βeta

Vol. 5 No. 1 (Mei) 2012, Hal. 1-20

p-ISSN: 2085-5893 / e-ISSN: 2541-0458
http://jurnalbeta.ac.id

ALGORITMA DIJKSTRA DAN ALGORITMA SEMUT DALAM MENYELESAIKAN MASALAH LINTASAN TERPENDEK (STUDI KASUS JARINGAN TRANSPORTASI PARIWISATA DI PULAU LOMBOK)

Borisman Bertinegara, Mamika Ujianita Romdhini, I Gede Adhitya Wisnu Wardhana

Abstract: On daily life, we often travel from one place to another place another by considering the efficiency, time and cost of having the accuracy required in determining the shortest path. The determination of the shortest path will be a consideration in the decision to show the path that will be pursued. The results obtained also require speed and accuracy with the help of computers. On testing, except performed the shortest route search, the search process is also carried out comparisons between Dijkstra's Algorithm and Ant Algorithm. Refers to a previous study by Izzat Pratama on 2011 is more focused on the development of tourism transport models with 2-dimensional , Pascal-based visualization technology and use Dijkstra's algorithm in the process of finding a route. The study that researcher developed more emphasis on comparison of the algorithms that is the Dijkstra Algorithm and Ant Algorithm. The results of this study form the shortest path between two specified nodes along the tracks and speed (running time) of both the algorithm that used. The Dijkstra's Algorithm in solving the shortest path between a specific pair of nodes has a speed 0 second and Ant Algorithm takes much longer, which is an average 37 seconds. However of the Ant Algorithm, the resulting trajectory is more varied. Of the results obtained, it can be said Dijkstra Algorithm is more efficient in the problem of finding the shortest path in the case of transportation networks on tourism of Lombok Island.

Keyword : *Dijkstra Algorithm; Ant Algorithm; Graph; The Shortest Path; Running Time*

A. PENDAHULUAN

Dalam kehidupan, sering dilakukan perjalanan dari satu tempat atau kota ke tempat yang lain dengan mempertimbangkan efisiensi, waktu dan biaya sehingga diperlukan ketepatan dalam menentukan jalur terpendek

antar suatu kota. Hasil penentuan jalur terpendek akan menjadi pertimbangan dalam pengambilan keputusan untuk menununjukkan jalur yang akan ditempuh. Hasil yang didapatkan juga membutuhkan kecepatan dan keakuratan dengan bantuan komputer.

Untuk menggunakan atau memfungsikan sebuah komputer maka harus terdapat program yang terdistribusi di dalamnya, tanpa program tersebut komputer hanyalah menjadi sebuah kotak yang tak berguna. Program yang terdapat pada komputer sangat bervariasi dan setiap program tersebut pasti menggunakan algoritma. Algoritma merupakan kumpulan perintah untuk menyelesaikan suatu masalah. Perintah-perintahnya dapat diterjemahkan secara bertahap dari awal hingga akhir. Masalah tersebut dapat berupa apapun dengan catatan untuk setiap masalah memiliki kriteria kondisi awal yang harus dipenuhi sebelum menjalankan algoritma. (I'ing Mutakhiroh dkk, 2007)

Untuk menentukan lintasan terpendek, ada beberapa algoritma yang bisa digunakan untuk menyelesaikannya yaitu algoritma *Best First Search, Greedy Search, Dijkstra*, *Semut (Ant Colony, Antco)*, *Moore, Bellman, Matriks, dan Ford*. Selain beberapa metode tersebut masih terdapat metode lain sebagai pengembangan dari metode Ford, yaitu Hert et al. (1968), Floyd (1962), Spira (1973), serta Johnson (1973) (Chartrand, 1993). Penulis mengimplementasikan dua algoritma yang berbeda dalam proses pencarian rute terpendek. Algoritma pertama adalah Algoritma Dijkstra, Algoritma Dijkstra merupakan algoritma untuk mencari lintasan terpendek yang diterapkan pada *graf* berarah dan berbobot, dimana jarak antar *vertex* adalah bobot dari tiap 2 *node* pada *graf* tersebut. Sedangkan yang kedua adalah Algoritma Semut (Antco), Antco 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 kaki pada lintasan yang telah dilalui. Semakin banyak semut yang melewati 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 dan sebaliknya.

Pada pengujian, selain dilakukan pencarian rute terpendek, juga dilakukan perbandingan proses pencarian antara Algoritma Dijkstra dan Algoritma Semut. Mengacu pada penelitian terdahulu yang berjudul "Model Jaringan Transportasi Pariwisata Di Pulau Lombok Menggunakan Algoritma Dijkstra" oleh Izzat Pratama yang lebih fokus Pada pembangunan model transportasi pariwisata dengan teknologi visualisasi 2 dimensi berbasis Pascal dan menggunakan algoritma Dijkstra dalam proses pencarian rutenya. Penelitian ini dikembangkan dengan lebih menekankan pada perbandingan algoritma Dijkstra dan Semut. Sehingga diharapkan tujuan akhir dari penelitian ini yaitu untuk mengetahui perbedaan hasil dari algoritma Dijkstra dan Semut pada pencarian rute terpendek.

Berdasarkan uraian pada latar belakang rumusan masalah yang diajukan pada skripsi adalah :

1. Bagaimana implementasikan Algoritma Dijkstra dan algoritma Semut dalam program komputer dengan menggunakan bahasa Borland Delphi 7?
2. Bagaimana perbandingan Algoritma Dijkstra dan algoritma Semut pada kasus Jaringan Transportasi Pariwisata di Pulau Lombok dalam menyelesaiakan masalah lintasan terpendek?

B. KAJIAN TEORI

Algoritma Djikstra

Algoritma Dijkstra berawal pada tahun 1959, pada sebuah tulisan yang berjudul *A Note on Two Problems in Connexion with Grafs* diterbitkan pada jurnal *Numerische Mathematik.* Pada tulisan ini, Edsger W. Dijkstra mengusulkan algoritma-algoritma untuk solusi dari dua masalah teoritis graf dasar: *the minimum weight.* Algoritma Dijkstra untuk masalah jalan terpendek adalah satu dari algoritma-algoritma paling ternama pada ilmu komputer dan sebuah algoritma paling popular pada operasi pencarian (Pangaribuan, 2006).

Menurut Fakhri (2008), algoritma Dijkstra merupakan salah satu bentuk algoritma *Greedy*. Algoritma ini termasuk algoritma pencarian graf yang digunakan untuk menyelesaikan masalah lintasan terpendek dengan satu sumber pada sebuah graf yang tidak memiliki sisi negatif,

dan menghasilkan sebuah pohon lintasan terpendek. Algoritma Dijkstra mencari lintasan terpendek dalam sejumlah langkah. Algoritma ini menggunakan strategi *Greedy* sebagai berikut:

Untuk setiap simpul sumber (*source*) dalam graf, algoritma ini akan mencari jalur dengan bobot minimum antara simpul tersebut dengan simpul lainnya. Algoritma ini juga dapat digunakan untuk mencari total bobot dari lintasan terpendek yang dibentuk dari sebuah simpul ke sebuah simpul tujuan. Sebagai contoh, bila simpul pada graf merepresentasikan kota dan bobot sisi merepresentasikan jarak antara 2 kota yang mengapitnya, maka algoritma Dijkstra dapat digunakan untuk mencari rute terpendek antara sebuah kota dengan kota lainnya.

Prinsip algoritma Dijkstra dalam menentukan jalur terpendek dari suatu graf adalah pada waktu penentuan jalur yang akan dipilih, akan dianalisis setiap bobot dari simpul yang belum dipilih, lalu dipilih simpul dengan bobot yang paling kecil. Apabila ada bobot yang lebih kecil melalui simpul tertentu, maka bobot akan dapat berubah, artinya jalur lintasan akan berubah. Algoritma Dijkstra akan berhenti ketika semua simpul sudah terlewati.

Misalkan diberikan graf berarah $D = (V, A)$ dengan $V = \{v_1, v_2, v_3, \cdots, v_n\}$ dan simpul v_i dan v_j terhubung oleh sisi berarah $v_{ij}, \forall v_i, v_j \in V$ dan $e_{ij} \in A$. Untuk menentukan lintasan terpendek dari sumber v_i ke simpul v_n, langkah-langkah Algoritma Dijkstra adalah sebagai berikut :

1. Dimisalkan sumber v_i sebagai sumber awal dan diberi label 0.
2. Untuk setiap simpul yang bersisian dengan sumber v_i, diberi label sementara sesuai dengan bobot sisi berarah yang berasal dari v_i dan yang lainnya diberi label sementara ∞.
3. Simpul yang mempunyai label sementara terkecil, diberikan label permanen.
4. Perhatikan simpul yang memperoleh label permanen yang terakhir, semua simpul yang bersisian dengan simpul tersebut diberi label Min{(label permanen+jarak antara simpul tersebut dengan label permanen), label simpul sebelumnya}.
5. Jika semua simpul telah mendapat label permanen, maka pencarian berhenti. Jika tidak, kembali ke langkah 3.

Algoritma Semut

Algoritma Semut 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 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.

Dalam algoritma semut, diperlukan beberapa variabel dan langkah-langkah untuk menentukan jalur terpendek, yaitu:

Langkah 1 :

a. Inisialisasi nilai parameter-parameter algoritma.
   Parameter-parameter yang di inisialisasikan adalah :
   1. Intensitas jejak semut antar kota dan perubahannya (τ_{ij})
   2. Banyak kota (n) termasuk koordinat (x,y) atau jarak antar kota (d_{ij})
   3. Kota berangkat dan kota tujuan
   4. Tetapan siklus-semut (Q)
   5. Tetapan pengendali intensitas jejak semut (α), nilai $\alpha \geq 0$
   6. Tetapan pengendali visibilitas (β), nilai $\beta \geq 0$
   7. Visibilitas antar kota $^{1}/_{d_{ij}}$ (η_{ij})
   8. Banyak semut (m)
   9. Tetapan penguapan jejak semut (ρ) , $0 < \rho \leq 1$ untuk mencegah jejak feromon yang tak terhingga.
   10. Jumlah siklus maksimum (NC_{max}) bersifat tetap selama algoritma dijalankan, sedangkan τij akan selalu diperbaharui harganya pada setiap siklus algoritma mulai dari siklus pertama $NC = 1$ sampai tercapai jumlah siklus maksimum $(NC = NC_{max})$ atau sampai terjadi konvergensi.

b. Inisialisasi kota pertama setiap semut.
Setelah inisialisasi τij dilakukan, kemudian m semut ditempatkan pada kotapertama tertentu secara acak.

Langkah 2 :

Pengisian kota pertama ke dalam *tabu list*. Hasil inisialisasi kota pertama setiap semut dalam langkah 1 harus diisikan sebagai elemen pertama *tabu list*. Hasil dari langkah ini adalah terisinya elemen pertama *tabu list* setiap semut dengan indeks kota tertentu, yang berarti bahwa setiap $tabu_k(1)$ bias berisi indeks kota antara *1* sampai *n* sebagaimana hasil inisialisasi pada langkah 1.

Langkah 3 :

Penyusunan rute kunjungan setiap semut ke setiap kota. Koloni semut yang sudah terdistribusi ke sejumlah atau setiap kota, akan mulai melakukan perjalanan dari kota pertama masing-masing sebagai kota asal dan salah satu kota-kota lainnya sebagai kota tujuan. Kemudian dari kota kedua masing-masing, koloni semut akan melanjutkan perjalanan dengan memilih salah satu dari kotakota yang tidak terdapat pada $tabu_k$ sebagai kota tujuan selanjutnya. Perjalanan koloni semut berlangsung terus menerus sampai semua kota satu persatu dikunjungi atau telah menempati *tabuk*. Jika *s* menyatakan indeks urutan kunjungan, kota asal dinyatakan sebagai $tabu_k(s)$ dan kota-kota lainnya dinyatakan sebagai $\{N\text{-}tabu_k\}$, maka untuk menentukan kota tujuan digunakan persamaan probabilitas kota untuk dikunjungi sebagai berikut:

$$P_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum_{k' \in \{N-tabu_k\}}[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}} \text{ untuk j} \in \{N - tabu_k\} \ldots. (1)$$

dan

$$P_{ij}^{k} = 0, \text{ untuk j lainnya } \ldots\ldots\ldots (2)$$

dengan i sebagai indeks kota asal dan j sebagai indeks kota tujuan.

Setelah hasil perhitungan probabilitas semut yang akan dipilih berikutnya selesai, kemudian dicari probabilitas kumulatifnya q_k dimana $q_1 = p_{ij}$ sedangkan $q_k = q_{k-1} + p_{ij}$ untuk $k = 1, 2, 3 \cdots n$. Kemudian di bangkitkan bilangan random (r) antara 0 sampai 1. Titik ke -k akan terpilih jika $q_{k-1} < r \leq q_k$.

Langkah 4 :

a. Perhitungan panjang rute setiap semut.

Perhitungan panjang rute tertutup (*length closed tour*) atau L_k setiap semut dilakukan setelah satu siklus diselesaikan oleh semua semut. Perhitungan ini dilakukan berdasarkan $tabu_k$ masing-masing dengan persamaan berikut :

$$L_k = d_{tabu_k(n),tabu_k(1)} + \sum_{s=1}^{n-1} d_{tabu_k(s),tabu_k(s+1)} \quad \text{..... (3)}$$

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

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad \text{..... (4)}$$

b. Pencarian rute terpendek

Setelah L_k setiap semut dihitung, akan didapat harga minimal panjang rute tertutup setiap siklus atau L_{minNC} dan harga minimal panjang rute tertutup secara keseluruhan adalah atau Lmin.

c. Perhitungan perubahan harga intensitas jejak kaki semut antar kota.

Koloni semut akan meninggalkan jejak-jejak kaki pada lintasan antar kota yang dilaluinya. Adanya penguapan dan perbedaan jumlah semut yang lewat, menyebabkan kemungkinan terjadinya perubahan harga intensitas jejak kaki semut antar kota. Persamaan perubahan ini adalah:

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \quad \text{..... (5)}$$

dengan $\Delta\tau_{ij}^{k}$ adalah perubahan harga intensitas jejak kaki semut antar kota setiap semut yang dihitung berdasarkan persamaan

$\Delta\tau_{ij}^{k} = \frac{Q}{L_k}$, untuk (i,j) ∈ kota asal dan kota tujuan dalam $tabu_k$ .....(6)

$\Delta\tau_{ij}^{k} = 0$, untuk (i,j) lainnya .....(7)

Langkah 5 :

a. Perhitungan harga intensitas jejak kaki semut antar kota untuk siklus selanjutnya. Harga intensitas jejak kaki semut antar kota pada semua lintasan antar kota ada kemungkinan berubah karena adanya penguapan dan perbedaan jumlah semut yang melewati. Untuk siklus

selanjutnya, semut yang akan melewati lintasan tersebut harga intensitasnya telah berubah. Harga intensitas jejak kaki semut antar kota untuk siklus selanjutnya dihitung dengan persamaan :

$$\tau_{ij} = \rho \cdot \hat{o}_{ij} + \Delta\tau_{ij} \text{ ..... (8)}$$

b. Atur ulang harga perubahan intensitas jejak kaki semut antar kota.

Untuk siklus selanjutnya perubahan harga intensitas jejak semut antar kota perlu diatur kembali agar memiliki nilai sama dengan nol.

Langkah 6 :

Pengosongan *tabu list*, dan ulangi langkah 2 jika diperlukan. *Tabu list* perlu dikosongkan untuk diisi lagi dengan urutan kota yang baru pada siklus selanjutnya, jika jumlah siklus maksimum belum tercapai atau belum terjadi konvergensi. Algoritma diulang lagi dari langkah 2 dengan harga parameter intensitas jejak kaki semut antar kota yang sudah diperbaharui.

C. PEMBAHASAN

Implementasi Algoritma Menggunakan Borlan Delphi 7

Sebelum mengimplementasi ke dalam Delphi 7, dijelaskan dahulu perangkat keras dan perangkat lunak yang digunakan dalam skripsi ini.

Spesifikasi perangkat keras yang digunakan adalah :

1. Processor Intel(R) Pentium(R) Dual CPU T2730 @ 1.73GHz.
2. Memory 1014Mb RAM.
3. Harddisk 120GB.
4. Perangkat output berupa monitor LCD Widescreen 14”.
5. Perangkat input berupa keyboard dan mouse.

Spesifikasi perangkat lunak yang digunakan adalah :

1. Operating system Microsoft WindowsXP Profesional.
2. Borland Delphi 7.

Membuat lembar kerja baru pada program Delphi dengan 2 Form kosong. Form pertama digunakan untuk tampilan pembuka, pada Form ini ditambahkan 2 buah Image, 1 buah Mainmenu dan 11 buah label yang diletakan di atas Form tersebut. Form kedua juga ditambahkan beberapa objek tambahan di atas Form, yaitu 7 buah Stringgrid, 5 buah button, 2 buah memo, 2 buah Groupbox, 4 buah label dan 3 buah Combobox.

Untuk proses pencarian lintasan terpendek menggunakan Algoritma Dijkstra dan Algoritma Semut dilakukan dengan source code berikut :

```
procedure TForm2.dijkstraClick(Sender: TObject);
var
 a,b:integer;
 st,fh:double;
begin
 startDij.Caption:=timetostr(now);
 st:=time;
 awal:=(mulai.ItemIndex)+1; tujuan:=(akhir.ItemIndex)+1;
 if (awal=0) or (tujuan=0) then
  showmessage('harus diisi!!!')
 else
  begin
   dij.Cells[awal,1]:=inttostr(1);
   dij.Cells[awal,3]:=inttostr(awal);
   jarak:=99999999;
   if (awal<>0) and (tujuan<>0) then
    begin
     for b:=1 to 77 do
     begin
      if node.Cells[b,awal]<>'' then
       begin
        dij.Cells[b,2]:=node.Cells[b,awal];
        dij.Cells[b,3]:=inttostr(awal);
       end;
     end;
     for b:=1 to 77 do
      begin
       if dij.Cells[b,2]<>'' then
        if dij.Cells[b,1]<>'' then
        else
         if strtofloat(dij.Cells[b,2]) < jarak then
          jarak:=strtofloat(dij.Cells[b,2]);
      end;
     for b:=1 to 77 do
```

```
          begin
            if floattostr(jarak)=dij.Cells[b,2] then
              dij.Cells[b,1]:=inttostr(1);
          end;
        for b:=1 to 77 do
          begin
            if dij.Cells[b,2]=floattostr(jarak) then
              c:=b;
          end;
        nex;
      end;
    nextDij;
    linDij.ColWidths[0]:=35;
    linDij.Cells[0,0]:=' Jalur ';
    a:=0;
    for b:=77 downto 1 do
      if dij.Cells[b,4]<>'' then
        begin
          linDij.Cells[(1+a),0]:=dij.Cells[b,4];
          a:=a+1;
        end;
    finishDij.Caption:=timetostr(now);
    fh:=time;
  end;
 startdij.Caption:=timetostr(fh-st);
 DijkstraStep.Enabled:=false;
 Next.Enabled:=false;
 Dijkstra.Enabled:=false;
end;

procedure TForm2.semutClick;
var
 st,fh,alpha,beta,p,p1,p2,sigma,sigmax,jalant:double;
 b,a,c,e,f,baris:integer;
begin
 startant.Caption:=timetostr(now);
```

```
st:=time;
alpha:=1; beta:=1; rho:=0.99; q:=1; m:=300; t:=0.01; max:=5;
awal:=(mulai.ItemIndex)+1; tujuan:=(akhir.ItemIndex)+1;
jalant:=10000000;
f:=0;
feromon;
repeat
 f:=f+1;
 siklus;
  for e:=1 to m do
   begin
    awal:=(mulai.ItemIndex)+1;
    studikasus;
    for a:=1 to 76 do
     begin
      clearant;
      sigma:=0;
      jalur.Cells[a,e]:=inttostr(awal);
      for b:=1 to 76 do
       begin
        if node.Cells[b,awal]<>'' then
                              begin
                                sigmax:=(power(strtofloat(fer.Cells[b,awal]),
                                alpha))*(power((1/(strtofloat(node.Cells[b,a
                                wal]))),beta));
          sigma:=sigma+sigmax;
        end;
       end;
      sigma:=roundto(sigma,-4);
      ant.Cells[2,1]:=floattostr(sigma);
      p1:=100;
      p:=0;
      rand;
      for b:=1 to 76 do
       if node.Cells[b,awal]<>'' then
        begin
```

```
                    p2:=roundto(((power(strtofloat(fer.Cells[b,awa
                    l]),alpha))*(power((1/(strtofloat(node.Cells[b,a
                    wal])))),beta)))/sigma,-4);
          ant.Cells[b,2]:=floattostr(p2);
          p:=p+p2;
          ant.Cells[b,3]:=floattostr(p);
          if r<p then
           if p<p1 then
            p1:=p;
         end;
        ant.Cells[1,4]:=floattostr(p1);
        for b:=1 to 76 do
         if ant.Cells[b,3]=ant.Cells[1,4] then
          begin
           for c:=1 to m+1 do
            begin
             node.Cells[awal,c]:='';
            end;
           c:=b;
          end;
         awal:=c;
       end;
    studikasus;
    end;
hapus;
for b:=1 to m do
 if jalur.Cells[75,b]<>'' then
  if strtofloat(jalur.Cells[75,b])<jalant then
   begin
    for c:=1 to 30 do
     linsemut.Cells[c,0]:='';
              jalant:=strtofloat(jalur.Cells[75,b]);
              baris:=b;
              a:=0;
              for c:=1 to 30 do
                if jalur.Cells[c,baris]<>'' then
```

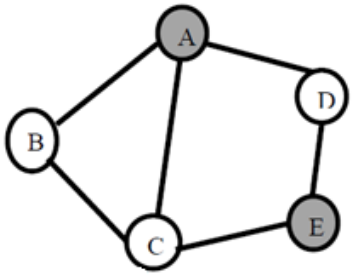
```
            begin
              linSemut.Cells[(1+a),0]:=jalur.Cells[c,baris];
              a:=a+1;
            end;
        end;
    until f=max;
    linSemut.ColWidths[0]:=35;
    linSemut.Cells[0,0]:=' Jalur ';
    hasilsemut.Lines.Add('');
    hasilsemut.Lines.Add(' jarak '+inttostr((mulai.ItemIndex)+1)+' ke
    '+inttostr((akhir.ItemIndex)+1)+' = '+floattostr(jalant));
    finishant.Caption:=timetostr(now);
    fh:=time;
    finishant.Caption:=timetostr(fh-st);
end;
```

D. UJI PROGRAM

Uji program program dilakukan dengan membandingkan hasil running program pada contoh sederhana dengan pencarian secara manual menggunakan Algoritma Dijkstra dan Algoritma Semut. Contoh kasus yang digunakan dalam pengujian ini adalah sebagai berikut :

Contoh penyelesaian kasus jalur terpendek dari kota A ke kota E menggunakan Algoritma Semut dan Algoritma Dijkstra.

Diketahui suatu graf



Gambar 3.3. *Graf*

Dengan jarak antar kota (dij) sebagai berikut:

Table 1. *Jarak Antar Kota*

	A	B	C	D	E
A	0	5	7	3	
B	5	0	4		
C	7	4	0		5
D	3			0	4
E			5	4	0

Algoritma Semut

Parameter – parameter yang digunakan adalah :

Alfa (α) = 1.00

Beta (β) = 1.00

Rho (ρ) = 0.99

Tetapan siklus semut (Q) = 1

(Mutakhiroh, I, 2007)

τ_{ij} awal = 0.01

Maksimum siklus (NC_{max}) = 2

Banyak semut (m) = 4

Dari jarak kota yang telah diketahui dapat dihitung visibilitas antar kota $(\eta_{ij}) = {}^{1}/_{d_{ij}}$

Tabel 2. *Visibilitas Antar Kota*

	A	B	C	D	E
A	0	0,2	0,14	0,33	
B	0,2	0	0,25		
C	0,14	0,25	0		0,2
D	0,33			0	0,25
E			0,2	0,25	0

Siklus 1

Semut 1 :

Probabilitas dari kota A ke kota berikutnya dihitung dengan persamaan (1).

$$\sum [\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}$$

$$= 0 + (0.01 \times 0.2) + (0.01 \times 0.14) + (0.01 \times 0.33) + 0$$

$$= 0.0067$$

$$Kota\ A = 0$$

$$Kota\ B = \frac{(0.01)^1 \times (0.2)^1}{0.0067} = 0.2985$$
$$Kota\ C = \frac{(0.01)^1 \times (0.14)^1}{0.0067} = 0.2089$$
$$Kota\ D = \frac{(0.01)^1 \times (0.33)^1}{0.0067} = 0.4926$$
$$Kota\ E = \frac{(0.01)^1 \times (0)^1}{0.0067} = 0$$

Probabilitas komulatif : 0 0.2985 0.5074 1 1

Bangkitkan bilangan random : $r = 0.6771$, maka kota yang dipilih selanjutnya adalah kota D.

Perjalanan semut terus berlanjut sampai ketemu kota tujuan, yaitu kota E. Setelah dari kota D, persinggahan semut yang mungkin (berketetanggaan) adalah kota A dan E. Karena kota A sudah dilalaui (kota awal) jadi kota yang dipilih selanjutnya adalah kota E.

Perjalanan semut 1 : A D E

Hasil perjalanan siklus I dari semua semut diperlihatkan dalam Tabel 3.

Tabel 3. *Siklus I*

Semut Ke-	Rute	Panjang Rute
1	A D E	7
2	A D E	7
3	A B C E	14
4	A B C E	14
5	A C E	12

Rute terpendek adalah A D E dengan panjang 7. Langkah selanjutnya adalah pembaruan feromon (jejak kaki semut) dengan menggunakan persamaan (8).

Hasil perubahan feromon pada siklus 1 diperlihatkan pada Tabel 4.

Tabel 4. *Perubahan Feromon*

	A	B	C	D	E
A	0.0099	0.1527	0.0993	0.2956	0.0099
B	0.1527	0.0099	0.1347	0.0099	0.005
C	0.0993	0.1347	0.0099	0.0099	0.2361
D	0.2956	0.0099	0.0099	0.0099	0.2956
E	0.0099	0.0099	0.2361	0.2956	0.0099

Untuk siklus selanjutnya feromon baru ini yang digunakan dalam proses perhitungan probabilitasnya. Proses siklus 2 hasilnya hampir sama dengan siklus 1. Hasil siklus 2 diperlihatkan pada Tabel 5.

Tabel 5. *Siklus II*

Semut Ke-	Rute	Panjang Rute
1	A B C E	14
2	A D E	7
3	A C E	12
4	A D E	7
5	A D E	7

Tabel 6. *Perubahan Feromon*

	A	B	C	D	E
A	0.0099	0.1527	0.0993	0.2956	0.0099
B	0.1527	0.0099	0.1347	0.0099	0.005
C	0.0993	0.1347	0.0099	0.0099	0.2361
D	0.2956	0.0099	0.0099	0.0099	0.2956
E	0.0099	0.0099	0.2361	0.2956	0.0099

Jadi lintasan terpendek dari kota A ke kota E menggunakan Algoritma Semut ini adalah A-D-E, hanya melalui satu simpul saja dengan jarak 7. Dibandingkan dengan hasil running program ternyata menghasilkan hasil yang sama yaitu lintasannya A-D-E dengan jarak 7.

Algoritma Dijkstra

Langkah pertama pada algoritma ini adalah memberi status ‘1’ (permanen) pada simpul awal, dalam kasus disini yaitu simpul A. Sedangkan simpul lainnya diberikan status sementara ‘0’. Karena tidak terdapat lintasan dari A ke A maka bobotnya 0.

Tabel 7. *Iterasi ke-1*

Simpul	A	B	C	D	E
Status	1	0	0	0	0
Bobot		5	7	3	
Awal	A	A	A	A	-

Pada tabel 3.6, pilih simpul bobot terkecil dengan status '0' diperoleh simpul D. Setelah itu ubah status pada simpul D menjadi '1' (permanen). Jika simpul sudah dipilih, maka cari simpul yang bertetanggaan dengan simpul D dan statusnya masih '0' kemudian beri simpul D ini jadi awalnya. Jika ternyata bobotnya sudah terisi bandingkan bobot dan ambil bobot terkecil.

Tabel 8 *Iterasi Ke-2*

Simpul	A	B	C	D	E
Status	1	1	1	1	1
Bobot		5	7	3	7
Awal	A	A	A	A	D

Algoritma berhenti karena simpul tujuannya sudah dipilih. Untuk melihat jalur mana yang terpilih telusuri awalnya. Jadi lintasan terpendek dari Algoritma Dijkstra ini adalah A-D-E dengan jarak 7. Dibandingkan dengan hasil running program ternyata menghasilkan lintasan dan jarak yang sama.

Algoritma Dijkstra				
Awal	Tujuan	Lintasan	Jarak (km)	Running Time (s)
19	27	19-49-50-27	31.5	0
	42	19-49-45-42	21	0
	60	19-18-17-16-15-14-73-72-53-60	93.6	0
	9	19-18-16-15-14-73-72-53-54-8-9	77.1	0
	50	49-50	16.5	0
38	53	38-41-48-52-17-16-15-14-73-72-53	89.1	0
	62	38-41-47-50-62	63.5	0
	7	38-41-48-52-17-16-15-14-73-72-53-54-8-7	109.1	0
	29	38-37-35-32-31-29	103.5	0
	51	38-41-48-51	33	0
57	26	57-53-72-73-14-15-16-17-18-19-20-21-22-23-24-25-26	136.5	0
	11	57-53-72-73-14-13-11	116	0
	51	57-53-72-73-14-15-16-17-51	114.1	0
	28	57-53-72-73-14-15-16-17-52-49-50-27-28	152.1	0
	71	57-60-35-33-34-71	128.9	0

Algoritma Semut				
Awal	Tujuan	Lintasan	Jarak (km)	Running Time (s)
19	27	19-49-50-27	31.5	38
	42	19-49-45-42	21	34
	60	19-18-17-16-15-14-73-72-53-60	93.6	39
	9	19-18-16-15-14-73-72-53-54-8-9	77.1	39
	50	49-50	16.5	33
38	53	38-41-48-52-17-16-15-14-73-72-53	89.1	38
	62	38-41-47-50-62	63.5	43
	7	38-41-48-52-17-16-15-14-73-72-53-54-8-7	109.1	38
	29	38-37-35-32-31-29	103.5	34
	51	38-41-48-51	33	42
57	26	57-53-72-73-14-15-16-17-18-19-20-21-22-23-24-25-26	136.5	47
	11	57-53-72-73-14-13-11	116	39
	51	57-53-72-73-14-15-16-17-51	114.1	43
	28	57-59-53-72-73-14-15-16-17-18-19-49-50-27-28	154.1	40
	71	57-59-60-35-33-34-71	130.9	38

Dari hasil *running* program di atas, Dijkstra sangat tangguh pada *running time* program dan jitu pada hasil lintasan yang diperoleh dibanding Semut. Rata-rata *running time* Dijkstra adalah 0 detik sedangkan Semut memiliki rata-rata *running time* 38,81 detik. Semut memiliki kelebihan dibanding Dijkstra, yaitu memiliki lebih banyak alternatif lintasan dibanding Dijkstra. Semut mempunyai lintasan yang lebih bervariatif dikarenakan banyak semut yang bekerja disana dan setiap semut memiliki informasi lintasan yang dia lalui sedangkan Dijkstra hanya memiliki satu lintasan saja tidak terdapat alternatif lintasan. Untuk keefisienan waktu dan jarak Dijkstra lebih efisien dibanding Semut.

E. KESIMPULAN

Kesimpulan yang didapat dari penelitian ini adalah :

1. Implementasi Algoritma Dijkstra dan Algoritma Semut di dalam penelitian ini menghasilkan sebuah program aplikasi pencarian lintasan terpendek menggunakan Delphi 7. Program aplikasi ini dapat digunakan untuk mencari lintasan terpendek dalam jaringan daerah pariwisata dengan 2 algoritma berbeda. Delphi 7 berhasil digunakan untuk membuat program pencarian lintasan terpendek menggunakan Algoritma Dijkstra dan Algoritma Semut.
2. Dalam hal running time Algoritma Dijkstra mengungguli Algoritma Semut. Dijkstra rata-rata 0 detik sedangkan Semut rata-ratanya adalah 38,81 detik.
3. Hasil lintasan yang diperoleh Dijkstra selalu optimal dibandingkan Semut karena lintasan yang diperoleh Dijkstra mempunyai jarak lebih kecil. Akan tetapi Semut mempunyai alternatif lintasan yang lebih bervariatif dibanding Dijkstra.
4. Algoritma Dijkstra lebih efisien dibanding Algoritma Semut karena running time dan lintasan yang dihasilkan Algoritma Dijkstra lebih optimal.

DAFTAR PUSTAKA

Chartrand, Gary & Oellermann, Ortrud R. 1993. *Applied and Algorithmic Graph Theory*. McGraw-Hill, Inc.

Dorigo, M. dan Socha K. 2007. *An Introduction to Ant Colony Optimization*. Tech.Rep/IRIDIA/2006-010. Université Libre de Bruxelles. Belgium.

Fakhri. 2008. *Penerapan Algoritma Dijkstra Dalam Pencarian Solusi Maximum Flow Problem*. Bandung

Munir, R. 2005. *Matematika Diskrit Edisi ketiga.* Informatika. Bandung.

Mutakhiroh, I., Indrato, Hidayat, T., 2007. *Pencarian Jalur Terpendek Menggunakan Algoritma Semut.* Seminar Nasional Aplikasi Teknologi Informasi 2007. Yogyakarta.

Mutakhiroh, I., Saptono, F., Hasanah, N., dan Wiryanata, R. 2007. *Pemanfaatan Metode Heuristik Dalam Pencarian Jalur Terpendek Dengan Algoritma Semut Dan Algoritma Genetika*. Seminar Nasional Aplikasi Teknologi Informasi 2007. Yogyakarta.

Pangaribuan, F. 2006. *Perubahan Persepsi Pada Algoritma Dijkstra*. Bandung.

Pratama, Izzat. 2011. *Model Jaringan Transportasi Pariwisata di Pulau Lombok Menggunakan Algoritma Dijkstra.* Skripsi. FMIPA UNRAM. MAtaram