SIMULASI LINTASAN JALUR TERPENDEK ALGORITMA DIJKSTRA BERBASIS *EXTENSIBLE MARKUP LANGUAGE* (XML)

Aris Puji Widodo
Program Studi Ilmu Komputer Jurusan Matematika FMIPA UNDIP
Jl. Prof. H. Soedarto, S.H, Semarang 50275
arispw@ilkom.undip.ac.id

Abstract. An optimation is a classical problem to use determine decision making. Traditionally to solve an optimation problem, it is needed a precission and long time. The improvement of this problem, needs simulation software development, less time relatively, and the result more than manually calculated, cause error probability to much. This research concern to make a simulation software based on Dijkstra Algorithm, as tool to give informations on the optimation decision making.

Keywords: optimation, decision making, Dijkstra, and tool

1. PENDAHULUAN

Perkembangan teknologi informasi memberikan pengaruh diberbagai bidang, mulai dari dunia pendidikan, kesehatan, perdagangan, pemerintahan, dan sebagainya. Dimana secara keseluruhan digunakan untuk meningkatkan efisiensi, efektifitas, dan keakuratan untuk mendukung aktivitas-aktivitas bisnis. Proses pengambilan keputusan merupakan suatu aktivitas yang sulit, karena hal ini menyangkut resiko yang mungkin terjadi. Resiko dalam setiap aktivitas tidak mungkin untuk dihilangkan, tetapi hanya dapat diminimalkan untuk terjadinya resiko tersebut. Salah satu cara untuk meminimalkan resiko, dapat dengan cara melakukan simulasi terlebih dahulu sebelum melakukan aktivitas yang sebenarnya. Hasil simulasi tersebut dapat digunakan untuk memprediksi kemungkinan terjadinya resiko dan membantu dalam hal pengambilan suatu keputusan untuk melakukan aktivitas bisnis.

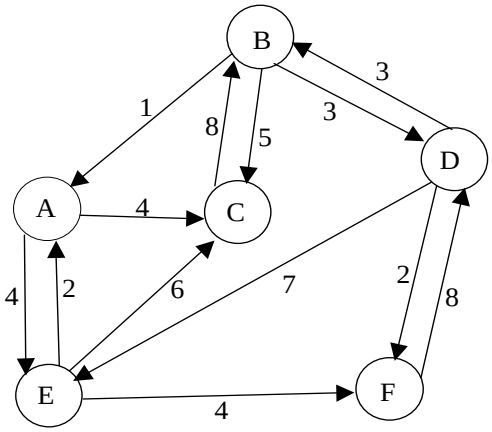
Algoritma Dijkstra merupakan metode yang paling efisien untuk menentukan lintasan jalur terpendek antara titik satu dengan titik lainnya [1]. Algoritma Dijkstra digunakan untuk menentukan *routing* pada jaringan komunikasi [6], *adaptive routing* untuk *mobile agent* [2]. Artikel ini membahas mengenai pengembangan perangkat lunak simulasi penentuan masalah lintasan jalur terpendek menggunakan algoritma Dijkstra. Aspek yang ditekankan pada artikel ini adalah membangun produk perangkat lunak simulasi. Produk perangkat lunak simulasi ini sebagai alat bantu yang memberikan informasi-informasi dalam penentuan keputusan untuk semua permasalahan yang berhubungan dengan optimasi dengan struktur yang *portable* dan terbebas dari berbagai variasi presentasi tertentu.

2. TINJAUAN PUSTAKA

2.1. Algoritma Dijkstra

Masalah penentuan jalur terpendek di dalam graph merupakan permasalahan optimasi klasik [1]. Graph yang digunakan adalah graph berarah dan memiliki suatu bobot. Bobot pada sisi graph dapat merepresentasikan jarak antar kota, waktu pengiriman, ongkos pembangunan dan sebagainya [3].

Graph *G(V,E)* terdiri dari *V* adalah himpunan titik dan *E* adalah himpunan garis [1,3]. Untuk representasi graph berbobot *G(V,E)* diberikan pada Gambar 1.



Gambar 1**.** Representasi Graph *G(V,E).*

Karakteristik algoritma Dijkstra [1,3] adalah sebagai berikut. Misalkan diberikan suatu graph berbobot *G(V,E)* dengan *n* buah titik dinyatakan dengan matrik ketetanggaan *M*=[m_{ij}], yang dalam hal ini

m_{ij} = bobot sisi (*i*,*j*).

m_{ii} = 0.

m_{ij} = ∞, jika tidak ada sisi dari titik *i* ke *j*.

Selain matrik *M*, juga digunakan matriks *S* = [s_i], yang dalam hal ini

s_i = 1, jika titik i termasuk ke dalam lintasan terpendek.

s_i = 0, jika titik i tidak termasuk ke dalam lintasan terpendek,

dan matrik *D* = [d_i], yang dalam hal ini

d_i = panjang lintasan dari titik awal *a* ke titik *i*.

Adapun konsep dasar algoritma Dijkstra untuk mencari lintasan terpendek dari titik awal *a* ke semua titik lainnya [1,3] adalah sebagai berikut.

Langkah 0 (inisialisasi).

- inisialisasi s_i = 0, dan $d_i = m_{ai}$, untuk i = 1, 2, 3, . . . n.

Langkah 1.

- isi s_a dengan 1 (karena titik a adalah titik asal lintasan terpendek, jadi sudah pasti terpilih).
- isi d_a dengan ∞ (tidak ada lintasan terpendek dari *a* ke *a*).

Langkah 2, 3, . . ., n-1.

- cari j sedemikian sehingga s_j = 0 dan $d_j = \min\{d_1, d_2, ..., d_n\}$.
- isi s_j dengan 1.
- perbaharui d_i, untuk i = 1, 2, 3, . . . n dengan d_i (baru) = min{ d_i (lama), $d_j + m_{ji}$ }.

2.1. *Pseudocode* Algoritma Dijkstra

Untuk merealisasikan ke dalam bahasa pemrograman, maka dari konsep dasar yang diberikan diatas diterjemahkan ke dalam notasi algoritmik. Notasi algoritmik untuk algoritma Dijkstra direpresentasikan ke dalam *pseudocode*, yang diberikan pada Kode Sumber 1 [4].

```
{d(i) adalah label jarak titik i}
{S himpunan permanen label titik}
{S' adalah himpunan temporary label titik}
{a adalah titik awal}
{mij adalah bobot pada garis (i,j)}
{D adalah lintasan terpendek pada G(V,E)}
{pred(j) = i if (i,j)∈ D}
{A(i) adalah garis adjacent titik i}
▪ Begin
    • S:=∅; S':=N;
    • d(i):=∞ untuk setiap titik i∈N;
    • d(s):=0 dan pred(s):=0;
    • while /S/<n do
    • begin
        * i ∈S' adalah titik, dimana d(i)=min{d(j):j ∈S'};
        * S:=S∪{I};
        * S':=S'-{I};
        * for each (I,j) ∈A(I) do
            ❖ If d(j) > d(i)+mij then d(j):=d(i)+mij and pred(j):=i;
    • end
▪ End.
```

Kode Sumber 1. *Pseudocode* Algoritma Dijkstra.

3. HASIL DAN PEMBAHASAN

3.1. Kebutuhan Fungsional

Kebutuhan fungsional perangkat lunak simulasi yang dikembangkan adalah sebagai berikut.

1. Titik, garis, dan bobot yang digunakan untuk merepresentasikan graph bersifat dinamik (dapat dilakukan penambahan dan penghapusan)
2. Lintasan terpendek dari titik sumber ke sumua titik (titik tujuan) diberikan dengan warna yang berbeda sebagai indikasi lintasan terpendek
3. Titik sumber dapat diubah-ubah sesuai dengan kebutuhan, demikian juga lintasan terpendeknya juga mengalami perubahan mengikuti titik sumber tersebut.
4. Graph yang dihasilkan dapat disimpan dan dapat digunakan kembali pada waktu yang berbeda.

3.2. Representasi Struktur Graph

Titik dan garis pada suatu graph direpresentasikan menggunakan struktur XML dengan tujuan untuk membentuk suatu struktur yang terbebas dari berbagai presentasi tertentu [7].

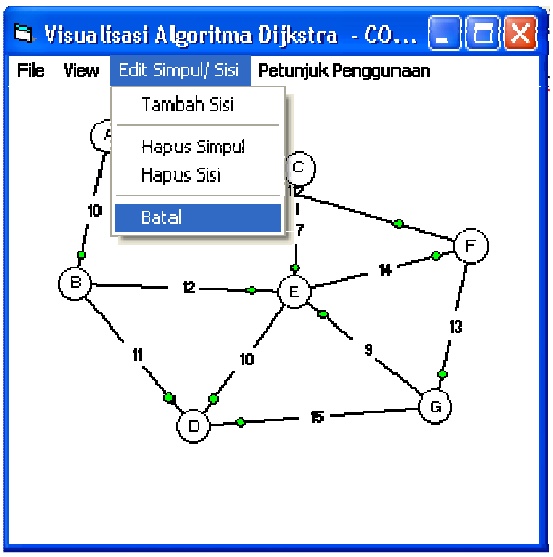
```
- <Web Height="6390" Width="6450">
  - <Node>
      <X>2115</X>
      <Y>2295</Y>
      <ID>A</ID>
    </Node>
  - <Node>
      <X>4290</X>
      <Y>1440</Y>
      <ID>B</ID>
    </Node>
  + <Node>
  + <Node>
  + <Node>
  + <Link>
  + <Link>
  + <Link>
  - <Link>
      <Nd1>A</Nd1>
      <Nd2>B</Nd2>
      <Cost>10</Cost>
      <ID>A-B</ID>
    </Link>
  + <Link>
  </Web>
```

Gambar 2. Struktur Graph Menggunakan XML

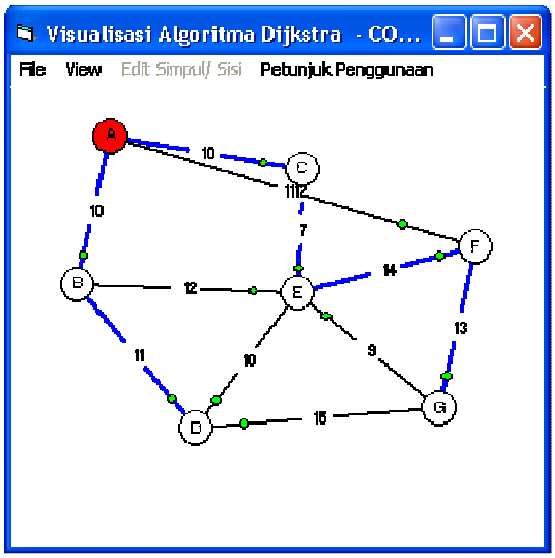
Setiap titik (<Node>) direpresentasikan dengan menggunakan 3 elemen, yaitu: <X> sebagai posisi koordinat terhadap sumbu x, <Y> sebagai posisi koordinat terhadap sumbu y, dan <ID> sebagai label titik. Untuk garis (<Link>) direpresentasikan dengan menggunakan 4 elemen, yaitu: <Nd1> sebagai titik sumber, <Nd2> sebagai titik tujuan, <Cost> sebagai bobot pada garis, dan <ID> sebagai label suatu garis, seperti yang diberikan pada Gambar 2.

3.3. Hasil Perangkat Lunak Simulasi

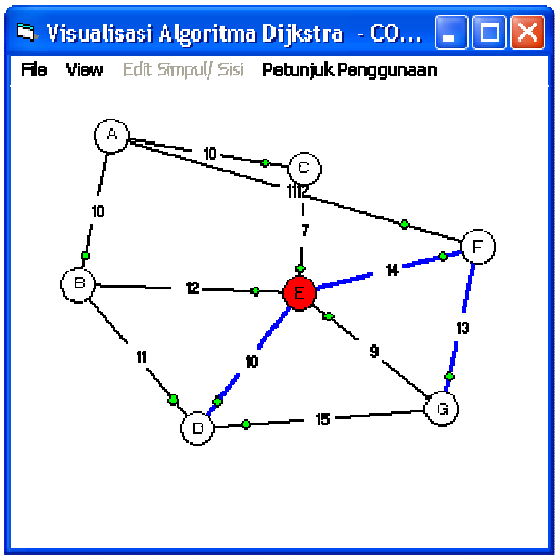
Penyusunan titik, garis, dan bobot untuk membentuk suatu graph diberikan pada Gambar 3, dimana setiap elemen titik dan garisnya dapat dilakukan perubahan, baik di tambah maupun di hapus. Gambar 4 adalah representasi lintasan terpendek graph dari titik sumber A ke semua titik dengan lintasan terpendeknya diberi warna biru. Untuk Gambar 5 adalah representasi lintasan terpendek graph dari titik sumber E ke semua titik. Titik sumber dapat dilakukan perubahan dengan melakukan klik pada titik yang diinginkan sebagai titik sumber.



Gambar 3. Pembentukan Graph dengan Titik dan Garis dapat diubah-ubah.



Gambar 4. Lintasan Terpendek Graph dengan Titik Sumber **A**



Gambar 5. Lintasan Terpendek Graph dengan Titik Sumber E

4. KESIMPULAN

Kesimpulan yang dapat diperoleh dari pengembangan perangkat lunak simulasi algoritma Dijkstra adalah sebagai berikut.

1. Struktur XML yang digunakan untuk merepresentasikan graph tidak bergantung pada satu platform dan presentasi tertentu, sehingga memberikan suatu struktur yang lebih bersifat dinamis.
2. Perangkat lunak simulasi yang dihasilkan, digunakan sebagai alat bantu yang memberikan informasi-informasi dalam penentuan keputusan untuk semua permasalahan yang berhubungan dengan optimasi. Dengan perangkat lunak simulasi ini diharapkan dapat melakukan estimasi terhadap kemungkinan resiko yang akan terjadi, sehingga keputusan bisnis yang diambil dapat seminimal mungkin memiliki resiko.

5. DAFTAR PUSTAKA

[1]. Ahuja, Ravinda K., Thomas L. Magnanti, James B. Orlin. (1993), *Network Flows*, Prentice Hall.

[2]. Di Caro, G. and M. Dorigo**.** (1997), *AntNet: A Mobile Agents Approach to Adaptive Routing*, Tech. Rep. IRIDIA/97-12, Université Libre de Bruxelles,Belgium,. *ftp://iridia.ulb.ac.be/pub/dorigo/tec.reps/TR.06.AntNet-TecRep-97-12.ps.gz*.

[3]. Dijkstra, E. W. (1959)**,** *A Note on Two Problems in Connection with Graphs*, Numer. Math, **1**: 269-271.

[4]. Morris, John. (1998), *Data Structures and Algorithms*.

[5]. Pressman, R.(1997), *Software Engineering A Practitioner's Approach* fourth edition, McGra-Hill.

[6]. Steenstrup, M., editor**.** (1995), *Routing in Communications Networks*,135-157, Prentice Hall.

[7]. Widodo. **(**2003), *Transformasi Dokumen XML*, Jurnal Matematika dan Komputer, UNDIP Semarang, **6**(3).