PENERAPAN ALGORITMA ANT COLONY OPTIMIZATION MENENTUKAN NILAI OPTIMAL DALAM MEMILIH OBJEK WISATA BERBASIS ANDROID

Euis Nurlaelasari
Program Teknik Informatika
STMIK Kharisma Karawang
Email: euisnurlaela333@gmail.com

Supriyadi
STMIK Kharisma Karawang
Email: supriyadistmikkharisma@yahoo.com

U. Tresna Lenggana
STMIK Kharisma Karawang
Email: trena.mobile.pro@gmail.com

ABSTRAK

Pencarian nilai yang optimal adalah permasalahan yang dapat dijumpai pada kehidupan sehari-hari. yaitu seperti, menentukan rute terpendek, menentukan jumlah optimal untuk persediaan hasil produksi dan lain-lain. Pencarian nilai optimal dapat digunakan untuk memperoleh nilai tertinggi dan terendah dari suatu permasalahan. Salah satu permasalahan yang populer dan dapat dipecahkan dengan algoritme optimasi adalah *Traveling Salesman Problem* (TSP) untuk menentukan rute terdekat dengan menggunakan algoritme *Ant Colony Optimization*. Namun pada kenyataannya, jarak bukanlah satu-satunya tolak ukur yang dapat diperhitungkan saat melakukan perjalanan. Oleh karena itu, penelitian ini bertujuan untuk melengkapi kekurangan pada penelitian sebelumnya dengan menambahkan variabel lain selain jarak. Algoritma *Ant Colony Optimization* digunakan untuk menentukan objek wisata dengan menghitung variabel biaya pada sebuah jarak. Sehingga dapat menghasilkan sebuah biaya transportasi terendah. Hasil dari penelitian ini berupa rancangan sistem dengan menggunakan UML (*Use case, Class, Sequence, Activity Diagram)* dan rancangan aplikasi pemilihan objek Wisata Karawang berbasis android dengan menerapkan Algoritme *Ant Colony Optimization*. Metode pengembangan sistem yang digunakan adalah S*ystem Development Life Cycle* (SDLC) *Waterfall* serta rancangan sistem berbasis *Object Oriented.*

Kata kunci: *ant colony optimization*, biaya, objek wisata, *traveling salesman problem*.

ABSTRACT

The search for an optimal value is a problem that can be found in everyday life such as, determining the shortest route, determining the optimal amount of inventory of production and others. The search for the optimal value can be used to obtain the highest and lowest value of a problem one of the most popular and solvable problems with the optimization algorithm is the Traveling Salesman Problem (TSP) to determine the nearest route by using ant colony optimization algorithm. But in reality, distance is not the only measurable benchmark when traveling. Therefore, this study aims to supplement the deficiencies in previous research by adding other variables besides the distance Algorithm Ant Colony Optimization is used to determine the object of tourism by calculating the variable cost at a distance. So that can produce a transportation cost lowest The result of this research is in the form of system design using UML (Use case, Class, Sequence, Activity Diagram) and the design of Karawang-based Tourism object selection app by applying Ant Colony Optimization Algorithm. System development method used is System Development Life Cycle (SDLC) Waterfall and Object Oriented based system design.

Keywords: *ant colony optimization, cost, trourist attraction, traveling salesman problem.*

1. PENDAHULUAN

Pencarian nilai yang optimal adalah permasalahan yang dapat dijumpai pada kehidupan sehari-hari. Permasalahan tersebut seperti menentukan rute terpendek [1], menentukan jumlah optimal untuk persediaan hasil produksi [2], penentuan nilai risiko kredit tertinggi pada bank [3], dan nilai optimum

profit, *weight* atau *density* pada *knapsack problem* [4]. Pencarian nilai optimal dapat digunakan untuk memperoleh nilai tertinggi dan terendah dari suatu permasalahan. Sebagai contoh, pada kasus pencarian *benefit* maka yang akan diambil adalah nilai tertinggi, sedangkan untuk permasalahan risiko kredit dan pencarian rute terdekat maka yang akan diambil adalah nilai terkecil.

Pencarian rute terdekat sangat populer dan bisa dijumpai pada *Traveling Salesman Problem* (TSP) [5]. Pada TSP, variabel yang dihitung adalah jarak antara titik atau objek yang akan dikunjungi sebagai bobot dari *graph*. Pada implementasinya pencarian rute terpendek bisa menggunakan algoritme optimasi seperti algoritme genetika [6], algoritme *Tabu Search* [7], algoritme *djikstra* [8] atau algoritme *Ant Colony Optimization* [1]. Dalam kenyataannya ketika melakukan sebuah perjalanan, jarak bukanlah satu-satunya tolak ukur yang dapat dihitung. Alat transportasi yang digunakan juga dapat diperhatikan yang nantinya akan menjadi bahan perhitungan dalam hal penggunaan biaya perjalanan. Namun belum ada sistem penunjang keputusan untuk menentukan nilai optimum berdasarkan variabel jarak dan biaya.

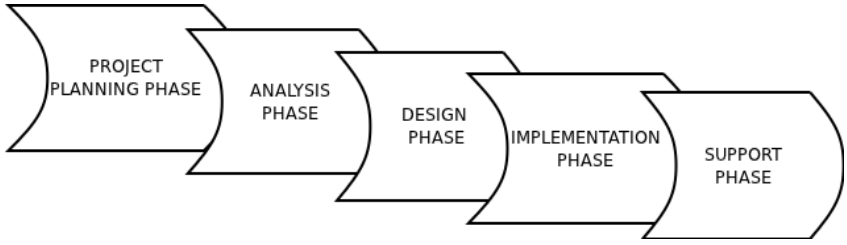
Oleh karena itu pada penelitian ini diharapkan memberikan solusi untuk penentuan nilai optimal dengan jarak dan biaya sebagai variabel yang dihitung menggunakan algoritme *Ant Colony Optimization* (ACO).

Pembangunan sistem akan menggunakan metode *System Development Life Cycle* (SDLC) *Waterfall* [9]. Berdasarkan pertimbangan diatas, maka penelitian ini bertujuan untuk melengkapi kekurangan pada penelitian sebelumnya dengan menambahkan variabel biaya pada jarak menggunakan Algoritme *Ant Colony Optimization* untuk menentukan nilai optimal dalam memilih Objek Wisata di Karawang.

2. METODOLOGI PENELITIAN

2.1 Metodologi *System Development Life Cycle* (SDLC) *Waterfall*

Metode ini merupakan metode yang sering digunakan oleh penganalisis sistem pada umumnya. Metode *waterfall* adalah suatu metodologi pengembangan perangkat lunak yang mengusulkan pendekatan kepada perangkat lunak semantik dan sekuensial yang mulai pada *Project Planning phase, Analysis phase, Design phase, Implementation phase, suport phase* [9]. Namun penulis membatasi proses hanya sampai *implementation phase*. seperti pada gambar 1.



Gambar 1. Tahapan Metode *Waterfall*

2.1.1 Project Planning Phase

Tahap perencanaan adalah tahap awal penelitian untuk memahami algoritme *Ant Colony Optimization* yang diterapkan untuk menghitung biaya transportasi terendah. Penjelasan aktivitas yang yang dilakukan pada tahap ini dapat dilihat pada Tabel 1.

Tabel 1. Rincian *project planning phase*

No	*Tahapan*	*Deskripsi*
1	Identifikasi Masalah	Belum adanya pencarian optimasi yang menghitung jarak dan biaya
2	Pengumpulan Data	Melakukan studi literstur melalui *ebook* dan buku mengenai algoritme ACO dan data mengenai objek wisata dan tarif angkutan umum Kabupaten Karawang
3	Menganalisis Teori	Melakukan analisis pada algoritme ACO dan metode pengembangan SDLC *Waterfall*
4	Pembuatan Jadwal	Membuat rencana pengembangan dan target pembuatan aplikasi
5	Mencari Solusi	Menentukan varibel yang akan dihitung yaitu jarak dan biaya menggunakan perhitungan algoritme ACO, dengan mencari nilai optimal terendah
6	Mengidentifikasi Kebutuhan	Menentukan *tools* yang dibutuhkan untuk pembangunan sistem.

2.1.2 Analysis Phase

a. 2.1.2.1 Analisis Teori

Pada tahap ini dilakukan analisis terhadap algoritme *ant colony optimization*. Ada beberapa varian ACO yang berkembang dari tahun 1991 hingga 2001, yakni *Ant System (AS). Elitist AS, Ant-Q, Ant Colony System, MAX-MIN AS, Rank-based AS, ANTS, BWAS* dan *Hyper-cube AS* [10]. Namun pada penelitian ini akan menggunakan *Ant System*. Berikut adalah tahapan perhitungan dari AS, dapat dilihat pada Tabel 2:

Tabel 2. Analisis Teori

No	*Tahapan*	*Deskripsi*
1	Identifikasi d_{ij}	d_{ij} adalah jarak dari *node i* ke *node j*. dalam penelitian ini d_{ij} digantikan dengan c_{ij} yaitu biaya dari *node i* ke *j*
2	Inisialisasi Parameter awal bagi *pheromone* pada waktu ke-*t*	*t* = 0 {*t* adalah penghitung waktu} NC = 0 {NC adalah penghitung *cycle*} Untuk setiap busur *(i,j)*, tentukan nilai awal untuk intensitas jejak *pheromone* , $\tau_{ij}(t) = 0$.
3	Menentukan Jumlah Semut	Letakkan *m* semut pada *n nodes* (kota). *k* = semut ke-*k*, *m* = jumlah semut. for *k* = 1 to *m* do Letakkan kota awal untuk semut ke-*k* di dalam $tabu_k(s)$ end
4	Membuat *tabulist*	Setiap semut membuat tabulist masing-masing. for *k* = 1 to *m* do pilih kota *j* sebagai kota berikutnya yang akan dikunjungi, dengan probabilitas $p_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum[\tau_{ij}(t)]^{\alpha}.[\eta_{ij}]^{\beta}}$, dimana $\eta_{ij} = \frac{1}{d_{ij}}$. {pada waktu *t*, semut ke-*k* berada pada kota $i = tabu_k$} Pindahkan semut ke-*k* ke kota *j* Masukkan kota *j* ke $tabu_k(s)$ end
5	Hitung probabilitas dan *pheromone* pada *tabulist*	Menghitung probabilitas pada *tabulist* dengan $L_k = \sum P_{ij}^{k}(t)$ dan menghitung *pheromone* dengan $\Delta\tau_{ij}^{k} = \frac{Q}{L_k}$. Kemudian tentukan *tabulist* dengan nilai terbaik yaitu *tabulist*

		dengan nilai probabilitas tertinggi.
6	Lakukan Iterasi	Jika iterasi dilakukan maka lakukan tahap 7 kemudian lakukan perhitungan kembali ke tahap 1 dengan nilai *pheromone* yang telah diperbaharui. Jika iterasi tidak dilakukan maka lakukan tahap 8.
7	*Update pheromone*	Tahap menambahkan *pheromone* untuk perhitungan pada iterasi selanjutnya, dengan $\tau_{ij}(t+n) = \rho.\tau_{ij}(t) + \Delta\tau_{ij}$.
8	Cetak hasil	Menampilkan hasil terbaik berdasarkan hasil tahap ke-5.

b. Analisis Sistem
Pada tahap ini dilakukan analisis sistem pada pembuatan aplikasi dengan menggunakan *Object Oriented Analysis* (OOA). Hasil dari tahapan ini adalah tujuan pembangunan sistem terhadap masalah serta manfaat yang akan diperoleh. Tahapan dari analisis tersebut yaitu:

1. *System activities (use case diagram, deskripsi use case, aktor dan skenario)*
2. *Class Diagram*
3. *Object interaction (sequence diagram)*
4. *Object behaviour (activity diagram)*

2.1.3 Design Phase

Pada tahapan ini akan membuat desain sistem berdasarkan kebutuhan pengguna dan sistem, yang telah diidentifikasi dan ditentukan solusinya pada tahapan analisis sebelumnya. Desain yang yang akan dirancang adalah berdasarkan *Object Oriented Design* (OOD) yang terdiri dari:

a. Desain Proses
b. Desain Antarmuka

2.1.4 Implementation phase

Tahap implementasi adalah tahap penerapan desain yang telah dirancang kemudian diaplikasikan dalam bentuk program aplikasi. Kode program disusun menuruti disain program yang telah dirancang sebelumnya. Penulisan kode program tersebut dilakukan dengan teknik *Object Orientied Programming* (OOP). Dimana sebuah sistem didefinisikan menggunakan set yang lebih kecil dari objek yang saling terkait.

Tahap implementasi dilakukan dengan beberpa tahapan, yaitu:

a. Instalasi Sistem
Menjelaskan tahapan-tahapan dilakukannya proses instalasi aplikasi android.
b. Pelatihan Prosedural
Pelatihan tatacara penggunaan aplikasi yang telah diinstal didalam komputer.
c. Pengujian Terhadap Sistem
Pengujian sistem dilakukan dengan pengujian *black box* dan *white box*. Pengujian *white box* dilakukan pada fungsi utama aplikasi android pemilihan objek wisata. Sedangkan pengujian *black box* dilakukan untuk menguji fungsi-fungsi didalam program sehingga sesuai dan dapat berjalan dengan benar.

3. HASIL DAN PEMBAHASAN

Berdasarkan pada metode yang telah dipilih, pada tahap ini merupakan pembahasan dari fase-fase SDLC *Waterfall*. Tahap pertama *Project Planning Phase* berisi hasil dari aktivitas mengidentifikasi masalah, mengumpulkan data, menganalisis data dan mengidentifikasi kebutuhan penelitian. Tahap *Analysis Phase* yaitu berisi pembahasan mengenai analisis teori Algoritme ACO dan analisis sistem untuk kebutuhan sistem. Tahap selanjutnya adalah membuat desain sistem menggunakan *Activity Diagram* dan desain antarmuka sistem. Tahap terakhir adalah implementasi, yaitu dengan membuat aplikasi Pemilihan Objek Wisata berbasis Android.

3.1 Project Planning Phase

Pada tahap ini dihasilkan rincian dari setiap aktivitas yang dilakukan, mulai dari identifikasi masalah yaitu bagaimana membuat aplikasi pemilihan objek wisata berdasarkan biaya transportasi terendah, pengumpulan data yang didapatkan dari Dinas Perhubungan Karawang, menganalisis data biaya angkutan umum dan objek wisata Kabupaten Karawang serta menerapkan data tersebut pada perhitungan algoritme ACO, melakukan perhitungan pemilihan objek wisata dengan biaya terendah menggunakan algoritme ACO dan mendefinisikan kebutuhan penelitian yaitu laptop, *Smartphone, Linux Ubuntu* 12.04 32-bit, *LibreOffice Writer, LibreOffice Impress, Eclipse* 2.1.1, Android SDK 23. *Gaphor, Dia Diagram*.

3.2 Analysis Phase

Di dalam tahapan analisis ini meliputi analisis teori algoritme ACO, yaitu berupa data biaya transportasi objek pariwisata di Karawang dan hasil perhitungan ACO terhadap variabel biaya pada rute pariwisata di Karawang. Serta analisis sistem untuk kebutuhan pembangunan aplikasi.

3.2.1 Analisis Teori

Pada tahapan ini akan menjelaskan tentang menentukan suatu *tour* terbaik berdasakan biaya transportasi terendah pada objek wisata di Karawang, yang akan dipecahkan dengan menggunakan algoritme ACO. Perhitungan algoritme ACO dapat dibagi dalam beberapa tahapan seperti berikut:

a. Identifikasi d_{ij}
   D_{ij} adalah jarak dari *node i* ke *node j*. pada penelitian ini, variabel d_{ij} diganti dengan c_{ij} yaitu biaya dari *node i ke node j*. Berikut data c_{ij} dalam bentuk matriks dapat dilihat pada Tabel 3.

Tabel 3. Matriks biaya antar *node* objek wisata

	Node Objek Wisata													
	1	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***
1	0	9500	6000	11500	11500	18500	7000	7000	7000	8725	8725	8725	24950	24950
2	9500	0	3500	16000	10000	10000	16500	16500	16500	16500	16500	18225	34450	34450
3	6000	3500	0	10000	10000	16000	13000	13000	13000	14725	14725	14725	30950	30950
4	11500	16000	10000	0	19000	19000	25500	25500	25500	27225	27225	27225	32450	32450
5	11500	10000	10000	19000	0	19000	19500	19500	19500	19500	19500	21225	37450	37450
6	18500	10000	16000	19000	19000	0	25500	25500	25500	27225	27225	27225	32450	32450
7	7000	16500	13000	25500	19500	25500	0	7000	7000	8725	8725	8725	24950	24950
8	7000	16500	13000	25500	19500	25500	7000	0	3500	5225	5225	8725	21450	21450
9	7000	16500	13000	25500	19500	25500	7000	3500	0	3500	3500	8725	19725	19725
10	8725	16500	14725	27225	19500	27225	8725	5225	3500	0	3500	10450	19725	19725
11	8725	16500	14725	27225	19500	27225	8725	5225	3500	3500	0	10450	19725	19725
12	8725	18225	14725	27225	21225	27225	8725	8725	8725	10450	10450	0	10450	10450
13	24950	34450	30950	32450	37450	32450	24950	21450	19725	19725	19725	10450	0	21450
14	24950	34450	30950	32450	37450	32450	24950	21450	19275	19725	19725	10450	21450	0

Keterangan:
Node baris maupun *node* kolom dapat menjadi *node i* atau *j*

Keterangan *node* objek wisata:

1. Vihara Shia Jin Kupoh
2. Tugu Perjuangan
3. Rumah Djiaw Kie Siong
4. Pantai Pelangi
5. Pantai Samudra Baru
6. Pantai Tanjung Pakis
7. Kampung Budaya
8. Monumen Surotokunto
9. Bendungan Walahar
10. Situs Cikubang
11. Kebon Kembang & Situ Kamojing
12. Makam Syekh Quro & Makam Syekh Bentong
13. Pasir Putih
14. Makam Mantan Bupati Karawang

b. Inisialisasi Parameter Awal bagi *Pheromone* pada Waktu Ke-*t*

Parameter α, β, ρ mengikuti nilai yang telah ditetapkan sebelumnya, sedangkan *pheromone* yang dilambangkan dengan simbol (τ_{ij}) pada waktu ke-*t*, mula-mula diberi nilai awal 1 yang nantinya akan mengalami perubahan dengan *update pheromone* pada tahap ke-7. Berikut adalah nilai parameter:

$$\tau_{ij}(t) = 1$$
$$\alpha = 1$$
$$\beta = 2$$
$$\rho = 0.5$$

Parameter α, β dan τ_{ij}digunakan pada tahap 4, sedangkan ρ digunakan pada tahap ke-7.

c. Menentukan Jumlah Semut

Pada tahap ini adalah penentuan jumlah semut yang akan diletakan pada tiap *node* untuk melakukan t*our*. Jumlah semut disimbolkan dengan (*m*) dan diberi nilai 14 sesuai dengan jumlah *node* objek wisata.

d. Membuat *Tabulist*

Semut yang telah diletakkan pada masing-masing *node* mulai melakukan *tour* dan menghasilkan sebuah *path* atau rute. Jumlah *tabulist* akan sebanding dengan jumlah semut, maka *tabulist* akan berjumlah 14.

Langkah-langkah menentukan *tabulist*:

1. Letakan semut pada *node i*
   Misalkan pada *tabulist* ke-1 oleh semut ke-1. Mula-mula semut diletakan pada *node* ke-1, maka *node i* sama dengan *node* 1. Selanjutnya semut akan menuju node *j*.
2. Menentukan *node j*
   Node j adalah *node* antara node 1 sampai dengan 14 pada *node* objek wisata. Cara untuk menentukannya ialah berdasarkan nilai probabilitas terbesar, untuk mengetahuinya maka digunakan rumus:

$$\boldsymbol{invers\ biaya} = \eta_{ij} = \frac{1}{c_{ij}}, \text{dan}$$
$$\boldsymbol{probabilitas} = p_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum[\tau_{ij}(t)]^{\alpha}.[\eta_{ij}]^{\beta}},$$

$\sum[\tau_{ij}(t)]^{\alpha}.[\eta_{ij}]^{\beta}$adalah jumlah nilai $[\tau_{ij}(t)]^{\alpha}.[\eta_{ij}]^{\beta}$dari semua *node i* ke *j* dan *k* adalah semut ke-*k*.

Maka:

$$\eta_{12} = \frac{1}{c_{12}} = 0.1052631579$$
$$p_{12}^{1}(t) = \frac{[\tau_{12}(t)]^{\alpha}.[\eta_{12}]^{\beta}}{\sum[\tau_{12}(t)]^{\alpha}.[\eta_{12}]^{\beta}} = \frac{[1]^{1}.[0.1052631579]^{2}}{3.9646431268} = 0.0110803324$$

Maka dari itu diperlukan untuk menghitung nilai probabilitas antar *node*. Selanjutnya berdasakan perhitungan *tabulist* ke-1, semut ke-1 yang berada di *node* ke-1 akan bergerak ke *node* ke-3 berdasarkan probabilitas terbesar dengan nilai $p_{13}^{k} = 0.1666666667$.

3. Mencari *node* berikutnya
   Setelah didapatkan *node j* yaitu 3 maka selanjutnya mencari *node* selanjutnya dengan mengubah *node j* menjadi *node i*. Lalu setelah *node* 3 menjadi *node i* maka selanjutnya kembali menentukan *node j* dengan rumus yang sama namun dengan syarat bahwa *node j* selanjutnya bukan *node* yang sudah pernah dilalui (misal: *node* 1 atau 3). Perncarian *node-node* selanjutnya dilakukan dengan cara yang sama hingga semua *node* terlewati. *Node-node* yang telah dilewati tersebut kemudian disimpan kedalam *tabulist* sebagai sebuat *path*.

e. Hitung Probabilitas
Pada tahap ini dilakukan pencarian perolehan probabilitas terbesar dan *pheromone* terkecil pada *path* untuk dijadikan rute terbaik. Berikut tahapannya:

1. Menentukan probabilitas *path*
   Menghitung probabilitas sebuah *tabulist* dengan rumus:

$$L_k = \sum P_{ij}^{k}(t), \sum P_{ij}^{k}(t)$$

   yaitu jumlah seluruh probabilitas pada *tabulist*. Misalkan pada *tabulist* ke-1 dengan *path* 1 – 3 – 2 – 5 – 4 – 6 – 7 – 8 – 9 – 11 – 10 – 12 – 13 – 14, maka jumlahkan nilai probabilitas 1 ke 3, 3 ke 2, 2 ke 5 dan seterusnya. Berdasarkan perhitungan maka *tabulist* 1 memperoleh nilai probabilitas $L_1 = 0.124031803$. Selanjutnya menghitung *tabulist* berikutnya dengan cara yang sama.
2. Menghitung perolehan *pheromone* pada *path*
   Menghitung *pheromone* sementara dengan rumus:

$$\Delta\tau_{ij}^{k} = \frac{Q}{L_k},$$

   Q adalah iterasi. Contoh dengan menghitung tabulist 1, yaitu:

$$\Delta\tau_{ij}^{k} = \frac{1}{0.124031803} = 8.0624483061.$$

   Kemudian hitung *pheromone* sementara pada *tabulist* lainnya dengan cara yang sama.
3. Menghitung *pheromone* global

   Pheromone global dihitung dengan menggunakan rumus $\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$, yaitu menjumlahkan *pheromone* sementara pada semua *tabulist.* Maka akan dihasilkan *pheromone* global pada perhitungan ini sebesar 134.9274224269.
4. Menghitung perolehan *pheromone* pada tiap *tabulist*

   Tahap ini dilakukan dengan dengan persamaan $\Delta\tau_{ij} = \Delta\tau_{ij} + \Delta\tau_{ij}^{k}$, yaitu menambahkan *pheromone* sementara pada masing-masing *tabulist* dengan *pheromone* global. Contoh pada *tabulist* 1:
   $\Delta\tau_{ij}$= 134.9274224269 + 8.0624483061 = 142.989870733.

Maka berdasarkan perhitungan dari tahapan 1 sampai 4 perhitungan algoritme ACO, diperoleh hasil seperti pada Tabel 4.

Tabel 4. Hasil perhitungan algoritme ACO pada objek wisata

Tabulist	*Path*	*Nilai Probabilitas*	*Perolehan Pheromone*
1	1 – 3 – 2 – 5 – 4 – 6 – 7 – 8 – 9 – 11 – 10 – 12 – 13 – 14	0.124031803	142.989870733
2	2 – 3 – 1 – 7 – 9 – 8 – 10 – 11 – 5 – 6 – 4 – 12 – 13 – 14	0.113611837	143.7293221797
3	3 – 2 – 1 – 8 – 11 – 10 – 9 – 7 – 12 – 13 – 14 – 4 – 5 – 6	0.111949492	143.8600221389
4	4 – 3 – 2 – 1 – 9 – 10 – 11 – 8 – 7 – 12 – 13 – 14 – 6 – 5	0.100486273	144.879030444
5	5 – 2 – 3 – 1 – 7 – 8 – 11 – 9 – 10 – 12 – 13 – 14 – 4 – 6	0.116981086	143.4758128906
6	6 – 2 – 3 – 1 – 7 – 8 – 11 – 9 – 10 – 12 – 13 – 14 – 4 – 5	0.116981086	143.4758128906
7	7 – 1 – 3 – 2 – 5 – 4 – 6 – 8 – 9 – 11 – 10 – 12 – 13 – 14	0.124031803	142.989870733
8	8 – 10 – 9 – 11 – 1 – 3 – 2 – 5 – 4 – 6 – 7 – 12 – 13 – 14	0.111849988	143.8679687594
9	9 – 7 – 1 – 3 – 2 – 5 – 4 – 6 – 8 – 10 – 11 – 12 – 13 – 14	0.092141916	1145.7802464554
10	10 – 9 – 8 – 11 – 12 – 13 – 14 – 7 – 1 – 3 – 2 – 5 – 4 – 6	0.107601829	144.2209447517
11	11 – 10 – 9 – 8 – 12 – 13 – 14 – 7 – 1 – 3 – 2 – 5 – 4 – 6	0.11995662	143.2637693496
12	12 – 1 – 3 – 2 – 5 – 4 – 6 – 7 – 8 – 9 – 11 – 10 – 13 – 14	0.103331972	144.6049692756
13	13 – 14 – 12 – 9 – 8 – 11 – 10 – 1 – 7 – 3 – 5 – 4 – 6 – 2	0.06835129	149.557330591
14	14 – 13 – 12 – 11 – 10 – 9 – 8 – 7 – 1 – 2 – 5 – 3 – 4 – 6	0.081373965	147.216365211

5. Lakukan Iterasi
   Jika akan dilakukan iterasi maka lakukan tahap 7 terlebih dahulu, jika tidak maka tahap 6 dan 7 dapat dilewati.
6. *Update pheromone*
   *Update pheromon*e dilakukan dengan persamaan sebagai berikut:

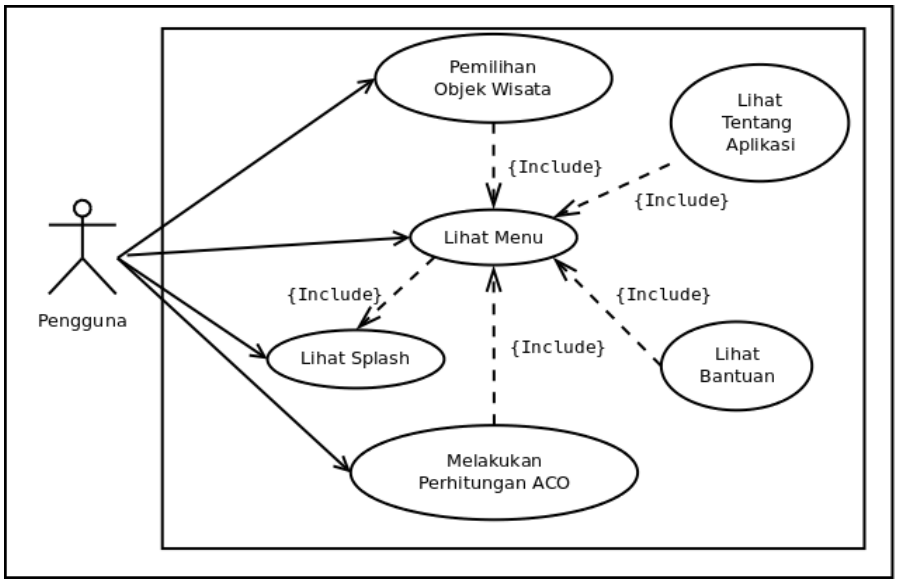
$$\tau_{ij}(t+n) = \rho\tau_{ij}(t) + \Delta\tau_{ij}$$

= 0.5 x 1 + 134.9274224269 = 135.4274224269

7. Cetak hasil
   Hasil Rute Terbaik diambil berdasarkan probabilitas terbesar dan *pheromone* terkecil yaitu pada *tabulist* 1 dan 7.

3.2.2 Analisis Sistem

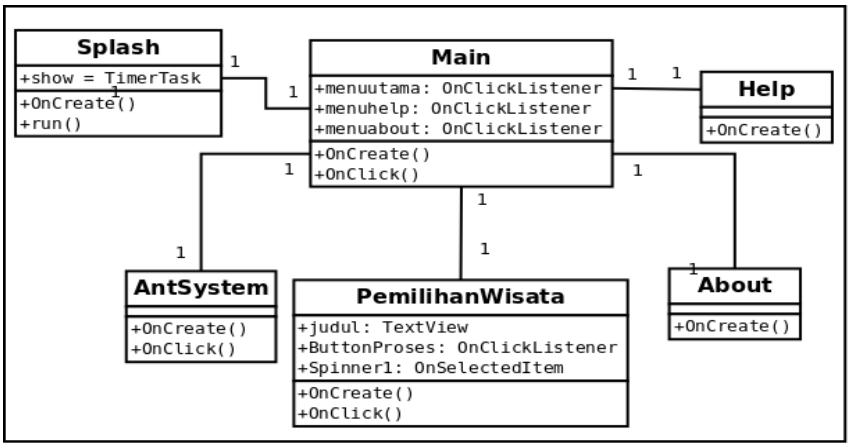
a. *System Activities* (*Use Case Diagram*)
Use case pada sistem terdiri dari satu aktor yaitu pengguna sebagai orang yang menggunakan aplikasi serta enam *use case*. Secara lebih rinci dapat dilihat pada Gambar 2.



Gambar 2. *Usecase Diagram*

b. *Class Diagram (Class Definition, Class Relation)*

1. *Class Diagram*
Aplikasi Pemilihan Objek Wisata memiliki 6 *class* yang terdiri dari *class* Splash, Main, AntSystem, PemilihanWisata dan Help seperti pada Gambar 3.



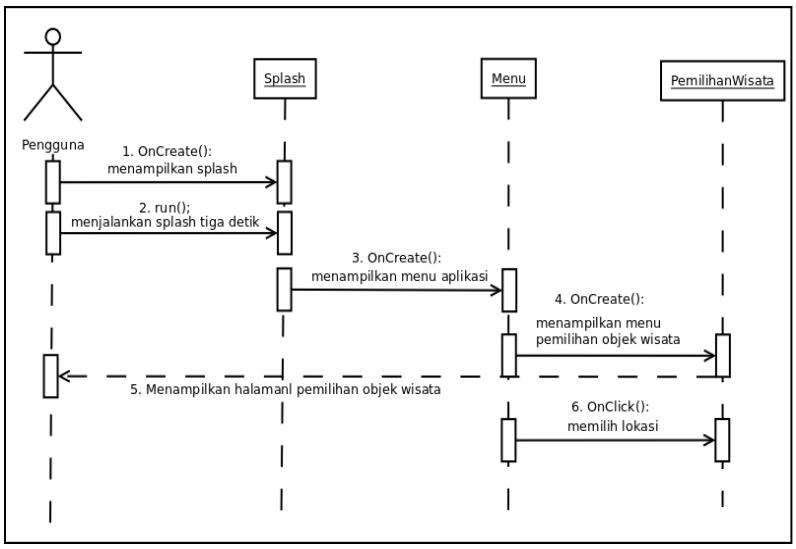
Gambar 3. Class Diagram

2. *Class Definition*
Berdasarkan *Class Diagram* pada Gambar 3, maka deskripsi setiap *class* dapat dilihat pada Tabel 5.

Tabel 5. Deskripsi *class* diagram

No	*Nama Class*	*Deskripsi*
1	Splash	Merupakan kelas untuk menampilkan splash
2	Main	Merupakan kelas untuk menampilkan menu aplikasi
3	PemilihanWisata	Merupakan kelas untuk melakukan pemilihan objek wisata
4	AntSystem	Merupakan kelas untuk melakukan perhitungan algoritme ACO dalam menentukan rute terbaik
5	Tentang	Merupakan kelas untuk menampilkan halaman tentang aplikasi
6	Bantuan	Merupakan kelas untuk menampilkan halaman bantuan

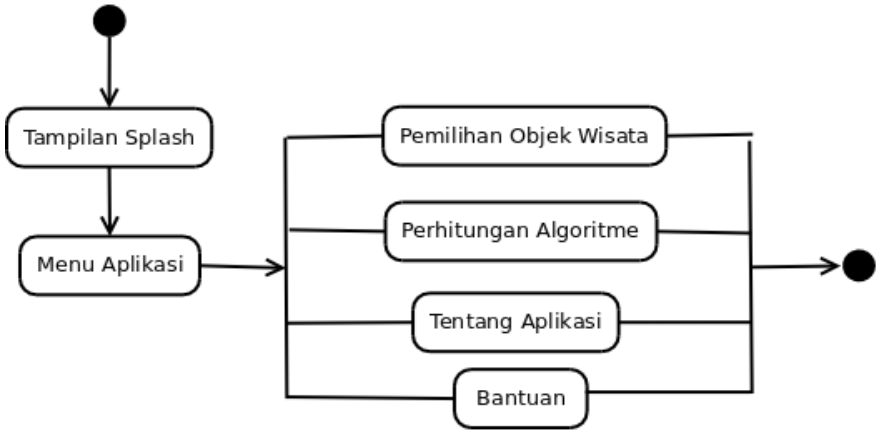
c. Object Interaction (Sequence Diagram)
Diagram Sequence aplikasi Pemilihan Objek Wisata dapat dilihat pada gambar 4.



Gambar 4. Sequence Diargam Pemilihan Objek Wisata

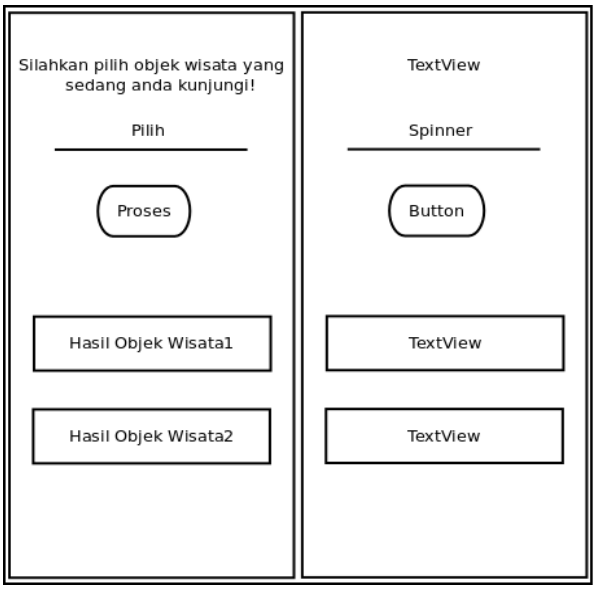
3.3 Design Phase

Pada tahapan *Design Phase* menggunakan *Object Oriented Design* (OOD) berupa desain proses dan desain antarmuka aplikasi. Dimana desain proses menggambarkan proses aplikasi secara keseluruhan, dimulai dari tampilan *splash* saat aplikasi pertama dibuka, pemilihan menu hingga keluar dari aplikasi. Dapat dilihat pada Gambar 5 desain proses aplikasi Pemilihan Objek Wisata.



Gambar 5. Desain Proses Aplikasi Pemilihan Objek Wisata

Berdasarkan desain proses pada Gambar 5, desain antarmuka aplikasi Pemilihan Objek Wisata pada menu Pemilihan Objek Wisata dapat dilihat pada Gambar 6.



Gambar 6. Desain Antarmuka Pemilihan Objek Wisata

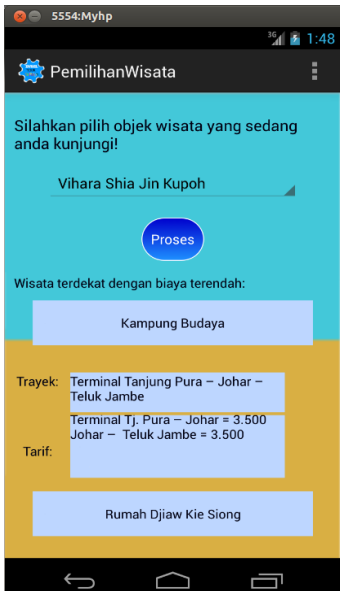
3.4 Implementation Phase

Tahap implementasi sistem merupakan tahap di mana sistem siap untuk dioperasikan dengan tujuan untuk menguji coba sistem yang telah dibuat.

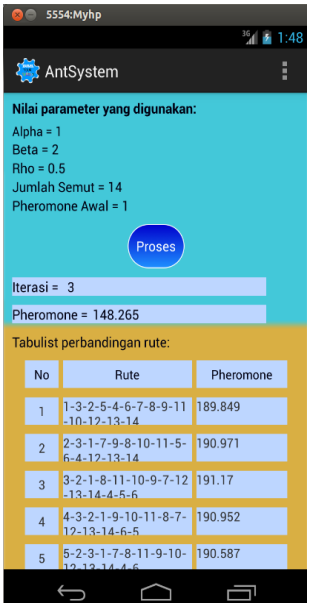
a. Instalasi Sistem
Instalasi sistem terdiri dari instalasi perangkat lunak dan perangkat keras. Perangkat lunak yang dibutuhkan agar dapat menjalankan aplikasi ini adalah sistem operasi minimum *Android Gingerbread 2.3.1*. Agar aplikasi dapat berjalan dengan baik dibutuhkan perangkat keras *smartphone*/tablet dengan minimal spesifikasi sebagai berikut:

1. *Processor* minimum 1000 *Mhz*
2. RAM dengan kapasitas minimum 1 Gb
3. *Internal Storage* dengan kapasitas minimum 1 Gb

b. Fitur Aplikasi
Aplikasi pemilihan objek wisata Karawang ini memiliki fitur Pemilihan Wisata, Perhitungan Algorutme ACO, Tentang Aplikasi dan Bantuan. Berikut adalah tampilan fitur Pemilihan Objek Wisata pada Gambar 7 dan Perhitungan Algoritme ACO pada Gambar 8.



Gambar 7. Menu Pemilihan Objek Wisata



Gambar 8. Perhitungan Algoritme ACO

4. KESIMPULAN

Berdasarkan tahap-tahap penelitian yang telah dilakukan maka dapat disimpulkan :

a. Dalam menentukan nilai optimal menggunakan algoritme ACO dengan variabel biaya dapat dilakukan dengan menentukan probabilitas terbesar dan nilai *pheromone* terkecil berdasarkan 8 tahapan perhitungan algoritme ACO, yakni identifikasi d_{ij}, inisialisasi parameter awal, menentukan jumlah semut, membuat *tabulist*, menghitung probabilitas dan *pheromone*, lakukan iterasi, kemudian lakukan *update pheromone* dan cetak hasil rute terbaik.
b. Pembangunan sistem dilakukan dengan melakukan beberapa tahapan metode SDLC *waterfall* dari *projet planning phase*, *analysis phase*, *design phase* dan *implementation phase*, yaitu pada setiap tahapan menjelaskan perkembangan pembangunan sistem.

DAFTAR PUSTAKA

[1] Fatkhurrozi, Bagus., dan Setyowati, Ika. 2015. "P'encarian Rute Terpendek Objek Wisata di Magelang Mengunakan Ant Colony Optimization (ACO)." *Prosiding SENATEK. 205-212.*
[2] Ramuna, Maretta., dan Mahmudy, Wayan. 2015. "Optimasi persediaan barang dalam produksi jilbab menggunakan alroritma genetika." *DORO: Repository Jurnal Mahasiswa PTIIK Universitas Brawijaya.* Volume, 5. No. 14.
[3] Kurniawan, Defri., dan Supriyanto, Catur. 2013. "Optimasi Algoritma *Support Vector Machine* (SVM) Menggunakan *ADABOOST* untuk Penilaian Risiko Kredit." *Jurnal Teknologi Informasi.* Volume, 9. Nomor, 1.
[4] Paryanti. 2009. "Optimasi Strategi Algoritma Greedy untuk Menyelesaikan Permasalahan *Knapsack* 0-1." *SemnasIF.* A-101-110.
[5] Dorigo, Marco., and Gambardella, LM.1996. "*Ant Colonies for the traveling salesman problem*." *Université Libre de Bruxelles.*
[6] Joni, IDMAB., dan Nurcahyawati, Vivine. 2012. "Penentuan Jarak Tetpendek pada Jalur Distribusi Barang di Pulau Jawa dengan Menggunakan Algoritma Genetika." *JANAPATI.* Volume, 1. Nomor, 3. 244-258.
[7] Togatorop, Disbun. 2014. "Perancangan Aplikasi Pencarian Jalur Terpendek dengan Algoritma *TABU* SEARCH." *Pelita Informatika Budi Darma.* Volume, 7. Nomor, 1. 49-54.
[8] Anadayani, Sri., dan Perwitasari, EW. 2014. "Penentuan Rute Terpendek Pengambilan Sampah di Kota Merauke Menggunakan Algoritma Dijkstra." *SEMANTIKA* . ISBN: 979-26-0276-3. 164-170.
[9] Satzinger, John, w., et al. 2010. *Systems Analysis and Design in a Changing World, Fifth Edition.* Boston: Course Technology.
[10] Suyanto. 2010. *Algoritma Optimsi Deterministik dan Probabilistik.* Yogyakarta: Graha Ilmu