Perbandingan Algoritma Dijkstra Dan Algoritma Ant Colony Dalam Penentuan Jalur Terpendek

Finsa Ferdifiansyah – NIM 0710630014
Jurusan Teknik Elektro Konsentrasi Rekayasa Komputer
Fakultas Teknik
Universitas Brawijaya, Jl. M.T. Haryono No. 167, Malang 65145, Indonesia
finsa@ferdifiansyah.com
Dosen Pembimbing : 1. Adharul Muttaqin ST.,MT
2. Ir. Muhammad Aswin, MT

***Abstrak* – Pencarian jalur terpendek merupakan pencarian sebuah jalur pada graf berbobot yang meminimalkan jumlah bobot sisi pembentuk jalur tersebut. Dengan begitu jalur yang dihasilkan merupakan jalur yang memiliki bobot atau jarak yang paling sedikit. Salah satu penerapan pencarian jalur terpendek terdapat pada aktivitas maskapai penerbangan yang mana jalur-jalur antar kota yang dilewatinya akan membentuk suatu graf berarah dan berbobot. Dari graf yang terbentuk inilah akan diproses menggunakan algoritma Dijkstra dan Ant Colony untuk menentukan jalur terpendek dari suatu kota ke kota yang lain.**

Pada proses Algoritma, Dijkstra memerlukan data jarak setiap kota terlebih dahulu sebelum memulai proses algortimanya. Sedangkan pada Algoritma Ant Colony, tidak memerlukan jarak setiap kota karena pada Ant Colony jarak antar kota dihitung setelah semut menyelesaikan perjalanannya. Sehingga Algoritma Dijkstra hanya bisa berjalan jika terlebih dahulu diketahui jarak tiap kota, sedangkan pada Algoritma Ant Colony tidak memerlukan jarak tiap kota untuk menjalankan prosesnya. Dari hasil proses kedua algoritma diketahui jalur yang dihasilkan oleh algoritma Dijkstra lebih konsisten dan tepat daripada algoritma Ant Colony yang mana memberikan hasil yang belum tentu sama dalam setiap prosesnya. Rata-rata memory yang digunakan pada algoritma Dijkstra sebesar 82,204 KB dan algoritma Ant Colony sebesar 90,404 KB. Sedangkan dari analisa kompleksitas waktu pada algoritma Dijkstra diperoleh persamaan $f(n) = O(2n + 13)$ dan pada Algoritma Ant Colony persamaannya $f(n) = O(5n + 3)$.

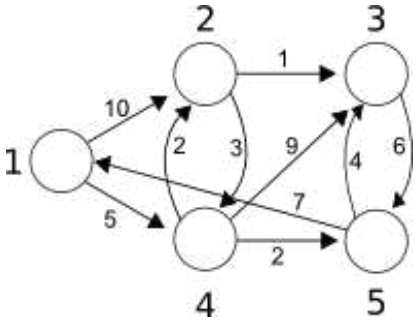
Kata Kunci— Dijkstra, Ant Colony, jalur terpendek, perbandingan algoritma

Dalam melakukan perjalanan jarak jauh seperti misalnya perjalanan antar kota-kota besar di Indonesia, tentunya tidak bisa diakukan hanya dengan sekali perjalanan karena terbatasnya transportasi atau kendala yang lain. Salah satu transportasi yang bisa melakukan perjalanan jarak jauh, antar pulau dan memerlukan waktu yang relatif singkat adalah pesawat terbang. Akan tetapi maskapai penerbangan tidak sepenuhnya menyediakan jasa perjalanan dari kota asal ke kota tujuan secara langsung, melainkan mereka memiliki jalur perjalanannya sendiri yang mana jalur tersebut menghubungkan antara kota atau bandara dengan kota terdekat lainnya. Untuk itu, kita harus mengetahui jalur-jalur antar kota yang akan dilewati agar bisa memperkitakan jalur mana saja yang akan dilalui sehingga bisa ditemukan jalur yang terpendek

Dalam ilmu komputer, untuk menentukan jalur terpendek diperlukan suatu algoritma. Algoritma yang bisa digunakan untuk menentukan jalur terpendek adalah algoritma *Dijkstra* dan *Ant Colony*. Dari permasalahan tersebut maka diperlukan suatu penelitian untuk menganalisa karakteristik maupun penerapan kedua algoritma tersebut.

A. Skema Algoritma Dijkstra

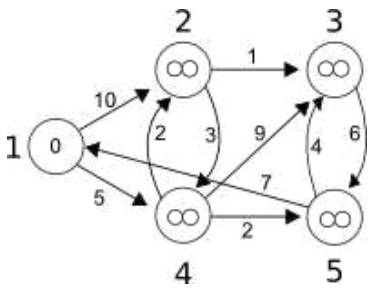
Pada graf berikut akan menentukan jalur terpendek dari *node* 1 ke *node* 3.



Gambar 1**.** Graf menentukan jalur terpendek dari *node* 1 ke *node* 3

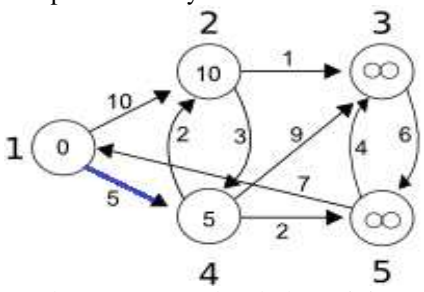
Ada beberapa jalur yang mungkin misalnya **1 -> 4 -> 3, atau 1 -> 2 -> 3** dan lain-lain, tetapi cara yang paling efektif adalah :

a. Menentukan *node* awal dengan memberikan nilai 0 pada *node* tersebut dan memberikan nilai tak hingga (∞) pada *node* yang lainnya



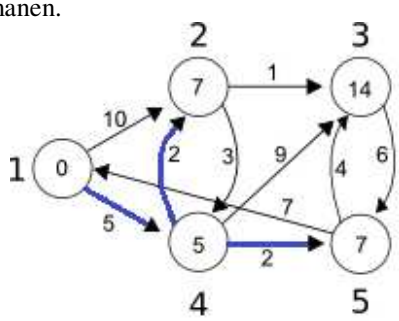
Gambar 2. *node* awal (1) bernilai 0 dan lainnya bernilai tak hingga (∞)

b. Menentukan bobot paling kecil untuk melangkah ke *node* selanjutnya. Pada gambar diketahui bahwa untuk melangkah ke *node* 4 memiliki bobot terkecil, kemudian memberikan nilai atau label permanen pada *node* 4 yaitu bernilai 5.



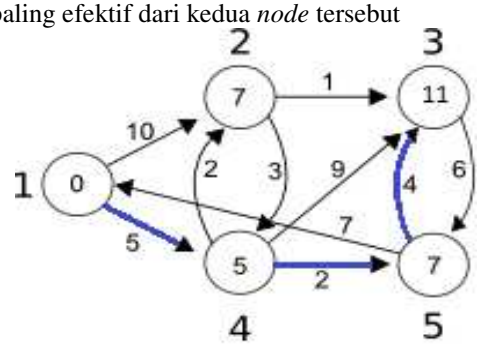
Gambar 3**.** Menentukan bobot minimum

c. Mencari *node* terpendek berikutnya yaitu yang memiliki bobot minimum dengan membandingkan nilai biaya menuju node tersebut atau melalui *node* yang telah memiliki nilai permanen.

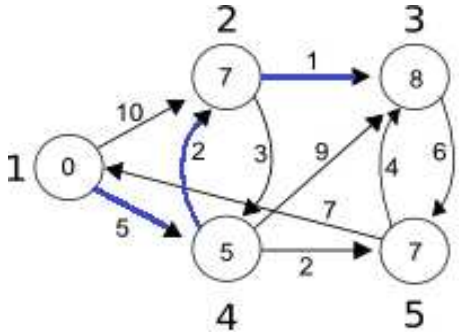


Gambar 4. Menentukan bobot minimum untuk menuju *node* selanjutnya

d. *Node* 2 dan *node* 5 memiliki nilai yang sama, maka selanjutnya dilakukan pencarian jalur yang paling efektif dari kedua *node* tersebut



Gambar 5. Menentukan jalur yang paling efektif dari *node* 5



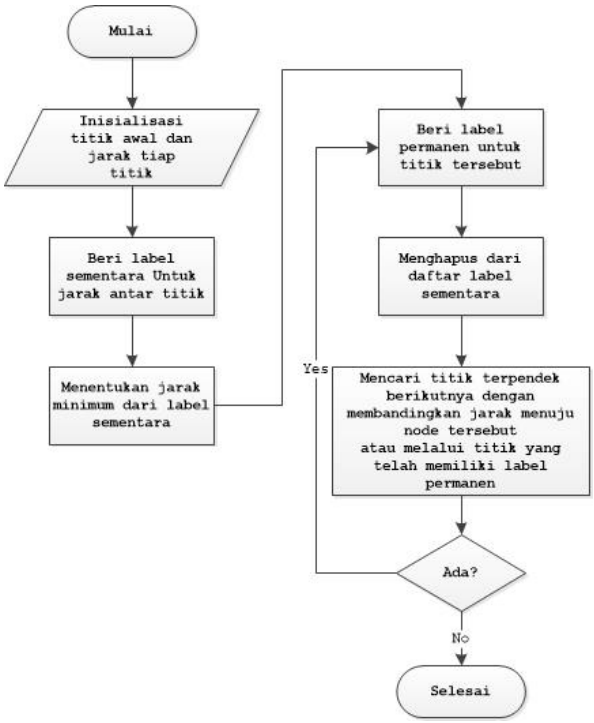
Gambar 6**.** Menentukan jalur yang paling efektif dari *node* 2

Setelah dibandingkan, ternyata jalur dari *node* 2 memiliki nilai paling minimum. Sehingga bisa ditentukan jalur terpendek dari *node* 1 ke *node* 3 adalah **1 » 4 » 2 » 3.**

B. Pseudocode Dijkstra

```
1   function Dijkstra(Graph, source):
2   for each vertex v in Graph:
3   dist[v] := infinity ;
4   previous[v] := undefined ;
5   end for
6
7   dist[source] := 0 ;
8   Q := the set of all nodes in Graph ;
9
10  while Q is not empty:
11  u := vertex in Q with smallest distance in
    dist[] ;
12  remove u from Q ;
13  if dist[u] = infinity:
14  break ;
15  end if
16
17  for each neighbor v of u:
18  alt := dist[u] + dist_between(u, v) ;
19  if alt < dist[v]:
20  dist[v] := alt ;
21  previous[v] := u ;
22  decrease-key v in Q;
23  end if
24  end for
25  end while
26  return dist;
```

C. Diagram Alir Algoritma Dijkstra



Gambar 7**.** Diagram Alir Dijkstra

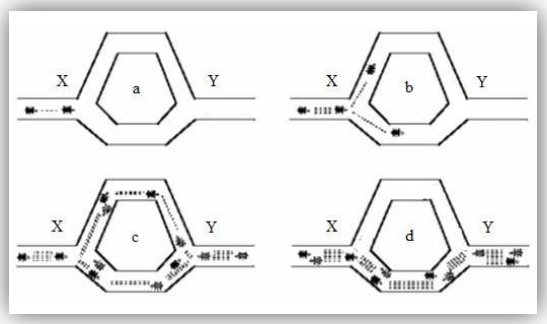
II. ALGORITMA ANT COLONY

Algoritma Semut (Ant Colony) diadopsi dari perilaku koloni semut yang dikenal sebagai sistem semut (Dorigo, 1996). Secara alamiah koloni semut mampu menemukan rute terpendek dalam perjalanan dari sarang ke tempat-tempat sumber makanan. Koloni semut dapat menemukan rute terpendek antara sarang dan sumber makanan berdasarkan jejak feromon pada lintasan yang telah dilalui. Semakin banyak semut yang melalui suatu lintasan, maka akan banyak jalur yang dibuat dan memungkinkan semakin besar untuk menentukan dari jalur terpendek yang dibuat oleh semut-semut tadi.

A. Skema Algoritma Ant Colony

Secara logika dan matematik semut-semut dari titik asal yang sama, misalnya sarang, bergerak sendiri-sendiri melewati jalur masing-masing menuju makanan sebagai titik tujuannya. Setelah mereka sampai pada titik tujuan, semut-semut ini di data satu per satu untuk mengetahui jalur yang dilalui beserta jaraknya. Semut yang menempuh jarak terpendek adalah pemenangnya dan jalur yang dilaluinya ditetapkan sebagai jalur terpendek.

Berikut adalah tahapan-tahapan algoritma Semut dalam graf :



Gambar 8. Perjalanan Semut

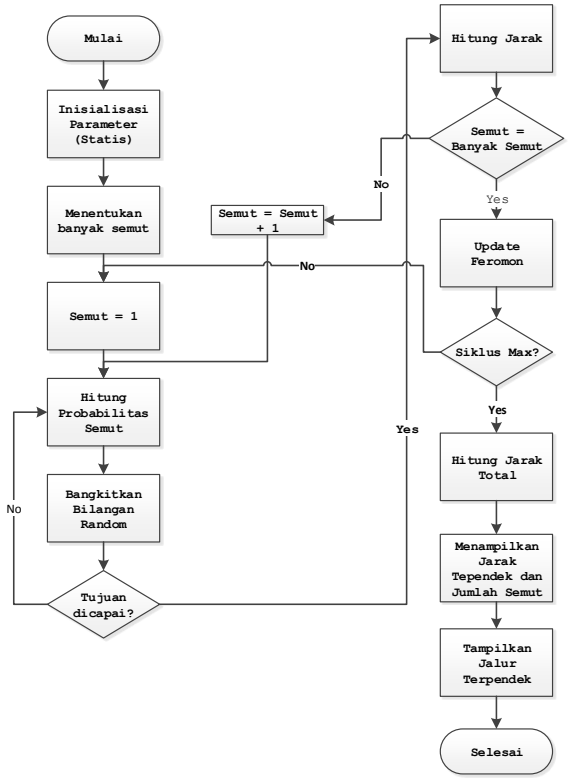
a. Pada gambar 2.9.a menunjukkan semut yang akan melakukan perjalanan mencari tempat dimana ada makanan, dari X menuju Y.
b. Semut akan melakukan gerakan secara acak menuju tempat makanan dengan jalur masing-masing. Seperti pada gambar 2.9.b semut berjalan dengan jalurnya masing-masing.
c. Setelah berjalan secara acak berdasarkan jalurnya masing-masing kemudian semut-semut tersebut akan bertemu lagi dimana tempat makanan berada seperti pada gambar 2.9.c
d. Pada saat melakukan perjalanan melalui jalurnya masing-masing, semut meninggalkan feromon sebagai jejak yang akan diikuti oleh semut yang lainnya. Semakin banyak semut dan semakin dekat jarak yang ditempuh maka feromon juga semakin kuat sehingga semut yang lainnya akan mengikuti jalur tersebut.
e. Pada optimisasi algoritma semut, proses tadi akan dilakukan secara berulang sesuai dengan siklus maksimum yang telah ditentukan.

B. Pseudocode Ant Colony

1	Inisialisasi parameter α, β, τ_0, m, Q
2	Inisialisasi titik pertama atau asal semut
3	**begin**
4	$\eta(a,b) = 1/D(a,b)$
5	$\tau(a,b) = \tau_0$
6	
7	//calculate next path, feromon and distance
8	**function** Probabilistik (from)
9	**for** m = 1; m++ : m
10	**for** t = 1; t++ : siklus
11	$P_{ab,m}(t) = \frac{[\tau_{ab}(t)]^{\alpha}\,[\eta_{ab}(t)]^{\beta}}{\sum_{t \epsilon b_{ak}}[\tau_{ab}(t)]^{\alpha}\,[\eta_{ab}(t)]^{\beta}}$
12	**end for**
13	**end for**
14	**return** dist, path;
15	
16	**function** updateFeromon ()
17	$\Delta\tau_{ab} = \frac{Q}{Lk}$
18	$next\ \tau = \rho.\tau_{ab} + \Delta\tau_{ab}$
19	**return** $next\ \tau$

```
20
21    // choice distance & shortpath
22    for x = 1; x++
23    if (min (dist(x)))
24    shortpath = path;
25    end if
26    end for
27    end;
```

C. Diagram Alir Algoritma Ant Colony

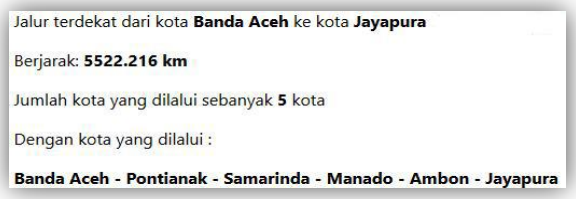


Gambar 9. Diagram Alir Ant Colony

III. PERBANDINGAN JALUR TERPENDEK

Dalam pengujian perbandingan jalur terpendek yang dihasilkan oleh kedua algoritma, penulis menguji kedua algoritma dengan data masukkan yang sama dan setiap masukkan dilakukan pengujian sebanyak sepuluh kali. Langkah-langkah pengujian untuk setiap algoritma yaitu dengan cara memasukkan data kota asal dan tujuan yang sama pada masing-masing algoritma. Pada algoritma Ant Colony nilai parameter jumlah semut dan siklus maksimum bernilai 50 dan 25.

Setelah memasukkan data dan melakukan proses pencarian dengan masing-masing algoritma, maka akan muncul hasil dari proses tersebut seperti pada gambar 10 dan gambar 11. Kemudian data hasil pengujian dicatat dan dibandingkan.



Gambar 10. Hasil pencarian jalur terpendek

Gambar 11. Jalur yang dihasilkan pada peta

Tabel 1. Hasil pengujian algoritma Dijkstra

No	Input		Output		
	Kota Asal	Kota Tujuan	Jarak Total (km)	Jumlah Kota	Jalur
1	Surabaya	Manado	1735,523	2	Surabaya - Makasar - Manado
2	Surabaya	Manado	1735,523	2	Surabaya - Makasar - Manado
3	Surabaya	Manado	1735,523	2	Surabaya - Makasar - Manado
4	Surabaya	Manado	1735,523	2	Surabaya - Makasar - Manado
5	Surabaya	Manado	1735,523	2	Surabaya - Makasar - Manado
6	Surabaya	Manado	1735,523	2	Surabaya - Makasar - Manado
7	Surabaya	Manado	1735,523	2	Surabaya - Makasar - Manado
8	Surabaya	Manado	1735,523	2	Surabaya - Makasar - Manado
9	Surabaya	Manado	1735,523	2	Surabaya - Makasar - Manado
10	Surabaya	Manado	1735,523	2	Surabaya - Makasar - Manado

Tabel 2. Hasil pengujian algoritma Ant Colony

No	Input		Output		
	Kota Asal	Kota Tujuan	Jarak Total (km)	Jumlah Kota	Jalur
1	Surabaya	Manado	1735,523	2	Surabaya - Makasar - Manado
2	Surabaya	Manado	1735,523	2	Surabaya - Makasar - Manado
3	Surabaya	Manado	1735,523	2	Surabaya - Makasar - Manado
4	Surabaya	Manado	1735,523	2	Surabaya - Makasar - Manado
5	Surabaya	Manado	1735,523	2	Surabaya - Makasar - Manado
6	Surabaya	Manado	1735,523	2	Surabaya - Makasar - Manado
7	Surabaya	Manado	1735,523	2	Surabaya - Makasar - Manado
8	Surabaya	Manado	1735,523	2	Surabaya - Makasar - Manado
9	Surabaya	Manado	1735,523	2	Surabaya - Makasar - Manado
10	Surabaya	Manado	1735,523	2	Surabaya - Makasar - Manado

Dari contoh pengujian diatas dapat diketahui bahwa penggunaan Algoritma *Dijkstra* dan *Ant Colony* menghasilkan jalur denan jalur terpendek yang sama. Pada Algoritma *Dijkstra* menghasilkan jalur dan jarak yang sama untuk setiap prosesnya, tetapi pada Algoritma *Ant Colony* jarak dan jalur yang dihasilkan untuk setiap prosesnya tidak selalu sama walaupun pada algoritma ini juga dapat menemukan jalur dan jarak terpendek.

Proses dan hasil algortima *Ant Colony* juga dipengeruhi oleh jumlah semut (m) dan sikusnya. Semakin besar jumlah semut dan siklusnya makan hasil dari algoritma *Ant Colony* akan semakin maksimal. Dengan rincian semakin banyak semut yang melakukan perjalanan, maka semakin besar kemungkinan untuk menemukan jarak terpendek. Sedangkan siklus perjalanan mempengaruhi banyaknya jalur yang ditempuh oleh semut. Dalam hal ini juga akan mempengaruhi jika dalam graf terdapat banyak simpul atau semakin banyak kota.

IV. PERBANDINGAN PENGGUNAAN MEMORY

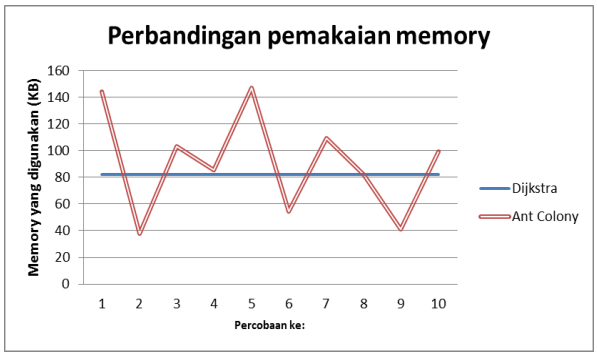
Penggunaan memory pada bahasa pemrograman PHP bisa dipantau dan ditampilkan menggunakan fungsi PHP `memory_get_usage()` . Fungsi ini berfungsi untuk mengetahi memory yang dibutuhkan selama proses. Penulis juga bisa menambahkan fungsi PHP `memory_get_peak_usage()` untuk melihat berapa jumlah penggunaan memory maksimum oleh kode PHP yang kita buat.

Untuk membaca atau mengetahui penggunaan memory, fungsi PHP `memory_get_usage()`diletakkan pada awal dan akhir source code aplikasi yang prosesnya ingin diketahui. Fungsi yang diletakkan diawal proses aplikasi disebut sebagai memory awal, sedangkan fungsi yang diletakkan pada akhir proses aplikasi disebut memory akhir. Untuk mengetahui memory yang digunakan pada aplikasi maka dilakukan pengurangan antara memory akhir dengan memory awal.

Dalam membandingkan memory yang dibutuhkan oleh kedua algoritma penulis melakukan pembacaan memory awal, memory akhir dan memory yang digunakan dari hasil pengurangan memory akhir dan memory awal. Pada masing-masing algoritma dilakukan pembacaan memory dengan data masukkan secara acak sebanyak sepuluh kali. Berikut adalah hasil pembacaan penggunaan memory kedua algoritma.

Tabel 3. Perbandingan penggunaan memory

No.	Dijkstra (KB)	Ant Colony (KB)
1	82,242	183,883
2	82,190	207,797
3	82,195	185,546
4	82,210	184,078
5	82,157	189,032
6	82,211	186,851
7	82,242	179,195
8	82,203	190,039
9	82,209	179,586
10	82,183	189,531
Jumlah	822,042	1875,538
Rata-rata	82,204	187,554



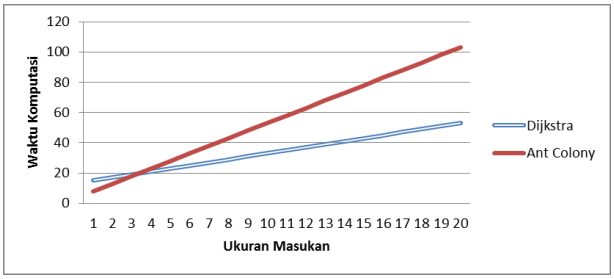
Gambar 12. Grafik perbandingan penggunaan memory

Dari hasil pengujian diatas dapat diketahui bahwa penggunaan memory pada Algoritma Dijkstra lebih rendah daripada algoritma Ant Colony. Pada algoritma Dijkstra memory yang digunakan antara proses yang satu dengan yang lain nilainya terlihat relatif lebih stabil dan konstan seperti pada gambar 5.11 yang mana grafiknya hampir lurus. Sedangkan penggunaan memory pada algoritma Ant Colony terdapat selisih yang lebih besar antar proses yang satu dengan yang lainnya dari pada algoritma Dijkstra.

Penggunaan memory pada Algoritma Ant Colony terjadi perubahan yang relatif lebih besar dibandingkan algoritma Ant Colony, hal dipengaruhi oleh jumlah semut dan siklusnya, semakin besar nilai keduanya, maka memory yang digunakan oleh Algortima Ant Colony juga semakin besar. Hal ini dikarenakan jumlah data masukan atau data yang diproses oleh algoritma tersebut semakin banyak.

V. PERBANDINGAN KOMPLEKSITAS WAKTU

Dari analisa *pseudocede* kedua algoritma diatas dapat diketahui bahwa Algoritma *Dijkstra* memiliki kompleksitas waktu $\boldsymbol{f(n) = O(2n + 13)}$ yang berarti lebih lebih rendah daripada Algoritma *Ant Colony* $\boldsymbol{f(n) = O(5n + 3)}$. Kedua kompleksites tersebut dapat dilihat dalam bentuk grafik seperti pada gambar 5.12



Gambar 13. Grafik perbandingan kompleksitas waktu

Dari grafik diatas dapat dilihat bahwa penggunaan algoritma Dijkstra memiliki pertambahan nilai komputasi waktu yang semakin besar tetapi tidak sebesar pada algoritma Ant Colony yang memiliki nilai waktu komputasi yang jauh semakin besar ketika ukuran data masukannya semakin besar.

VI. KESIMPULAN

Berdasarkan hasil perancangan, implementasi, pengujian dan analisis sistem maka dapat diambil kesimpulan sebagai berikut:

1. Perancangan aplikasi menentukan jalur terpendek dengan menggunakan algoritma *Dijkstra* tidak perlu menentukan parameter-parameter yang banyak seperti pada algoritma *Ant Colony*.
2. Pada proses Algoritma, Dijkstra memerlukan data jarak setiap kota terlebih dahulu sebelum memulai proses Algortimanya. Sedangkan pada Algoritma Ant Colony, tidak memerlukan jarak setiap kota karena pada Ant Colony jarak antar kota dihitung setelah semut menyelesaikan perjalanannya. Sehingga Algoritma Dijkstra hanya bisa berjalan jika terlebih dahulu diketahui jarak tiap kota, sedangkan pada Algoritma Ant Colony tidak memerlukan jarak tiap kota untuk menjalankan prosesnya.
3. Penggunaan memory pada algoritma *Dijkstra* menggunakan memory yang lebih rendah dengan rata-rata 82,204 KB daripada algoritma *Ant Colony* yang mana selamat pengujian memiliki rata-rata memory sebesar 90,404 KB. Pada algoritma *Dijkstra*, penggunaan memory untuk setiap prosesnya relatif sama, sedangkan pada algoritma Ant Colony penggunaan memory antar proses terdapat perbedaan yang besar karena tergantung pada parameter masukannya terutama banyak semut dan siklusnya.
4. Dari analisa kompleksitas waktu terhadap pseudocode masing-masing algoritma, Dijkstra menghasilkan persamaan $\boldsymbol{f(n) = O(2n + 13)}$ dan Ant Colony menghasilkan persamaan $\boldsymbol{f(n) = O(5n + 3)}$ yang mana bisa diketahui dari kedua fungsi tersebut waktu yang dibutuhkan algortima Dijkstra untuk memproses data lebih pendek daripada algoritma Ant Colony.

VII. REFERENSI

Jong Jek Siang. 2004. *Matematika Diskrit dan Aplikasinya pada Ilmu Komputer*, Yogyakarta: Penerbit Andi.

Abdul Kadir.2003. *Dasar Pemrograman Web Dinamis Menggunakan PHP*, Yogyakarta: Penerbit Andi.

Prahasta, E., 2001. *Konsep-konsep Dasar Sistem Informasi Geografis*. Bandung: CV. Informatika

Wardy, Ibnu Sina, 2006. *Penggunaan Graf dalam Algoritma Semut untuk Melakukan Optimisasi.* Bandung : Institut Teknologi Bandung.

Google Developers. *Getting Started - Google Maps JavaScript API v3* https://developers.google.com/maps/documentation/javascript/tutorial 17 April 2012