony System (ACS) yang merupakan salah satu algoritma pada metode *Ant Colony* memiliki performa yang jauh lebih baik dari algoritma lain seperti *Genetic Algorithm* (GA), *Evolutionary Programming* (EP), dan *Simulated Annealing* (SA) (Dorigo dan Gambardella, 1997:13). Pada kasus *Traveling Salesman Problem* dengan 75 kota, ACS hanya membutuhkan simulasi tur sebanyak 3.480 kali untuk menemukan jalur tur terbaik, sedangkan GA membutuhkan 80.000 kali simulasi tur untuk menemukan jalur tur terbaik dan algoritma lain seperti EP dan SA bahkan membutuhkan jumlah simulasi tur yang jauh lebih banyak lagi.

Google telah membuat API untuk mengintegrasikan fungsi dari layanan Google Maps ke aplikasi web yang lain dan memanfaatkan layanan peta digital dari Google. Menentukan letak suatu titik pada Google Maps dengan memanfaatkan API yang dapat menerima input koordinat dalam format derajat lintang (*latitude*) dan derajat bujur (*longitude*). Pencarian jalur juga salah satu pemanfaatan Google Maps API, dengan menentukan daerah awal kemudian menunjuk satu lokasi lain yang berbeda, dapat ditunjukkan jalur yang dapat dilalui diantara lokasi tersebut (Svenneberg, 2010:4).

Ant Colony

Metode Ant Colony diadopsi dari perilaku koloni semut, secara alamiah koloni semut mampu menemukan rute terpendek dalam perjalanan dari sarang ke sumber makanan berdasarkan jejak semut-semut yang lain pada lintasan yang telah dilalui. Semakin sering lintasan dilalui maka semakin banyak pula semut lain yang akan melewati lintasan tersebut dan sebaliknya lintasan yang dilalui semut dalam jumlah sedikit, semakin lama semakin berkurang kepadatan semut yang melewatinya, atau bahkan tidak dilewati sama sekali.

Pada saat semut melewati lintasan, semut meninggalkan sejumlah informasi, berupa zat kimia yang disebut *pheromone*. Melalui *pheromone* inilah terjadi komunikasi tidak langsung dan juga pertukaran informasi antar semut selagi membangun suatu solusi. Proses peninggalan *pheromone* ini dikenal sebagai *stigmetry*, yaitu sebuah proses memodifikasi lingkungan yang tidak hanya bertujuan untuk mengingat jalan pulang ke sarang, tetapi juga memungkinkan para semut berkomunikasi dengan sesamanya (Dorigo dan Stutzle, 2004:1).

Agar semut mendapatkan jalur optimal dalam perjalanannya, diperlukan beberapa proses:

1. Pada awalnya, semut berkeliling secara acak, hingga menemukan makanan.
2. Ketika menemukan makanan mereka kembali ke sarangnya sambil memberikan tanda dengan jejak *pheromone*.
3. Jika semut-semut lain menemukan jalur tersebut, maka mereka tidak akan bepergian dengan acak lagi, melainkan akan mengikuti jejak tersebut.
4. Jika pada akhirnya mereka pun menemukan makanan, mereka kembali menguatkan jejaknya.
5. *Pheromone* yang berkonsentrasi tinggi pada akhirnya akan menarik semut-semut lain untuk berpindah jalur, menuju jalur paling optimal, sedangkan jalur lainnya akan ditinggalkan.

Algoritma Ant System

Sama halnya dengan cara kerja semut dalam mencari jalur yang optimal, untuk mencari jalur terpendek dalam penyelesaian masalah *Traveling Salesman Problem* (TSP) diperlukan beberapa langkah untuk mendapatkan jalur yang optimal, antara lain:

1. Menentukan *pheromone* awal masing-masing semut. Inisialisasi *pheromone* awal (τ_0) pada algoritma semut ditentukan melalui persamaan 1.1 berikut ini (Dorigo dan Stutzle, 2004:70):

$$\tau_0 = \frac{m}{Cgreedy} \text{.....(Persamaan 1.1)}$$

keterangan:
m = jumlah semut
$Cgreedy$ = panjang tur yang dihasilkan melalui algoritma *Greedy*
τ_0 = *pheromone* awal

2. Nilai visibilitas η_{ij} dihitung berdasarkan persamaan 1.2 berikut ini (Dorigo dan Stutzle, 2004:67):

$$\eta_{ij} = \frac{1}{dij} \text{.....(Persamaan 1.2)}$$

keterangan
η_{ij} = invers jarak dari kota i ke kota j
d_{ij} = jarak dari kota i ke kota j

3. Pada aturan pembangunan solusi, semut akan memilih kota berdasarkan perbandingan nilai probabilitas yang dihitung dan nilai acak yang dibangkitkan. Aturan menghitung nilai probabilitas dinyatakan dalam persamaan 1.3 sebagai berikut (Dorigo dan Stutzle, 2004:70):

$$P_{ij,k}(t) = \begin{cases} \dfrac{\left[\tau_{ij}(t)\right]^{\alpha}.\left[\eta_{ij}\right]^{\beta}}{\sum\left[\tau_{ij}(t)\right]^{\alpha}.\left[\eta_{ij}\right]^{\beta}} & \text{Jika } j \in J_{ik} \\ 0, \text{ jika } j \notin J_{ik} \end{cases}$$

.....(Persamaan 1.3)

Keterangan :
$P_{ij,k}(t)$ = peluang semut ke-k untuk mengunjungi kota j dari kota i pada siklus ke-t
$\tau_{ij}(t)$ = inisialisasi *pheromone* antara kota i dan kota j pada siklus ke-t
η_{ij} = invers jarak dari kota i ke kota j
J_{ik} = kumpulan kota yang akan dikunjungi oleh semut yang berada pada kota i
α = parameter yang mengontrol *pheromone* ($\alpha \geq 0$)
β = parameter yang mengontrol jarak ($\beta \geq 0$)

4. Nilai acak ($0 < q_0 \leq 1$) yang dibangkitkan pada algoritma semut ini untuk membandingkan dengan nilai probabilitas pada suatu kota j apakah q_0

< $P_{ij,k}(t)$, jika nilai acak yang dibangkitkan lebih kecil maka sesegera mungkin semut akan mengunjungi kota *j* tersebut, jika nilai acak yang dibangkitkan lebih besar maka dibandingkan dengan nilai probabilitas pada kota selanjutnya, sampai semua kota dikunjungi oleh semut.

5. Setelah semua kota dikunjungi oleh semut maka secara lokal (pada semut itu sendiri) *pheromone* akan diperbaharui dengan menghitung perbaikan jejak *pheromone* atau perubahan harga *pheromone* antar kota. Persamaannya sebagai berikut (Dorigo dan Stutzle, 2004:72):

$\Delta \tau_{ij,k} = \frac{1}{Lk}$, jika $(i,j) \in T_k(t)$ .....(*Persamaan 1.4*)

Keterangan:

$\Delta \tau_{ij,k}$ = jumlah *pheromone* yang ditambahkan oleh semut ke *k*

$L_k(t)$ = panjang rute keseluruhan

$T_k(t)$ = rute keseluruhan

6. Setelah semua semut menyelesaikan turnya, *pheromone* akan diperbaharui secara menyeluruh (global) pada semut yang memiliki panjang rute terpendek. Persamaan perubahan *pheromone* global, sebagai berikut (Dorigo dan Stutzle, 2004:72):

$\tau_{ij\ (baru)} \leftarrow (1 - \rho)\ \tau_{ij} + \Delta\ \tau_{ij,k}(t)$ .....(*Persamaan 1.5*)

Keterangan:

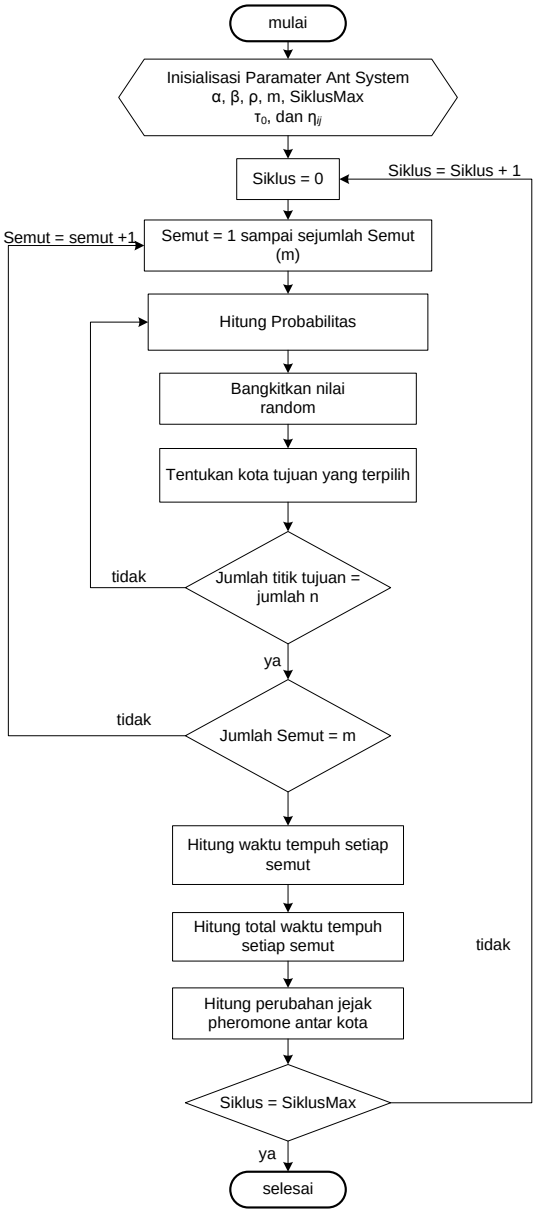
$\tau_{ij\ (baru)}$ = konsentrasi *pheromone* yang baru

τ_{ij}= inisialisasi *pheromone* antar kota i dan kota j

ρ = parameter laju penguapan *pheromone* ($0 < \rho \leq 1$)

$\Delta\tau_{ij,k}(t)$ = jumlah *pheromone* yang ditambahkan oleh semut *k*

7. Nilai *pheromone* yang baru digunakan pada siklus selanjutnya. Siklus selanjutnya dimulai lagi pada langkah nomor 3.



Gambar 1. Diagram Alir Algoritma Ant System

METODE

Metode pengembangan sistem yang digunakan dalam penelitian ini yaitu dengan menggunakan metode pengembangan perangkat lunak *Waterfall*. Berikut penjelasan mengenai tahapan-tahapannya:

Analisis sistem

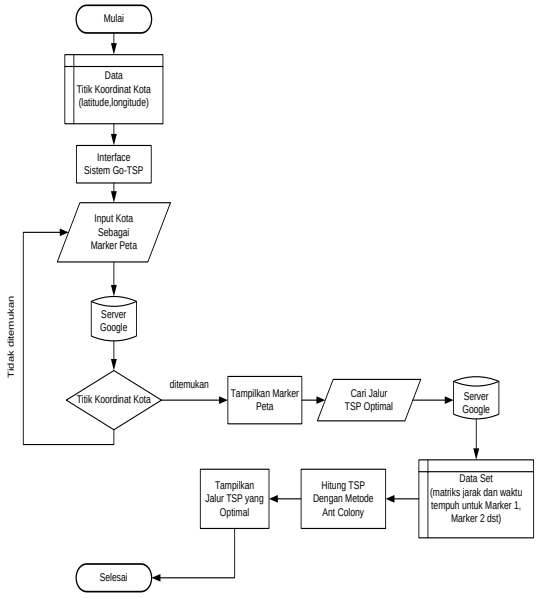
Tahap analisis sistem dilakukan dengan beberapa macam identifikasi yaitu dengan

melakukan pengamatan pada layanan berbasis Google Maps seperti RouteXL dan Gebweb. Flowchart atau Diagram Alir digunakan sebagai visualisasi alur algoritma yang digunakan dalam penelitian ini.

Desain

Alur Proses Sistem

Proses berjalannya sistem terdiri atas permintaan *Marker*, permintaan data jarak dan waktu tempuh pada Google Maps, dan perhitungan Algoritma *Ant System*.



Gambar 2. Rancangan Alur Proses Sistem

Pengkodean

Dalam pengkodean penulis menggunakan metode pemrograman berorientasi objek (OOP), yang berupa penggunaan Google Maps API dan menggunakan bahasa pemrograman JavaScript.

Testing

Langkah-langkah dari strategi testing sebagai berikut:

a. Parameter Algoritma Ant System

Pada Tahap ini parameter-parameter yang berpengaruh dalam proses perhitungan dalam pencarian rute perjalanan pola *Traveling Salesman Problem* dengan algoritma Ant System dicari kombinasi terbaik dengan mengubah nilai parameter, serta mengamati hasil optimasi dari setiap kombinasi. Setelah didapatkan hasil optimasi, proses selanjutnya yaitu membandingkan hasil dengan metode pencarian TSP dengan algoritma Greedy.

b. Perbandingan Hasil Optimasi

Hasil perbandingan antara panjang bobot nilai yang dihasilkan oleh algoritma ant system, algoritma ant system dengan *Local Search*, dan algoritma greedy, serta kecepatan perhitungan yang dilakukan pada ketiga algoritma tersebut.

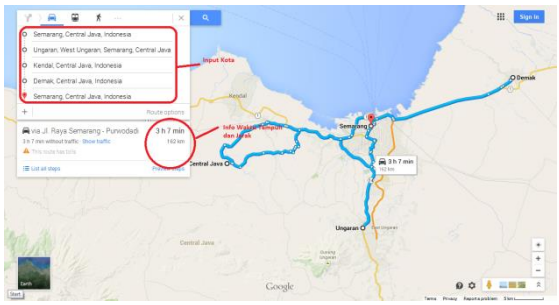
HASIL DAN PEMBAHASAN

Sistem yang dikembangkan dengan memanfaatkan Google Maps API untuk perencanaan dengan pola TSP ini selanjutnya disebut dengan Sistem Ant-TSP. Berikut adalah gambar tampilan Ant-TSP:

Gambar 3. Tampilan Ant-TSP

Pengujian Perencanaan Perjalanan pada Google Maps

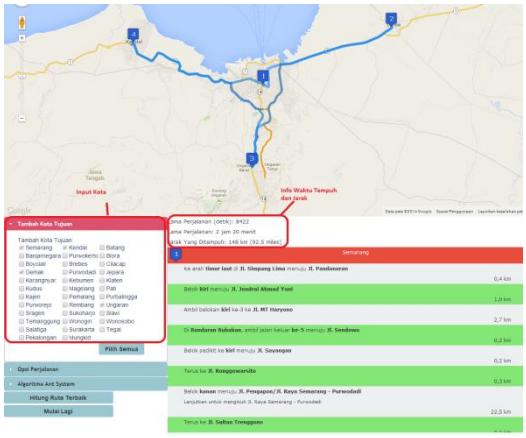
Memulai sebuah perjalanan dengan melewati banyak kota dengan pola TSP pada Google Maps dimulai dengan mengisi kota awal pada *Tab Menu* atas kiri yang ditunjukkan pada Gambar 4, kemudian dilanjutkan pada *Tab Menu* dibawahnya untuk melewati kota selanjutnya. Pada ilustrasi TSP ini, kota yang akan dilewati adalah Semarang – Ungaran – Kendal – Demak – Semarang. Maka perjalanan yang disarankan pada Google Maps hanya sesuai urutan awal masukan kota. Total jarak 162 Km dengan waktu tempuh 3 jam 7 menit.



Gambar 4. Hasil Perencanaan Perjalanan Google Maps

Pengujian Perencanaan Perjalanan pada Ant-TSP

Sistem Ant-TSP memanfaatkan Google Maps API yang telah disisipkan Metode *Ant Colony* untuk mengembangkan sebuah sistem perencanaan perjalanan pola TSP dengan hasil yang optimal, berdasarkan jumlah bobot waktu tempuh untuk melewati semua kota. Dengan ilustrasi TSP yang sama pada perencanaan perjalanan dengan Google Maps, kota yang dilewati adalah Semarang – Ungaran – Kendal – Demak – Semarang. Hasil yang diperoleh adalah dengan melewati Semarang – Demak – Ungaran – Kendal – Semarang dengan jarak keseluruhan sepanjang 148 km dengan waktu tempuh 2 jam dan 20 menit. Hasil system Ant-TSP ditunjukkan pada gambar 5 sebagai berikut:

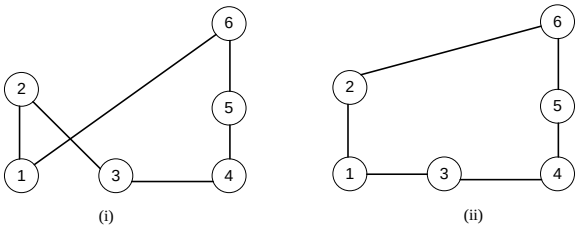


Gambar 5. Hasil Perencanaan Perjalanan Ant-TSP

Pada pengujian perencanaan perjalanan Ant-TSP dengan inputan kota kurang dari 10, algoritma *Ant System* mampu memberikan hasil yang optimal. Sedangkan untuk inputan kota lebih dari 10, algoritma *Ant System* tidak mampu memberikan hasil yang optimal. Oleh karena itu untuk menambah kinerja algoritma *Ant System* ditambahkan algoritma *Local Search 2opt*, untuk membuktikan hasil dari algoritma *Ant System* dengan penambahan algoritma *Local Search 2-opt* maka dilakukan uji coba. Perbandingan hasil perjalanan yang dihasilkan kedua algoritma *Ant System* sebelum dan setelah ditambah dengan algoritma *Local Search 2-opt* juga dibandingkan dengan algoritma *Greedy*.

Local Search 2-opt

Local Search digunakan untuk mendapatkan bobot minimum pada *Traveling Salesman Problem*. Pada penelitian ini local search yang digunakan merupakan adaptasi *2-opt*. Pada *2-opt* ini, dilakukan *2-change,* yakni mengganti 2 buah sisi yang lama dengan 2 buah sisi yang baru. Lihat ilustrasi berikut: (ilustrasi diambil dari Dorigo dan Stutzle :32).



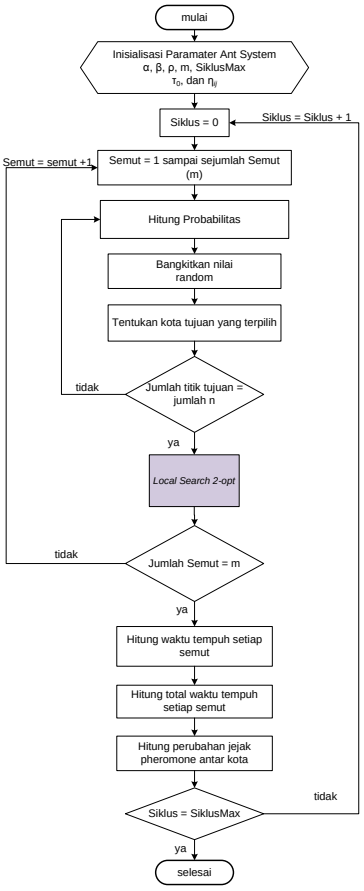
Gambar 6. Jalur TSP sebelum dan sesudah *Local Search 2-opt*

Pada gambar 6 (i) adalah jalur TSP lama sebelum perubahan, sedangkan gambar 2.21 (ii) adalah jalur TSP baru setelah dilakukan *2-change*. Misal jalur TSP {1,6,5,4,3,2} seperti yang tampak pada gambar (i). Pada jalur TSP tersebut kemudian dilakukan penghapusan 2 buah sisi, misalkan sisi {1,6} dan sisi {2,3), yang akan menghasilkan dua buah *path* yang terpisah yakni{1,2} dan {3,4,5,6}. Kedua buah *path* tersebut kemudian disambungkan kembali (untuk membentuk jalur TSP baru) dengan menggunakan sisi yang berbeda, yakni sisi {1,3} dan sisi{2,3}. Sehingga jalur TSP baru yang dihasilkan adalah {1,3,4,5,6,2} gambar (ii).

Apabila bobot jalur TSP tersebut lebih besar dari bobot jalur tsp sebelum dilakukan proses *2-change*, maka jalur TSP yang baru tersebut akan diabaikan, dan proses *2-change* dilakukan kembali untuk sisi yang lainnya. Proses ini dihentikan apabila tidak mungkin lagi didapatkan jalur TSP yang lebih baik.

Algoritma Ant System dengan Local Search 2-opt

Gambar berikut ini adalah modifikasi dari gambar 1. Bagian yang diarsir menunjukkan letak algoritma *Local Search 2-opt* dalam algoritma *Ant System.*



Gambar 7. Flowchart Algoritma *Ant System* dengan *Local Search 2-opt*

Parameter-parameter yang digunakan algoritma *Ant System* dalam uji sistem sebagai berikut:

Alfa (α) = 1.0
Beta (β) = 2.0
Rho (ρ) = 0.5
Banyak semut (*m*) = banyak kota (*n*)
(Dorigo dan Stutzle, 2004:71).

Algoritma *Greedy*

Jumlah bobot untuk *Asymmetric Traveling Salesman Problem* (ATSP) yang dibangun oleh algoritma *Greedy* pada persamaan berikut ini:

Jika n = banyak kota, himpunan kota-kota yang akan dikunjungi adalah $V(Kn)$ = {1, 2, …, n}, dimana 1 sebagai kota awal dan kota akhir perjalanan. Untuk mengunjungi kota selanjutnya dicari bobot min{i, j}, dengan bobot(i, i+1) = in untuk i = 1, 2, …, n-1.

Tur yang dibangun oleh algoritma *Greedy*, misal himpunan T =(1, 2, …, n, 1). Jumlah bobot T =

$$\sum_{i=1}^{n-1} in + \text{bobot}(n, 1)$$

……(*Persamaan* 1.6)

Pseudo-code Local Search 2-opt

1. Matriks D = (d_{ij}) yaitu matriks data waktu tempuh antar kota.
2. T adalah jalur TSP yang dihasilkan oleh algoritma *Ant System*. Himpunan kota pada T adalah, x_1, x_2,…, x_n dan $x_{n+1} = x_1$. Bobot T disimbolkan dengan $v(T)$.

$$v(T) = \sum_{i=1}^{n} d_{xi,xi+1}$$

……(Persamaan 3.2)

3. Nilai Δ = 0
   For i = 1,…,n – 2
   If i = 1 **Then** n' = n – 1, **Else** n' = n;
   For j = i + 2,…,n'
   Compute $\Delta_{ij} = d_{xi,xj} + d_{xi+1,xj+1} - d_{xi,xi+1} - d_{xj,xj+1}$
   If $\Delta_{ij} < \Delta$, **Let** $\Delta := \Delta_{ij}$, $i^* = i$, $j^* = j$;
   EndFor
   EndFor
   If Δ = 0, **go to** 5
   Let $i = i^*$, dan $j = j^*$
4. Mengubah tur dengan menghilangkan sisi diantara simpul ke i^{th} dan i+1th dan pada simpul j^{th} dan $[j+1]^{th}$, dan sisipkan sisi diantara simpul ke i^{th} dan j^{th} dan pada simpul$[\ i+1]^{th}$ dan$[\ j+1]^{th}$.

Ganti posisi dari elemen $i+l$ dan $j - l + 1$ pada array x,

For $l = 1,\ldots,[\, j - i \,] / 2;$

$v(T) := v(T) + \Delta$

5. Nilai bobot $v(T)$ baru ditemukan, dan himpunan kota-kota baru yang bersesuaian $x_1,\ldots,x_n$.

Hasil Pengujian

Tabel 1. Jumlah Bobot Jarak dan Waktu Tempuh untuk 6 Kota

Percobaan	Ant System		Ant System *Local Search*		Greedy	
	Jarak (m)	Waktu Tempuh (s)	Jarak (m)	Waktu Tempuh (s)	Jarak (m)	Waktu Tempuh (s)
1	676673	40690	673369	39643	698916	41315
2	699569	40611	673369	39643	698916	41315
3	699569	40611	673369	39643	698916	41315
4	699569	40611	673369	39643	698916	41315
5	699569	40611	673369	39643	698916	41315
6	699569	40611	673369	39643	698916	41315
7	699569	40611	673369	39643	698916	41315
8	699569	40611	673369	39643	698916	41315
9	699569	40611	673369	39643	698916	41315
10	699569	40611	673369	39643	698916	41315
Terbaik	676673	40611	673369	39643	698916	41315

Tabel 2. Waktu Komputasi 6 Kota untuk Masing-Masing Algoritma

Percobaan	*Execution Time* (detik)		
	Ant System	Ant System *Local Search*	Greedy
1	0,051	0,066	0,00099
2	0,072	0,068	0,00099
3	0,055	0,066	0,00102
4	0,051	0,066	0,00102
5	0,054	0,076	0,00102
6	0,051	0,075	0,00102
7	0,05	0,062	0,00103
8	0,047	0,062	0,00102
9	0,048	0,078	0,00102
10	0,05	0,062	0,00102
Rata-rata	0,053	0,068	0,001

Tabel 3. Jumlah Bobot Jarak dan Waktu Tempuh untuk 15 Kota

Percobaan	Ant System		Ant System *Local*		Greedy	
	Jarak (m)	Waktu Tempuh (s)	Jarak (m)	Waktu Tempuh (s)	Jarak (m)	Waktu Tempuh (s)
1	1264892	73944	1008000	60195	1161515	68558
2	1291844	74045	1008000	60195	1161515	68558
3	1359597	78381	1008000	60195	1161515	68558
4	1291119	74299	1008000	60195	1161515	68558
5	1213734	72161	1008000	60195	1161515	68558
6	1225123	73425	1008000	60195	1161515	68558
7	1302256	74712	1008000	60195	1161515	68558
8	1252881	73160	1008000	60195	1161515	68558
9	1293766	74495	1008000	60195	1161515	68558
10	1285188	74055	1008000	60195	1161515	68558
Terbaik	1213734	72161	1008000	60195	1161515	68558

Tabel 4. Waktu Komputasi 15 Kota untuk Masing-Masing Algoritma

Percobaan	*Execution Time* (detik)		
	Ant System	Ant System *Local Search*	Greedy
1	0,145	0,281	0,00099
2	0,187	0,265	0,00102
3	0,23	0,297	0,00103
4	0,13	0,265	0,00099
5	0,131	0,265	0,00299
6	0,107	0,499	0,00102
7	0,135	0,312	0,00102
8	0,159	0,280	0,00099
9	0,122	0,250	0,00102
10	0,15	0,265	0,00102
Rata-rata	0,150	0,298	0,001

Tabel 5. Jumlah Bobot Jarak dan Waktu Tempuh untuk 35 Kota

Percobaan	Ant System		Ant System *Local*		Ant System	
	Jarak (m)	Waktu Tempuh (s)	Jarak (m)	Waktu Tempuh (s)	Jarak (m)	Waktu Tempuh (s)
1	2548021	147682	1324872	80353	1636366	96072
2	2629901	150218	1326016	80445	1636366	96072
3	2677074	157453	1324872	80353	1636366	96072
4	2419806	141715	1324872	80353	1636366	96072
5	2528445	146174	1324872	80353	1636366	96072
6	2557414	152062	1324872	80353	1636366	96072
7	2684858	155124	1324872	80353	1636366	96072
8	2563855	150338	1324872	80353	1636366	96072
9	2558863	153088	1324872	80353	1636366	96072
10	2583328	151866	1324872	80353	1636366	96072
Terbaik	2419806	141715	1324872	80353	1636366	96072

Tabel 6. Waktu Komputasi 35 Kota untuk Masing-Masing Algoritma

Percobaan	*Execution Time* (detik)		
	Ant System	Ant System *Local Search*	Greedy
1	0,16983	0,4181	0,00125
2	0,18192	0,4470	0,00099
3	0,16508	0,3038	0,00115
4	0,19125	0,3013	0,00117
5	0,19648	0,2924	0,0012
6	0,18234	0,3005	0,00099
7	0,17069	0,3033	0,00102
8	0,19235	0,2914	0,0012
9	0,17254	0,3028	0,00105
10	0,18449	0,3097	0,00099
Rata-rata	0,181	0,327	0,001

Dari hasil pengujian diperoleh beberapa kelebihan dan kekurangan algoritma *Ant System*, *Ant System Local Search*, dan *Greedy* dalam menentukan jalur optimal *Traveling Salesman Problem*.

Kelebihan :

- Algoritma *Ant System* menghasilkan variasi jalur optimal yang lebih banyak

daripada algoritma *Ant System Local Search* dan *Greedy*.

- Algoritma *Ant System Local Search* menghasilkan bobot jarak dan bobot waktu tempuh paling minimum dibandingkan dengan algoritma *Ant System* dan *Greedy*.
- Algoritma *Greedy* memilki waktu komputasi paling cepat dibandingkan dengan algoritma *Ant System* dan *Ant System Local Search*.

Kekurangan

- Algoritma *Ant System* menghasilkan bobot jarak dan bobot waktu tempuh paling besar dan jalur TSP yang dihasilkan tidak optimal dibandingkan algoritma *Ant System Local Search* dan *Greedy*.
- Algoritma *Ant System Local Search* membutuhkan waktu komputasi yang lama jika dibandingkan dengan algoritma *Ant System* dan algoritma *Greedy*.
- Algoritma *Greedy* dalam proses memilih kota selanjutnya yang akan dikunjungi hanya berdasar kota yang terdekat dan secara keseluruhan menghasilkan jalur TSP yang tidak optimal.

SIMPULAN

Sistem untuk menemukan jalur *Traveling Salesman Problem* dengan menggunakan metode *Ant Colony* telah diimplementasikan dan dilakukan uji coba untuk membandingkan hasil yang didapatkan dengan algoritma yang dibangun yaitu algoritma *Ant System Local Search*. Simpulan yang diperoleh sebagai berikut:

Berdasarkan perhitungan dengan membandingkan bobot jarak dan waktu tempuh dari ketiga algoritma, algoritma *Ant System Local Search* mempunyai bobot jarak dan waktu tempuh yang paling kecil dari dua algoritma yang lain, serta jalur *Traveling Salesman Problem* yang dibangun optimal. Pada hasil waktu komputasi dalam perhitungan jalur *Traveling Salesman Problem*, algoritma *Greedy* membutuhkan waktu komputasi yang paling sedikit dibandingkan dengan algoritma *Ant System* dan *Ant System Local Search*.

Proses perhitungan untuk menentukan jalur *Traveling Salesman Problem*, sebagian besar digunakan untuk meminta data ke Server Google, hal ini ditunjukkan dengan presentase yang besar pada bagian permintaan data dibandingkan dengan bagian yang lain, yaitu bagian perhitungan algoritma dan pembangunan solusi.

Berdasarkan simpulan yang telah dikemukakan, dapat diajukan saran dalam pengembangan sistem lebih lanjut sebagai berikut:

Agar lebih cepat proses perhitungan dalam penentuan jalur *Traveling Salesman Problem*, data jarak dan waktu tempuh yang didapat dari Server Google disimpan pada *database* sistem.

DAFTAR PUSTAKA

Dorigo, Marco dan Luca M. Gambardella. 1997. Ant Colonies for the traveling salesman problem. BioSystem. InPress.

Dorigo, Marco dan Thomas Stutzle. 2004. *Ant Colony Optimization*. Massachusetts: MIT Press.

Dorigo, Marco et al. 1996. The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on System, Man, and Cybernetics*. Vol. 26: pp. 1-13.

Gutin, Gregory et al. 2002. Traveling salesman should not be greedy: domination analysis of greedy-type heuristic for the TSP. *ELSEVIER*. Discrete Applied Mathematics: 81-86.

Hansen, Pierre dan Nenad Mladenovic. 2006. First vs. best improvement: Empirical Study. *ELSEVIER*. Discrete Applied Mathematics: 802-817.

Mulia, Dedi. 2011. *Aplikasi Algoritma Ant System (AS) Dalam Kasus Traveling Salesman Problem (TSP)*. Jakarta: UIN Jakarta.

Munir, Rinaldi. 2007. *Matematika Diskrit*. Bandung : Informatika ITB.

Siang, Jong Jek. 2009. *Matematika Diskrit dan Aplikasinya pada Ilmu Komputer*. Yogyakarta: ANDI OFFSET.

_____. 2011. *Riset Operasi dalam Pendekatan Algoritmis*. Yogyakarta: ANDI OFFSET.

Svennerberg, Gabriel. 2010. *Beginning Google Maps API 3*. New York: Apress.