PERBANDINGAN ALGORITMA DIJKSTRA DAN ALGORITMA ANT COLONY OPTIMATION DALAM TRAVELLING SALESMAN PROBLEM (TSP) PADA KOTA SEMARANG

Achmad Ridwan, Aisyatul Karima, S.Kom, MCS
Teknik Informatika, Fakultas Ilmu Komputer, Universitas Dian Nuswantoro
Jl. Imam Bonjol No. 207, Semarang, 50131, (024) 3517261
111201005522@mhs.dinus.ac.id, aisyatul.karima@gmail.com

Abstrak

Traveling Salesman Problem (TSP) merupakan salah satu masalah optimasi klasik dengan konsep yang sederhana namun rumit dipecahkan secara konvensional. Tujuannya agar menememukan rute terpendek untuk melewati sejumlah kota dengan jalur tertentu sehingga setiap kota hanya terlewati satu kali dan perjalanan diakhiri dengan kembali ke kota semula. Permasalahan Traveling Salesman Problem itu sendiri adalah jalur terpendek, dalam pencarian jalur terpendek dapat menggunakan berbagai metode dalam penerapannya. Metode tersebut yaitu metode Konvensional dan metode Heuristic. Metode Konvesional adalah metode yang diterapkan dengan cara perhitungan matematis biasa. Ada beberapa metode Konvensional yang biasa digunakan untuk melakukan pencarian jalur terpendek, salah satu contohnya adalah Algoritma Dijkstra. Sedangkan Metode Heuristic adalah metode yang diterapkan dengan perhitungan kecerdasan buatan yang digunakan untuk melakukan pencarian dan optimasi, salah satu contohnya Algoritma Ant Colony atau semut, dari penjelasan singkat di atas tujuan dari penelitian ini adalah menganalisis perbandingan dalam mencari jalur terpendek yang menggunakan metode Konvensional dengan metode Heuristic. Metode Konvensional menggunakan Algoritma Dijkstra sebagai perhitungannya, sedangkan metode Heuristic menggunakan Algoritma Ant Colony atau semut dalam perhitungannya.

Kata Kunci: *Algoritma Dijkstra, Algoritma Ant Colony, TSP*

Abstract

Traveling salesman Problem (TSP) is one of the optimization of the vehicle with a simple concept but complicated to figure out that conventionally. The purpose was to find the short to get through a number of cities with specific pathways so that every city just passed a time and travel on back to the city. The problem traveling salesman The problem itself is the shortest, in the search for the shortest can use various methods in its application. The method is that method of conventional and methods Heuristic. The conventional is a method applied by the way the mathematical calculations.There are several methods the conventional, which are usually used to perform a search of the shortest, one example is the algorithm Dijkstra. The method Heuristic is the method applied by the artificial intelligence used to carry out search and optimization, one example algorithm Ant Colony or ants, from a brief explanation on the purpose of this research is analyzing the ratio in search of the shortest using a method of conventional methods Heuristic. The landing using the algorithm Dijkstra as the math, while the Heuristic to use algorithm Ant Colony or ants in the calculations.

.

Keywords: *Algorithm Dijkstra, Algorithm Ant Colony, TSP*

1. PENDAHULUAN

Dalam hidup ini kita sering melakukan perjalanan dari satu tempat ke tempat yang lain, tentu saja perjalanan yang kita lakukan tidak tanpa pertimbangan terlebih dahulu, dengan jumlah penduduk sekitar 1.6 juta jiwa. Kota Semarang dipastikan akan mengalami kepadatan lalu lintas, dimana setiap penduduk yang menggunakan alat transportasi untuk bepergian menyebabkan kemacetan di banyak ruas jalan [13]. Sedangkan setiap hari masyarakat dituntut untuk efisien dan efektif dalam memanfaatkan waktu dan biaya yang digunakan. Karena itulah dengan adanya informasi tentang jalur terpendek antar daerah sangat membantu masyarakat dalam mengoptimalkan perjalanannya.

Jalur terpendek adalah suatu jaringan pengarahan perjalanan dimana pengarah jalan ingin menentukan jalur terpendek antar dua kota, berdasarkan beberapa jalur alternative, dimana titik tujuan hanya satu [1]. Pada dasarnya permasalahan pencarian jalur terpendek antar kota merupakan pencarian jalur terpendek antar titik yang telah diketahui koordinatnya. Dengan mengetahui konsep pencarian jalur terpendek antar titik, untuk selanjutnya dapat diterapkan pada pencarian jalur terpendek pada berbagai kota yang ingin diketahui.

Dalam pencarian jalur terpendek dapat menggunakan berbagai metode dalam penerapannya. Metode tersebut di bagi menjadi 2, yaitu metode Konvensional dan metode Heuristic. Metode Konvesional adalah metode yang diterapkan dengan cara perhitungan matematis biasa. Ada beberapa metode Konvensional yang biasa digunakan untuk melakukan pencarian jalur terpendek, salah satu contohnya adalah Algoritma Dijkstra. Sedangkan Metode Heuristic adalah metode yang diterapkan dengan perhitungan kecerdasan buatan yang digunakan untuk melakukan pencarian dan optimasi, salah satu contohnya Algoritma Ant Colony atau semut, sementara ini metode yang paling efisien untuk permasalahan jalur terpendek adalah metode Konvesional Algoritma Dijkstra. secara umum,
metode Konvesional Algoritma Dijkstra cenderung lebih mudah dipahami daripada metode Heuristic, tetepi jika dibandingkan dengan metode Heuristic, hasil yang diperoleh lebih variatif, lebih akurat, tingkat kesalahan yang dihasilkan pada perhitungan lebih kecil dan waktu perhitungan yang diperlukan lebih singkat [14]. Namun, Algoritma Dijkstra menjadi tidak efisien karena simpul-simpul pada jalur akan dikunjungi kembali sehingga banyak komputasi atau perhitungan-perhitungan yang diulang [2].

Dari penjelasan singkat di atas tujuan dari penelitian ini adalah menganalisis perbandingan dalam mencari jalur terpendek yang menggunakan metode Konvensional dengan metode Heuristic. Metode Konvensional menggunakan Algoritma Dijkstra sebagai perhitungannya, sedangkan metode Heuristic menggunakan Algoritma Ant Colony atau semut dalam perhitungannya. Selain itu, tujuan lain dari penelitian ini adalah untuk memberikan informasi tentang perbedaan dan keunggulan masing-masing dalam penerapannya sehingga pengguna dapat menggunakan salah satu metode sesuai dengan yang diinginkan dan dibutuhkan. Adapun pemilihan Algoritma Ant Colony pada skripsi ini lebih ditekankan bahwa Algoritma Ant Colony dapat digunakan sebagai alternatif lain untuk penyelesaian masalah jalur terpendek. Algoritma Ant Colony mungkin tidak selalu mencapai hasil terbaik, tetapi bisa berharga dalam memecahkan masalah.

Tujuan dari laporan tugas akhir yang dibuat oleh penulis adalah sebagai

berikut :

1. Untuk mengetahui kekurangan dan kelebihan dari metode Algoritma *Dijkstra* dan Algoritma *Ant Colony*
2. Untuk mengetahui implementasi dari Algoritma *Dijkstra* dan *Ant Colony*, mana yang lebih baik.

2. LANDASAN TEORI

1. Dijkstra

Algoritma *Dijkstra* merupakan suatu algoritma yang digunakan untuk mencari lintasan terpendek untuk mencapai titik tujuan dari titik sumber pada sebuah graph. Pada prakteknya algoritma ini tidak hanya mencari lintasan terpendek dari sumber ke semua titik pada graph. Dalam proses menemukan semua jalan terpendek untuk semua tujuan, akan terbentuk pohon lintasan terpendek (spanning tree) sebagai hasil akhil dari algoritma Dijkstra yang menjadi root adalah sumber sedangkan yang menjadi leaf adalah titik tujuan

Cara kerja Algoritma Dijkstra

Algoritma ini mencari panjang lintasan terpendek dari verteks a ke verteks z dalam sebuah graph berbobot tersambung. Langkah-langkah dalam menentukan lintasan terpendek pada algoritma Dijkstra yaitu:

- Pada awalnya pilih node dengan bobot yang terendah dari node yang belum terpilih, diinisialisasikan dengan ’0’ dan yang sudah terpilih diinisialisasikan dengan ’1’
- Bentuk tabel yang terdiri dari node, status, bobot dan predecessor. Lengkapi kolom bobot yang diperoleh dari jarak node sumber ke semua node yang langsung terhubung dengan node sumber tersebut.
- Jika node sumber ditemukan maka tetapkan sebagai node terpilih.
- Tetapkan node terpilih dengan lebel permanen dan perbaharui node yang langsung terhubung.
- Tentukan node sementara yang terhubung pada node yang sudah terpilih sebelumnya dan merupakan bobot terkecil dilihat dari tabel dan tentukan sebagai node terpilih berikutnya.
- Apakah node yang terpilih merupakan node tujuan? Jika ya, maka kumpulan node terpilih atau predecessor merupakan rangakaian yang menunjukkan lintasan terpendek.
- Begitu seterusnya hingga semua node terpilih.

Pseudocode algoritma Dijkstra adalah sebagai berikut:

```
procedure Dijkstra (INPUT m: matriks, a : simpul awal)
{
mencari lintasan terpendek dari simpul awal a ke semua simpul
  lainnya.
masukan : matriks ketetanggaan (m) dari graph berbobot G dan
  simpul awal a
keluaran : lintasan terpendek dari a ke semua simpul lainnya.
}
kamus :
        S: array [1..n] of integer
        d: array [1..n] of integer
        i: integer
Algoritma:
        {Langkah 0 (inisialisasi:)}
        Traversal [1..n]
        s_i ← 0
                d_i ← m_ai
                { Langkah 1: }
                s_a ← 1
                d_a ← ∞
                { langkah 2,3,...,n-1:}
                Traversal {2..n-1}
                        Cari j sedemikian sehingga s_j = 0
                Dan
                        d_j = min {d1,d2,...,dn}
                        s_j ← 1 {simpul j sudah terpilih}
                        Perbarui d, untuk i = 1,2,3,s.d.n dengan :
                        d_i (baru) = min {d_i (lama , d_j + m_ji}
```

Jika menggunakan algoritma Dijkstra untuk menentukan jalur terpendek dari suatu graph,maka akan menemukan jalur yang terbaik, karena pada waktu penentuan jalur yang akan dipilih, akan dianalisis bobot dari node yang belum terpilih, lalu dipilih node dengan bobot yang terkecil. Jika ternyata ada bobot

yang lebih kecil melalui node tertentu, maka bobot akan dapat berubah. Algoritma Dijkstra akan berhenti ketika semua node sudah terpilih, dan dengan algoritma Dijkstra ini dapat menemukan jarak terpendek dari seluruh node, tidak hanya untuk node dari asal dan tujuan tertentu saja.

Algoritma Dijkstra menggunakan waktu sebesar O(V*logV+E) di mana V dan E adalah banyaknya verteks dan arc. Kompleksitas algo rit ma Dijkstra ada lah O(n^2). Sehingga untuk mencari semua pasangan verteks terpendek, total waktu asimptotik komputasinya adalah: T(n)=n.O(n^2)=O(n^3), algoritma Dijktra lebih menguntungkan dari sisi running time.

2. Algortima Ant Colony

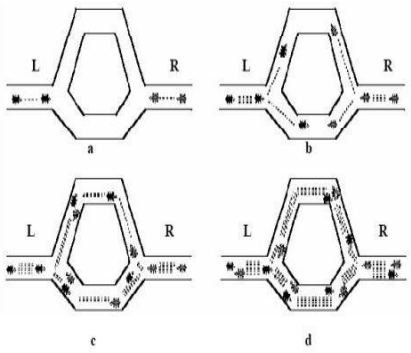
Ant Colony Optimization (ACO) diadopsi dari perilaku koloni semut yang dikenal sebagai sistem semut [10]. Semut mampu mengindera lingkungannya yang kompleks untuk mencari makanan dan kemudian kembali ke sarangnya dengan meninggalkan zat Pheromone pada rute-rute yang mereka lalui.

Pheromone adalah zat kimia yang berasal dari kelenjar endokrin dan digunakan oleh makhluk hidup untuk mengenali sesama jenis, individu lain, kelompok, dan untuk membantu proses reproduksi. Berbeda dengan hormon, Pheromone menyebar ke luar tubuh dan hanya dapat mempengaruhi dan dikenali oleh individu lain yang sejenis (satu spesies).

Proses peninggalan *Pheromone* ini dikenal sebagai stigmery, yaitu sebuah proses memodifikasi lingkungan yang tidak hanya bertujuan untuk mengingat jalan pulang ke sarang, tetapi juga memungkinkan para semut berkomunikasi dengan koloninya.

Seiring waktu, bagaimanapun juga jejak Pheromone akan menguap dan akan mengurangi kekuatan daya tariknya. Lebih cepat setiap semut pulang pergi melalui rute tersebut, maka Pheromone yang menguap lebih sedikit. Begitu pula sebaliknya jika semut lebih lama pulang pergi melalui rute tersebut, maka Pheromone yang menguap lebih banyak.

Secara jelasnya cara kerja semut menemukan rute terpendek dalam ACO adalah sebagai berikut : Secara alamiah 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 kaki pada lintasan yang telah dilalui. Semakin banyak semut yang melalui suatu lintasan, maka akan semakin jelas bekas jejak kakinya. Hal ini akan menyebabkan lintasan yang dilalui semut dalam jumlah sedikit, semakin lama akan semakin berkurang kepadatan semut yang melewatinya, atau bahkan akan tidak dilewati sama sekali. Sebaliknya lintasan yang dilalui semut dalam jumlah banyak, semakin lama akan semakin bertambah kepadatan semut yang melewatinya, atau bahkan semua semut akan melalui lintasan tersebut [8].



gambar 2.16: Perjalanan semut dari sarang ke sumber makanan

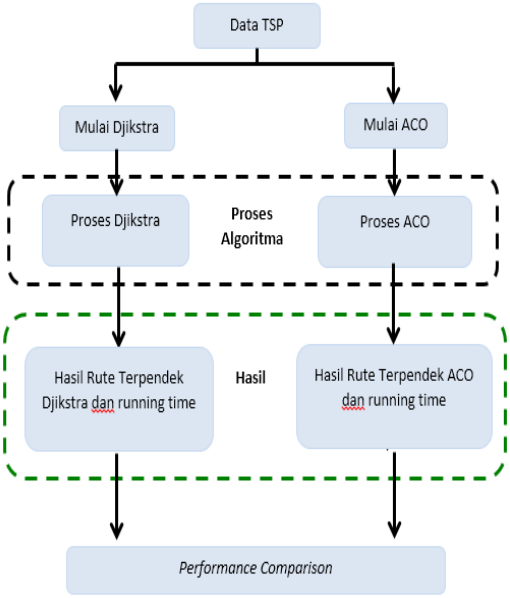
Gambar 2.16a di atas menunjukkan ada dua kelompok semut yang akan melakukan perjalanan. Satu kelompok bernama L yaitu kelompok yang berangkat dari arah kiri yang

merupakan sarang semut dan kelompok lain yang bernama kelompok R yang berangkat dari kanan yang merupakan sumber makanan. Kedua kelompok semut dari titik awal keberangkatan sedang dalam posisi pengambilan keputusan jalan sebelah mana yang akan diambil. Kelompok semut L membagi dua kelompok lagi. Sebagian melalui jalan atas dan sebagian melalui jalan bawah. Hal ini juga berlaku pada kelompok semut R. Gambar 2.16.b dan gambar 2.16.c menunjukkan bahwa kelompok semut berjalan pada kecepatan yang sama dengan meninggalkan Pheromone (jejak kaki semut) di jalan yang telah dilalui. Pheromone yang ditinggalkan oleh semut - semut yang melalui jalan atas telah mengalami banyak penguapan karena semut yang melalui jalan atas berjumlah lebih sedikit dari pada jalan yang di bawah. Hal ini dikarenakan jarak yang ditempuh lebih panjang daripada jalan bawah. Sedangkan Pheromone yang berada di jalan bawah, penguapannya cenderung lebih lama. Karena semut yang melalui jalan bawah lebih banyak daripada semut yang melalui jalan atas. Gambar 2.16.d menunjukkan bahwa semut-semut yang lain pada akhirnya memutuskan untuk melewati jalan bawah karena Pheromone yang ditinggalkan masih banyak. Sedangkan Pheromone pada jalan atas sudah banyak menguap sehingga semut-semut tidak memilih jalan atas tersebut. Semakin banyak semut yang melalui jalan bawah maka semakin banyak semut yang mengikutinya.

Demikian juga dengan jalan atas, semakin sedikit semut yang melalui jalan atas, maka Pheromone yang ditinggalkan semakin berkurang bahkan hilang. Dari sinilah kemudian terpilihlah rute terpendek antara sarang dan sumber makanan..

3. METODE PENELITIAN

Metode penelitian terdiri dari tahap tahap dilakukan penelitian terhadap algoritma yang digunakan yaitu algoritma djikstra dan ant colony. Tahap penelitian dapat dilihat pada gambar 3.1.



Gambar 3.1 : Metode Penelitian

Tahap-tahap implementasi metode djikstra

Pertama-tama tentukan titik mana yang akan dijadikan node awal, lalu beri bobot jarak pada node pertama ke node terdekat satu persatu, Dijkstra akan melakukan pengembangan pencarian dari satu titik ke titik lain selanjutnya tahap demi tahap inilah urutan logika dari algoritma Dijkstra:

1. Beri nilai bobot (jarak) untuk setiap titik ke titik lainnya, lalu set nilai 0 padanode awal dan nilai tak hingga terhadap node lain.
2. Set semua node “belum terjamah” dan set node awal sebagai node keberangkatan.
3. Dari node keberangkatan, pertimbangkan node tetangga yang belum terjamah dan hitung jaraknya dari titik keberangkatan. Sebagai contoh, jika titik keberangkatan A

ke b memiliki bobot jarak 6 dan dari B ke node C berjarak 2, maka jarak ke C melewati B menjadi 6+2=8. Jika jarak ini lebih kecil dari jarak sebelumnya maka jarak sebelumnya dihapus, dan menyimpan ulang data jarak deng jarak yang baru.

4. Saat kita selesai mempertimbangkan setiap jarak terhadap node tetangga, tandai node yang telah terjamah sebagai "node terjamah". Node terjamah tidak akan di cek kembali kembali, jarak yang disimpan adalah jarak terakhir dan yang paling minimal bobotnya.

Tahap-tahap implementasi metode Ant Colony

Sama halnya dengan cara kerja semut dalam mencari jalur yang optimal, untuk mencari jalur terpendek, diperlukan beberapa langkah untuk mendapatkan jalur yang optimal, antara lain :

1. Pertama-tama inisialisasi parameter-parameter, antara lain banyaknya semut, menentukan titik awal, pheromone awal

2. menentukan titik-titik yang akan dituju, kemudian ulangi proses sampai semua titik terlewati. untuk menentukan titik yang akan dituju dapat menggunakan persamaan 2 atau 3, yaitu:

   jika $q \geq q_o$ maka pemilihan titik yang akan dituju menerapkan aturan yang ditunjukan oleh persamaan 2

$$\text{temporary (t,u)} = [\tau(t,u\_i)] \cdot [\cap (t,u_i)] \; i = 1.2.3, \ldots\ldots, n$$

$$v = \max\{[r(t,u_i)] \cdot [n(t,u_i)]\} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

dimana v = titik yang akan dituju

sedangkan jika $q > q_o$ digunakan peramaan (2)

$$v = p_i\,(t,v) = \frac{[r(t,v)] \cdot [n(t,v)]}{\sum_{i=0}^{n} [r(t,u_i)] \cdot [n(t,u_i)]} \quad \ldots\ldots(2)$$

Dengan

$$n(t,u_i) = \frac{1}{jarak\,(t,u)}$$

jika titik yang dimaksud bukanlah titik yang akan akan dilalui, maka kembali ke titik sebelumnya.

3. Apabila telah mendapatkan titik yang dituju, pheromone masing-masing pada titik tersebut diubah dengan menggunakan persamaan 3, yaitu :

   $$r(t,v) \leftarrow (1-p) \cdot \tau(t,v) + \rho \cdot \Delta\tau(t,v)$$

   .....(3)

   $$\Delta\tau(t,v) = \frac{1}{L_{nn} \cdot c}$$

   Dimana

   L_{nn} = panjang perjalanan yang diperoleh

   c =jumlah lokasi

   p =parameter dengan nilai 0 sampai 1

   $\Delta\tau$ =perubahan pheromone

4. Setelah proses diatas selesai, hitung panjang lintasan masing-masing semut, kemudian akan didapatkan panjang lintasan yang minimal.
5. Ubah pheromone pada titik-titik yang yang termuat dalam lintasan tersebut.
6. Setelah semua proses telah dilalui, maka akan didapatkan lintasan dengan panjang lintasan yang minimal.

4. HASIL DAN PEMBAHASAN

Data yang digunakan dalam penelitian merupakan data TSP pada kota semarang bersumber dari google maps dengan menaruh titik sumber dan tujuan sehingga diperoleh jarak antar kecamatan di semarang. Data terdiri dari nama-nama dan jarak antar kecamatan di kota semarang. Jumlah kecamatan pada kota semarang sebanyak 16. Daftar kecamatan kota semarang dan jarak ditunjukan pada tabel 4.1 dan tabel 4.2

Tabel 4.1: Daftar kecamatan kota Semarang

No	Nama
1	banyumanik
2	candisari
3	gajah mungkur
4	gayam sari
5	genuk
6	gunung pati
7	Mijen
8	ngaliyan
No	Nama
9	pedurungan
10	semarang barat
11	semarang tengah
12	semarang timur
13	semarang utara
14	semarang selatan
15	tembalang
16	Tugu

Tabel 4.2: Jarak Antar Kecamatan (google maps)

No	Nama	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	K11	K12	K13	K14	K15	K16
1	K1	0	10.7	10.1	14.6	23.2	14.7	24.5	22.6	17	16.6	13.1	14.2	16	11.3	8.4	31.6
2	K2	10.7	0	3.8	6.9	12.7	15.7	18.9	19.9	7.8	9.1	6.2	6.5	9.6	3.6	7.8	17.5
3	K3	10.1	3.8	0	9.8	16.2	13.3	15.3	17.6	11	6.6	5.5	8.2	7.8	5	10.9	16.1
4	K4	14.6	6.9	9.8	0	9.7	18.9	19.5	20.4	3.8	8.5	4.1	2.7	7	3.9	8.6	19.2
5	K5	23.2	12.7	16.2	9.7	0	24.3	26.8	24.9	5.8	13.6	11.5	7.6	10.7	10.8	11.1	23.6
6	K6	14.7	15.7	13.3	18.9	24.3	0	14.1	19.9	28.8	12.7	16.2	18.2	17.1	16.1	24.8	19.9
7	K7	24.5	18.9	15.3	19.5	26.8	14.1	0	9.1	29.1	12.3	16.5	18.5	17.6	16.7	29.4	16.7
8	K8	22.6	19.9	17.6	20.4	24.9	19.9	9.1	0	23.5	8.8	13.1	15.1	14.2	14.9	24.2	6.6
9	K9	17	7.8	11	3.8	5.8	28.8	29.1	23.5	0	12.2	8.5	6	6	6.4	7.1	28.3
10	K10	16.6	9.1	6.6	8.5	13.6	12.7	12.3	8.8	12.2	0	4.9	7.1	5.8	6.6	19.7	9.2
11	K11	13.1	6.2	5.5	4.1	11.5	16.2	16.5	13.1	8.5	4.9	0	3.1	3.7	2.6	10.4	13.6
12	K12	14.2	6.5	8.2	2.7	7.6	18.2	18.5	15.1	6	7.1	3.1	0	4.8	3.5	10.4	16.3
13	K13	16	9.6	7.8	7	10.7	17.1	17.6	14.2	6	5.8	3.7	4.8	0	6	13.8	14.8
14	K14	11.3	3.6	5	3.9	10.8	16.1	16.7	14.9	6.4	6.6	2.6	3.5	6	0	8.7	15.1
15	K15	8.4	7.8	10.9	8.6	11.1	24.8	29.4	24.2	7.1	19.7	10.4	10.4	13.8	8.7	0	32.4
16	K16	31.6	17.5	16.1	19.2	23.6	19.9	16.7	6.6	28.3	9.2	13.6	16.3	14.8	15.1	32.4	0

Keterangan

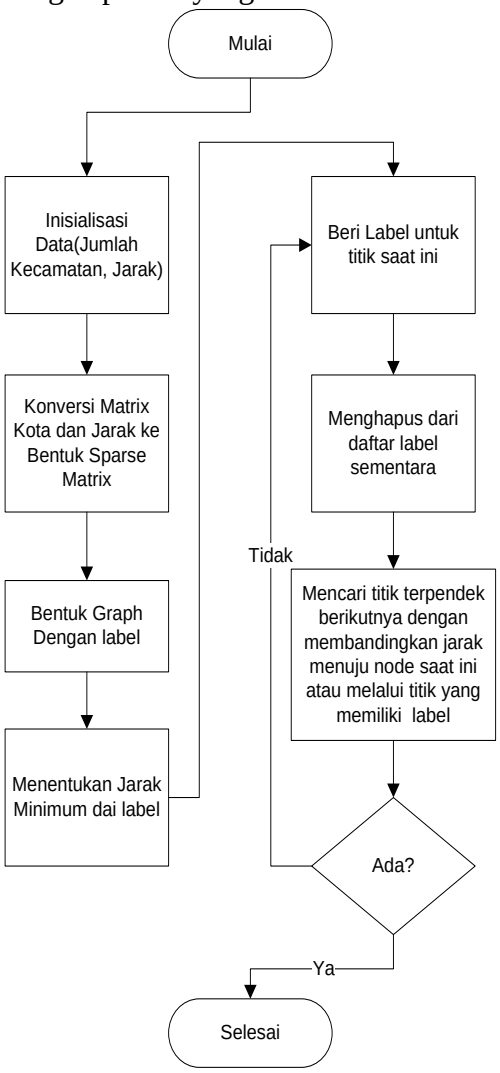
K1 banyumanik
K2 candisari
K3 gajah mungkur
K4 gayam sari
K5 genuk
K6 gunung pati
K7 Mijen
K8 ngaliyan
K9 pedurungan
K10 semarang barat
K11 semarang tengah
K12 semarang timur
K13 semarang utara
K14 semarang selatan
K15 tembalang
K16 Tugu

1. Pembahasan Dijkstra

Pada tahap ini dijelaskan tahap-tahap pengujian data dengan algoritma dijkstra. Tahap terdiri dari inisialisasi data, pembentukan graph dan hasil dari pencarian rute.

- Flowchart Algoritma Dijkstra

Pada tahap ini dijelaskan algoritma dijkstra dengan flowchart yang menggambarkan urutan proses secara detail dan hubungan antara satu proses dengan proses yang lain.



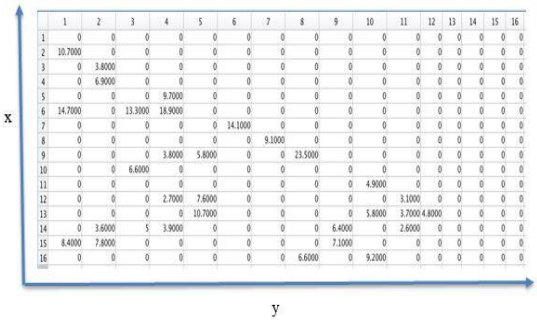
Gambar 4.1: Flowchart Dijkstra [12]

- Tahap Inisialisasi Data

Tahap awal pada pengujian algoritma dijkstra adalah menginisialisasi matrix yang digunakan untuk pengujian. Matrix digunakan untuk membentuk node-node sebagai kecamatan. Matrix merepresentasikan jarak antar kecamatan, contohnya baris 2 kolom 1 menunjukan jarak

kecamatan 1 (banyumanik) ke kecamatan 2 (candisari) yaitu 10.7 kilometer.

Tabel 4.3: Inisialisasi matrix pada *Algoritma Dijkstra*



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	10.7000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	3.8000	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	6.9000	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	9.7000	0	0	0	0	0	0	0	0	0	0	0	0
6	14.7000	0	13.3000	18.9000	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	14.1000	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	9.1000	0	0	0	0	0	0	0	0	0
9	0	0	0	3.8000	5.8000	0	0	23.5000	0	0	0	0	0	0	0	0
10	0	0	6.6000	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	4.9000	0	0	0	0	0	0
12	0	0	0	2.7000	7.6000	0	0	0	0	0	3.1000	0	0	0	0	0
13	0	0	0	0	10.7000	0	0	0	0	5.8000	3.7000	4.8000	0	0	0	0
14	0	3.6000	5	3.9000	0	0	0	0	6.4000	0	2.6000	0	0	0	0	0
15	8.4000	7.8000	0	0	0	0	0	0	7.1000	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	6.6000	0	9.2000	0	0	0	0	0	0

x

y

- Pembentukan Graph

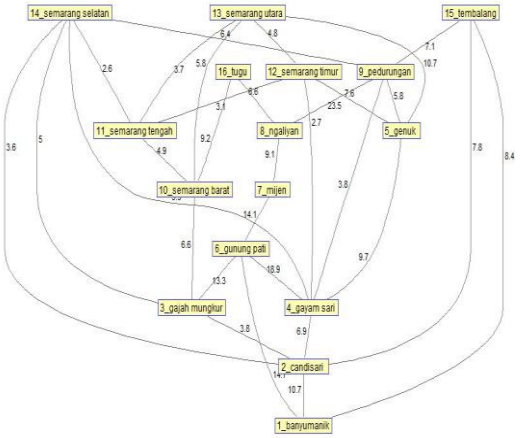
Proses pembentukan graph dilakukan dengan membentuk rute. Rute dihasilkan dari inisialisasi matrix pada tahap sebelumnya. Rute yang dibentuk merepresentasikan jarak antar kecamatan. Pada gambar 4.1 kolom 1 menunjukan inisialisasi kecamatan dan kolom 2 menunjukan jaraknya.

```
rute_UG =

    (2,1)       10.7000
    (6,1)       14.7000
   (15,1)        8.4000
    (3,2)        3.8000
    (4,2)        6.9000
   (14,2)        3.6000
   (15,2)        7.8000
    (6,3)       13.3000
   (10,3)        6.6000
   (14,3)        5.0000
    (5,4)        9.7000
    (6,4)       18.9000
    (9,4)        3.8000
   (12,4)        2.7000
   (14,4)        3.9000
    (9,5)        5.8000
   (12,5)        7.6000
   (13,5)       10.7000
    (7,6)       14.1000
    (8,7)        9.1000
    (9,8)       23.5000
   (16,8)        6.6000
   (14,9)        6.4000
   (15,9)        7.1000
   (11,10)       4.9000
   (13,10)       5.8000
   (16,10)       9.2000
   (12,11)       3.1000
   (13,11)       3.7000
   (14,11)       2.6000
   (13,12)       4.8000
```

Gambar 4.2: Inisialisasi Jarak

Setelah dibentuk rute kemudian dibentuk graph berisi node-node yang merepresentasikan posisi kecamatan dan baris jarak antar kecamatan.



gambar 4.3: pembentukan graph

- Pencarian Rute

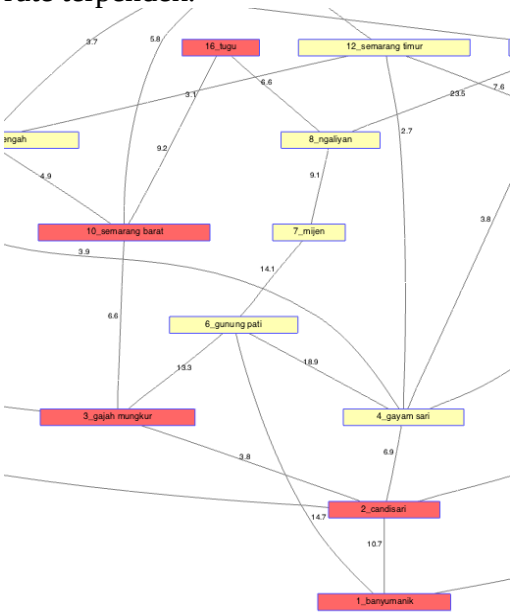
Hasil pencarian menghasilkan rute dari dari satu node ke semua node yang ada dan menghasilkan total keseluruhan rute dari node1 sampai 16. Contohnya jalur terpendek node 1 sampai node 16 yaitu 30,3 kilometer. Dengan rute 1-2-3-10-16 (banyumanik-candisari-gajahmungkur-semarang barat-tugu)

Tabel 4.4: Jarak Terbaik Semua Rute

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	10.7000	14.5000	17.6000	21.3000	14.7000	28.8000	36.9000	15.5000	21.1000	16.9000	20	20.6000	14.3000	8.4000	30.30
10.7000	0	3.8000	6.9000	15.8000	17.1000	31.2000	26.2000	10	10.4000	6.2000	9.3000	9.9000	3.6000	7.8000	19.60
14.5000	3.8000	0	8.9000	17.2000	13.3000	27.4000	22.4000	11.4000	6.6000	7.6000	10.7000	11.3000	5	11.6000	15.80
17.6000	6.9000	8.9000	0	9.6000	18.9000	33	26.5000	3.8000	10.7000	5.8000	2.7000	7.5000	3.9000	10.9000	19.90
21.3000	15.8000	17.2000	9.6000	0	28.5000	38.4000	29.3000	5.8000	15.6000	10.7000	7.6000	10.7000	12.2000	12.9000	24.80
14.7000	17.1000	13.3000	18.9000	28.5000	0	14.1000	23.2000	22.7000	19.9000	20.9000	21.6000	24.6000	18.3000	23.1000	29.10
28.8000	31.2000	27.4000	33	38.4000	14.1000	0	9.1000	32.6000	24.9000	29.8000	32.9000	30.7000	32.4000	37.2000	15.70
36.9000	26.2000	22.4000	26.5000	29.3000	23.2000	9.1000	0	23.5000	15.8000	20.7000	23.8000	21.6000	23.3000	30.6000	6.60
15.5000	10	11.4000	3.8000	5.8000	22.7000	32.6000	23.5000	0	13.9000	9	6.5000	11.3000	6.4000	7.1000	23.10
21.1000	10.4000	6.6000	10.7000	15.6000	19.9000	24.9000	15.8000	13.9000	0	4.9000	8	5.8000	7.5000	18.2000	9.20
16.9000	6.2000	7.6000	5.8000	10.7000	20.9000	29.8000	20.7000	9	4.9000	0	3.1000	3.7000	2.6000	14	14.10
20	9.3000	10.7000	2.7000	7.6000	21.6000	32.9000	23.8000	6.5000	8	3.1000	0	4.8000	5.7000	13.6000	17.20
20.6000	9.9000	11.3000	7.5000	10.7000	24.6000	30.7000	21.6000	11.3000	5.8000	3.7000	4.8000	0	6.3000	17.7000	
14.3000	3.6000	5	3.9000	12.2000	18.3000	32.4000	23.3000	6.4000	7.5000	2.6000	5.7000	6.3000	0	11.4000	16.70
8.4000	7.8000	11.6000	10.9000	12.9000	23.1000	37.2000	30.6000	7.1000	18.2000	14	13.6000	17.7000	11.4000	0	27.40
30.3000	19.6000	15.8000	19.9000	24.8000	29.1000	15.7000	6.6000	23.1000	9.2000	14.1000	17.2000	15	16.7000	27.4000	

- Sampel Rute Node

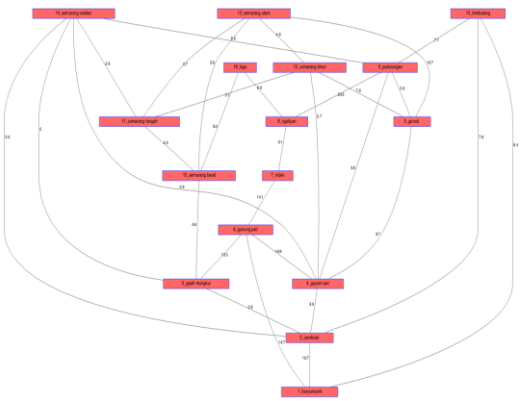
Setelah diketahui hasil perhitungan algoritma dijkstra pada tahap sebelumnya. Dilakukan penandaan graph hasil sampel rute untuk mencari rute terpendek.



gambar 4.4: sampel rute terpendek

- Hasil Rute Semua Node

Semua node yang telah dikunjungi yaitu (16-8-7-6-1-15-5-9-4-12-11-14-2-3-13-10) = 110 Kilometer. Jadi urutan kota yang dikunjungi untuk memperoleh rute terpendek adalah Tugu - Ngaliyan - Mijen - Gunung Pati - Banyumanik - Tembalang - Genuk - Pedurungan - Gayamsari - Semarang Timur - Semarang Tengah - Semarang Selatan - Candisari - Gajahmungkur - Semarang Utara - Semarang Barat.

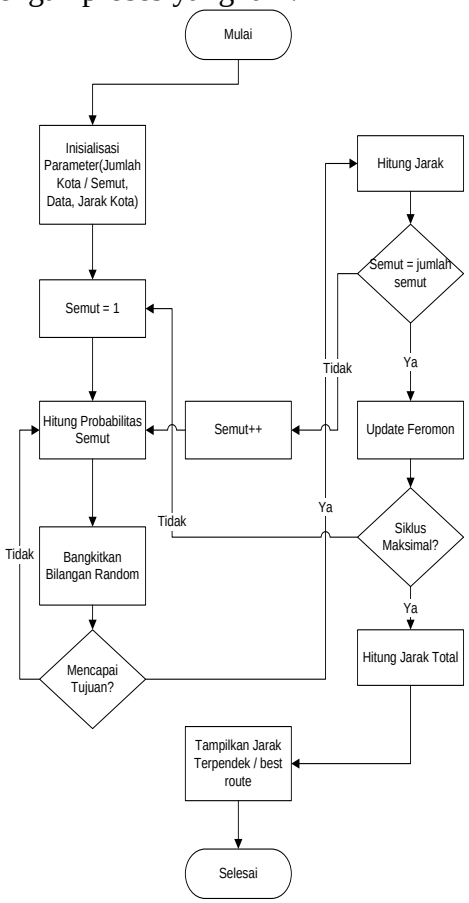


gambar 4.5: Semua Rute

2. Pembahasan Ant Colony

- Flowchart Algoritma Ant Colony

Pada tahap ini dijelaskan algoritma Ant Colony dengan flowchart yang menggambarkan urutan proses secara detail dan hubungan antara satu proses dengan proses yang lain.

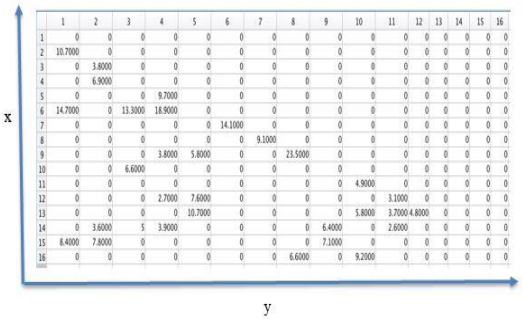


Gambar 4.6: Flowchart Ant Colony [12]

- Tahap Inisialisasi Data

Tahap awal pada pengujian algoritma Ant Colony adalah menginisialisasi matrix yang digunakan untuk pengujian. Matrix digunakan untuk membentuk node-node sebagai kecamatan

Tabel 4.5: inisialisasi matrix algoritma Ant Colony

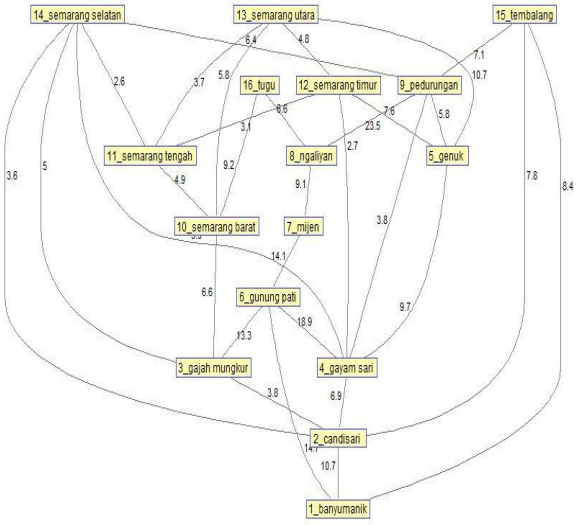


	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	10.7000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	3.8000	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	6.9000	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	9.7000	0	0	0	0	0	0	0	0	0	0	0	0
6	14.7000	0	13.3000	16.9000	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	14.1000	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	9.1000	0	0	0	0	0	0	0	0	0
9	0	0	0	3.8000	5.8000	0	0	23.5000	0	0	0	0	0	0	0	0
10	0	0	6.6000	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	4.9000	0	0	0	0	0	0
12	0	0	0	2.7000	7.6000	0	0	0	0	0	3.1000	0	0	0	0	0
13	0	0	0	0	10.7000	0	0	0	0	5.8000	3.7000	4.8000	0	0	0	0
14	0	3.6000	5	3.9000	0	0	0	0	6.4000	0	2.6000	0	0	0	0	0
15	8.4000	7.8000	0	0	0	0	0	0	7.1000	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	6.6000	0	9.2000	0	0	0	0	0	0

x

y

- Pembentukan Graph



Gambar 4.7: graph Ant Colony

- Pencarian Rute

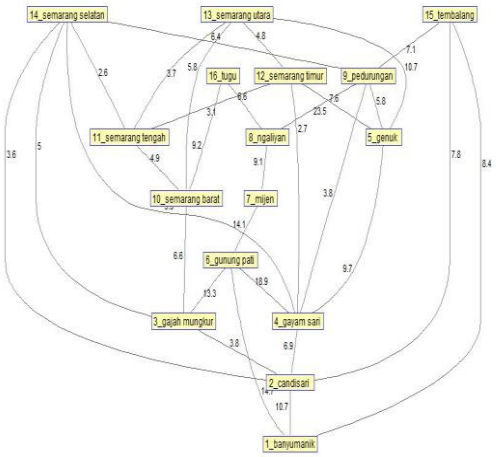
Berikut ini adalah proses tampilan Algoritma Ant Colony dalam mencari rute terpendek untuk menyusuri node (kota). Pada proses pencarian rute terpendek, nilai optimal terdapat pada iterasi ke 6 dari 200 iterasi.

```
ANT Colony Optimization (ACO) for
Traveling Salesman Problem (TSP)
Iteration 1: Best Tour Length = 115.5
Iteration 2: Best Tour Length = 115.5
Iteration 3: Best Tour Length = 115.5
Iteration 4: Best Tour Length = 113
Iteration 5: Best Tour Length = 113
Iteration 6: Best Tour Length = 110.6
Iteration 7: Best Tour Length = 110.6
Iteration 8: Best Tour Length = 110.6
Iteration 9: Best Tour Length = 110.6
```

Gambar 4.8: Proses Iterasi Ant Colony

- Hasil Rute Semua Node

Semua node yang telah dikunjungi yaitu (13-11-14-2-3-10-16-8-7-6-1-15-5-9-4-12) = 113 Kilometer. Jadi urutan kota yang dikunjungi untuk memperoleh rute terpendek adalah Tugu - Ngaliyan - Mijen - Gunung Pati - Banyumanik - Tembalang - Genuk - Pedurungan - Gayamsari - Semarang Timur - Semarang Tengah - Semarang Selatan – Candisari – Gajah mungkur - Semarang Utara - Semarang Barat.



Gambar 4.9: Hasil Semua Node Ant Colony

- Hasil Pengujian

Untuk melakukan pengujian pada algoritma,maka perlu diketahui parameter-parameteryang digunakan oleh algoritma koloni semutdan Dijkstra. Parameter pada Ant Colony antaralain:

- Siklus yaitu banyaknya siklus maksimumpencarian jalur terpendek.
- Semut yaitu banyaknya semut yang akanmelakukan dalam satu kali siklus.
- α (alfa) adalah tetapan pengendali inten-sitas jejak semut, dimana α 0
- β (beta) adalah tetapan pengendali visi-bilitas, dimana β 0.
- ρ (rho) adalah tetapan penguapan jejaksemut untuk mencegah jejak semut yangtak terhingga, dimana 0 < ρ < 1.
- Asal adalah Titik awal
- Point adalah titik tujuan yang diasum-sikan sebagai node pelanggan.

Dijkstra hanya memerlukan masukan berupa titik asal dan titik tujuan (simpangan) untuk menghasilkan jarak terpendek. yang dijadikan acuan pengujian Jarak paling optimal

3. Pengujian Jarak Paling Optimal

Berikut hasil perbandingan rute optimal dan running time pada Travelling Salesman Problem menggunakan algoritma *Dijkstra* dan algoritma *Ant Colony*, Untuk performa waktu proses algoritma *Ant Colony* lebih ungul karena hanya membutuhkan waktu 19 detik untuk memprosesnya, dibandingkan dengan Algoritma *Dijkstra* dengan waktu 3 menit, sedangkan untuk rute terdekat algoritma *Dijkstra* lebih optimal dengan jarak yang di tempuh 110 km dibandingkan dengan algoritma yang menempuh jarak 113 km

Tabel 4.6 : Perbandingan hasil

Algoritma	Rute	Jarak yang di tempuh	Running Time
Dijkstra	1-6-7-8-16-10-3-2-14-11-13-12-4-9-5-15-1	110 km	3 menit
Ant Colony	14-11-13-12-4-9-5-15-1		
	2-14-11-13-12-4-9-5-15-1-6-7-8-16-10-3-2	110 km	3 menit
	3-2-14-11-13-12-4-9-5-15-1-6-7-8-16-10-3	110 km	3 menit
	4-9-5-15-1-6-7-8-16-10-3-2-14-11-13-12-4	110 km	3 menit
	5-15-1-6-7-8-16-10-3-2-14-11-13-12-4-9-5	110 km	3 menit
	1-15-2-3-10-13-11-14-12-4-9-5-16-8-7-6-1	113 km	19 detik
	2-14-11-12-4-9-5-15-1-6-7-8-16-10-13-3-2	113 km	19 detik
	3-10-13-11-14-12-4-9-5-16-8-7-6-1-15-2-3	113 km	19 detik
	4-9-5-16-8-7-6-1-15-2-3-10-13-11-14-12-4	113 km	19 detik
	5-16-8-7-6-1-15-2-3-	113 km	19 detik

	10-13-11-14-12-4-9-5		

5. KESIMPULAN DAN SARAN

Berdasarkan penelitian yang telah dilakukan, maka dapat disimpulkan beberapa hal sebagai berikut :

1. Pencarian jalur terpendek dengan metode koloni semut tergantung dari parameter-parameter yang dimasukkan antara lain:banyaknya titik, banyak semut, Alfa (tetapan pengendali intensitas feromon), Beta(tetapan pengendali visibilitas), Rho (tetapan penguapan feromon).
2. Algoritma koloni semut membutuhkan waktu rata-rata 19 detik dengan iterasi sebanyak 100 sedangkan untuk mendapatkan jarak terpendek dari pada waktu rata-rata Dijkstra yaitu 263,87 detik.
3. Jarak optimal untuk menempuh semua titik kecamatan di kota semarang pada algoritma koloni semut sebesar 113 km sedangkan dijkstra mempunyai jarak sebesar 110 km . algoritma Dijkstra mempunyai jarak optimal dibandingkan algoritma Ant Colony.

Berdasarkan kesimpulan dan analisis laporan , saran dari peneliti untuk penelitian lebih lanjut yaitu :

1. Penelitian lebih lanjut dapat membandingkan algoritma yang telah diteliti oleh penulis yaitu koloni semut dan dijkstra dengan algoritma lain seperti bee colony, pso atau algoritma genetika.
2. Penelitian dapat dikembangkan dengan melakukan optimasi pada algoritma yang telah diteliti. Optimasi dapat dilakukan dengan cara menggabungkan algoritma yang ada sehingga dapat meningkatkan hasil agar lebih optimal..

REFERENSI

[1] Mutakhiroh, I., Saptono, F., Hasanah, N., dan Wiryadinata, R,. (2007). Pemanfaatan Metode Heuristik Dalam Pencarian Jalur Terpendek Dengan Algoritma Semut dan Algoritma Genetik. Seminar Nasional Aplikasi Teknologi Informasi. ISSN: 1907-5022. Yogyakarta.

[2] R. Kumar dan M. Kumar (2010). Exploring Genetic Algorithm for Shortest Path Optization in Data Networks. Global Journal of Computer Science and Technology Vol.10.

[3] eksono, Agus. 2009. Algoritma Ant Colony Optimiation (ACO) untuk Menyelesaikan Travelling Salesman Problem (TSP). Semarang: Universitas Diponegoro

[4] Mindaputra, Eka. 2009. Penggunaan Algoritma Ant Colony System dalam Travelling Salesman Problem (TSP) pada PT. Eka Jaya Motor. Semarang: Universitas Diponegoro

[5] Munir, Rinaldi. "Strutur Diskrit". Program Studi Teknik Informatika,2008..

[6] Nurhayati, Oky Dwi. 2010. Dasar Algoritma. http://eprints.undip.ac.id/18630/pertemuan2.pdf, 03 Agustus 2015, pk.15.40 WIB

[7]Kusumadewi, S., dan H., Purnomo,Penyelesaian Masalah Optimasi dengan Teknik-teknik Heuristi,Yogyakarta: Graha Ilmu, 2005

[8] Suarga. 2006. Algoritma Pemrograman. Andi, Yogyakarta

[9] Iing M, Fajar. 2007. Pemanfaatan Metode Heuristik Dalam Pencarian Jalur Terpendek Dengan Algoritma Semut dan Algoritma Genetika. Yogyakarta: Universitas Islam Indonesia.