Perancangan Aplikasi Pencarian Jalur Terpendek Untuk Lokasi Toko Bangunan Di Kota Bogor Dengan Metode A* *(A-Star)* Berbasis Android

Erniyati M.Kom
Program Studi Ilmu Komputer FMIPA Universitas Pakuan Bogor
Email : neni_erniyati@yahoo.com

ABSTRACT

Abstract
Distribution is one of the most important thing in the business, especially delivery of goods by land transportation. Determination of traffic path plays a critical role in the goods distribution. The best transportation track path will help the business to minimize the shipping cost as well as a more efficient delivery time. A designated algorithm or system is required to search for the shortest and the most optimal traffic path from various numbers of existing alternative pathways. Case study was carried out at PT. Tulu Atas; the company that their business activities is controlled by land transportation. A * algorithm is one pathfinding algorithm that uses the shortest distance estimation to achieve the goal and has a heuristic value that is used as the basis for consideration. A mobile application is developed and implemented on android operating system based on this study to determine the shortest traffic path.

Keyword *: the shortest path problem, A*(A-Star) algorithm, android operating system*

1. PENDAHULUAN

Distribusi merupakan salah satu hal yang sangat penting dalam suatu bidang usaha terutama dalam melakukan pengiriman barang melalui jalan darat. Salah satu kendala yang dihadapi dalam pendistribusi barang adalah penentuan jalur lintasannya. Penentuan jalur lintasan yang terbaik dapat meminimalkan biaya pengiriman dan waktunya lebih efisien. Salah satu cara penentuan jalur lintasan terbaik adalah dengan mencari jarak terpendek pada jalur distribusinya. Jarak terpendek dapat dihitung dengan menggunakan pendekatan teori graf.

Menurut teori graf, persoalan lintasan terpendek adalah untuk mencari lintasan antara dua buah simpul pada graf berbobot yang memiliki gabungan jumlah nilai bobot pada sisi graf yang dilalui dengan jumlah paling minimum. Persoalan lintasan terpendek paling banyak ditemui yaitu dalam bidang transportasi, seperti pada pencarian rute terpendek untuk menempuh dua kota [1].

Algoritma A* (*A-Star*) merupakan salah satu jenis algoritma yang digunakan untuk menyelesaikan kasus yang berhubungan dengan path finding (pencarian jalan). Dalam hasil pencariannya A* dikatakan complete dan optimal. Algoritma A* menggunakan cara pencarian secara *Best First Search*, dimana pencarian dilakukan dengan cara melebar ke setiap node pada level yang sama, dan nantinya akan menemukan rute terbaik dari titik awal sampai tujuan. Algoritma A* juga dilengkapi suatu fungsi heuristik. Fungsi heuristik yang terdapat pada algoritma A* digunakan sebagai optimasi dalam menentukan node tujuan [2].

Kota Bogor merupakan salah satu kota yang memiliki lintasan atau jalur yang sangat banyak, sehingga untuk melakukan pendistribusian barang membutuhkan waktu dan biaya yang banyak. Untuk itu, dalam proses pencarian jalur terpendek diperlukan suatu sistem yang dapat membantu dalam mencari dan menentukan lintasan terpendek sehingga didapat suatu jalur yang paling optimal. Sistem ini selanjutnya dihubungkan dengan perangkat mobile yang berbasis android.

2. METODE PENELITIAN

2.1 Lintasan Terpendek (*Shortest Path*)

Dalam Jurnal Pawitri (2007) disebutkan bahwa lintasan terpendek (*shortest path*) merupakan lintasan minimum yang diperlukan untuk mencapai suatu titik dari titik tertentu. Dalam pencarian lintasan terpendek masalah yang dihadapi adalah mencari lintasan mana

yang akan dilalui sehingga didapat lintasan yang paling pendek dari satu verteks ke verteks yang lain.

Ada beberapa macam persoalan lintasan terpendek, antara lain :

- Lintasan terpendek antara dua buah vertex (a pair shortest path).
- Lintasan terpendek antara semua pasangan vertex (all pair shortest path).
- Lintasan terpendek dari verteks tertentu ke semua verteks yang lain (single source shortest path).
- Lintasan terpendek antara dua buah verteks yang melalui beberapa verteks tertentu (intermediate shortest path).

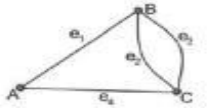
2.2 Graf

2.2.1 Definisi Graf

Graf adalah kumpulan simpul (nodes) yang dihubungkan satu sama lain melalui sisi/busur (edges) [4]. Suatu graf G terdiri dari dua himpunan yaitu himpunan V dan himpunan E.

a. *Verteks* (simpul) V = himpunan simpul yang terbatas dan tidak kosong.
b. Edge (sisi/busur) E = himpunan busur yang menghubungkan sepasang simpul.

Dapat dikatakan graf kumpulan dari simpul-simpul yang dihubungkan oleh sisi-sisi.

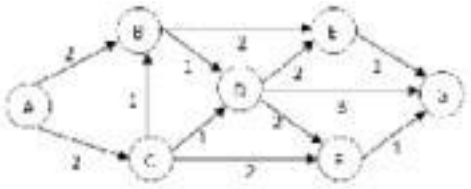


Gambar 2.1 Graf

2.2.2 Macam-macam Graf

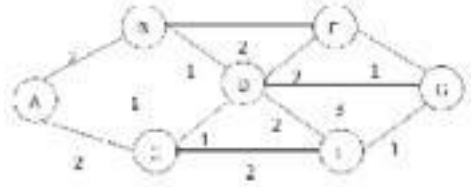
Menurut arah dan bobotnya, graf dibagi menjadi empat bagian, yaitu :

1. Graf berarah dan berbobot yaitu tiap busur mempunyai anak panah dan bobot. Gambar 2.2 menunjukan graf berarah dan berbobot yang terdiri dari tujuh titik yaitu titik A,B,C,D,E,F,G. Titik menunjukan arah ke titik B dan titik C, titik B menunjukan arah ke titik D dan titik C, dan seterusya. Bobot antar titik A dan titik B pun telah di ketahui.



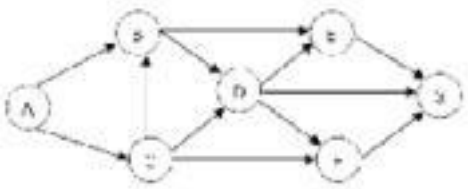
Gambar 2.2 Graf berarah dan berbobot

2. Graf tidak berarah dan berbobot yaitu tiap busur tidak mempunyai anak panah tetapi mempunyai bobot. Gambar 2.3 menunjukan graf tidak berarah dan berbobot. Graf terdiri dari tujuh titik yaitu titik A,B,C,D,E,F,G. Titik A tidak menunjukan arah ke titik B atau C, namun bobot antara titik A dan titik B telah diketahui. Begitu juga dengan titik yang lain.



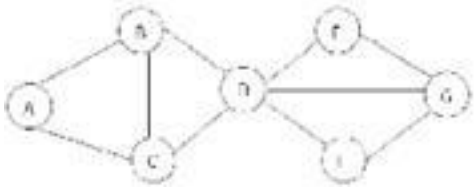
Gambar 2.3 Graf tidak berarah dan berbobot

3. Graf berarah dan tidak berbobot yaitu tiap busur mempunyai anak panah yang tidak berbobot. Gambar 2.4 menunjukan graf berarah dan tidak berbobot.



Gambar 2.4 Graf berarah dan tidak berbobot

4. Graf tidak berarah dan tidak berbobot yaitu tiap busur tidak mempunyai anak panah dan tidak berbobot.



Gambar 2.5 Graf tidak berarah dan tidak berbobot

2.3 Algoritma *Pathfinding*

Tujuan dari algoritma *pathfinding* adalah untuk menemukan jalur terbaik dari *verteks* awal ke *verteks* akhir. Secara umum algoritma *pathfinding* digolongkan menjadi dua jenis (Russel, dkk, 1995), yaitu :

1. *Algoritma Uniformed Search* adalah algoritma yang tidak memiliki keterangan tentang jarak atau biaya dari *path* dan tidak memiliki pertimbangan akan *path* mana yang lebih baik. Yang termasuk dalam algoritma ini adalah algoritma *Breadth First Search.*
2. *Algoritma Informed Search* adalah algoritma yang memiliki keterangan tentang jarak atau biaya dari *path* dan memiliki pertimbangan berdasarkan pengetahuan akan *path* mana yang lebih baik. Yang termasuk algoritma ini adalah algoritma A*.

2.4 Algoritma A* *(A-Star)*

Algoritma ini pertama kali ditemukan pada tahun 1968 oleh Peter Hart, Nils Nilsson dan Bertram Raphael [5]. Dalam tulisan mereka, algoritma ini dinamakan algoritma A. Penggunaan algoritma ini dengan fungsi heuristik yang tepat dapat memberikan hasil yang optimal, maka algoritma ini pun disebut A*.

Pencarian menggunakan algoritma A* mempunyai prinsip yang sama dengan algoritma algoritma BFS, hanya saja dengan 2 faktor tambahan [6].

- Setiap sisi mempunyai “cost” yang berbeda-beda, seberapa besar cost untuk pergi dari satu simpul ke simpul yang lain.
- Cost dari setiap simpul ke simpul tujuan bisa diperkirakan. Ini membantu pencarian sehingga lebih kecil kemungkinan mencari kearah yang salah.

A * adalah *generic search algorithm* yang dapat digunakan untuk mencari solusi untuk banyak masalah, salah satunya adalah pathfinding [7]. *A** adalah sebuah *graph* atau metode pohon pencarian yang digunakan untuk mencari jalan dari sebuah node awal ke node tujuan (*goal node*) yang telah ditentukan. Beberapa terminologi dasar yang terdapat pada algoritma ini adalah starting point, simpul (*nodes*), A, open list, closed list, harga (*cost*), halangan (*unwalkable*). Pada kondisi yang tepat, Algoritma A* akan memberikan solusi yang terbaik dalam waktu yang optimal [2]. Algoritma ini memeriksa node dengan menggabungkan *g(n),* yaitu cost yang dibutuhkan untuk mencapai sebuah node dan *h(n)* yaitu cost yang didapat dari node ke tujuan (Russel, 2003). Sehingga dapat dirumuskan sebagai berikut :

$$f(n) = g(n) + h(n)$$

f(n) = perkiraan total *cost* terendah dari setiap *path* yang akan dilalui dari *node* n ke *node* tujuan.

g(n) = biaya yang sudah dikeluarkan dari keadaan awal sampai keadaan *n*

h(n) = perkiraan heuristik atau *cost* atau *path* dari *node* n ke tujuan yang diperoleh dari pencarian nilai jarak Euclid menggunakan koordinat setiap node.

Untuk menentukan nilai *h(n)* ditunjukkan oleh persamaan :

$$h(n) = \sqrt{(Xn - Xgoal)^2 + (Yn - Ygoal)^2}$$

h(n) = nilai untuk *node/* titik *n Xn* = nilai X dari *node n*
Yn = nilai Y dari *node n*
Xgoal = nilai X dari *node* tujuan *Ygoal* = nilai Y dari *node* tujuan

Adapun langkah-langkah dalam pencarian jalur terpendek dengan algoritma A* *(A-Star)* ditunjukkan pada *pseudocode* sebagai berikut,

1. Add the starting node to the open list.
2. Repeat the following steps :
   a. Look for the open list. Refer to this node as the current node.
   b. Switch it to the closed list.
   c. For each reachable node from the current node
      i. If it is on the closed list, ignore it.
      ii. If it isn't on the open list, add it to the open list. Make the current node the parent of this node. Record the f,g, and h value of this node.
      iii. If it is on the open list already, check to see if this is a better puth. If so change its parent to the current node, and recalculate the f and g value.
   d. Stop when
      i. Add the target node to the closed list.
      ii. Fail to find the target node, and the open list is empty.
3. Tracking backwards from the target node to the starting node. That is your path.

Gambar 2.6 *Pseudocode* A* *(A-Star)*
(N.Nilsson 1998)

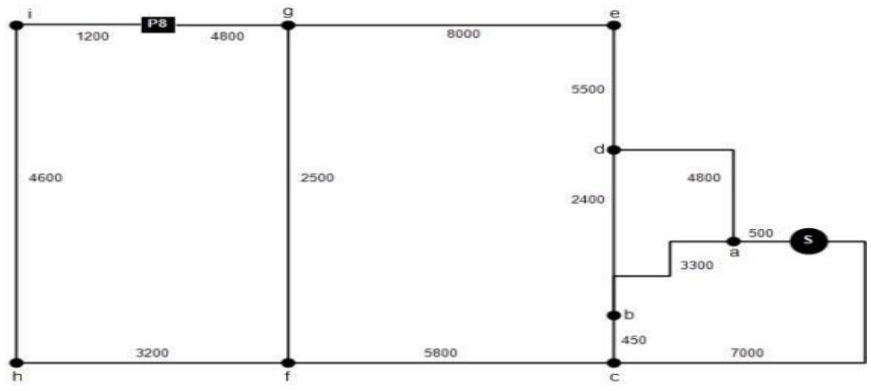
3. HASIL DAN PEMBAHASAN

Adapun *software/tools* yang digunakan dalam pembuatan aplikasi pencarian jalur terpendek ini adalah *Android Studio* : Membuat koding program, *Google Maps* : Menentukan jarak sebenarnya antara dua titik yang sebenarnya, http://twcc.fr/# : Konversi koordinat latitude longitude ke decimal, *Adobe Photoshop* : Desain aplikasi, *Genymotion* : Emulator program.

3.1 Analisis Masalah

Algoritma A* (*A-Star*) merupakan suatu algoritma yang termasuk pada kategori metode pencarian yang memiliki informasi (*informed search method*). Algoritma A* menggunakan estimasi jarak terdekat *(cost/* jarak sebenarnya) untuk mencapai tujuan (*goal*) dan memiliki nilai heuristik yang digunakan sebagai dasar pertimbangan pemilihan jalur.

Adapun analisis algoritma A* yang akan dicari jalur terpendek antar titik awal S menuju titik tujuan P8 terdapat pada gambar 3.1



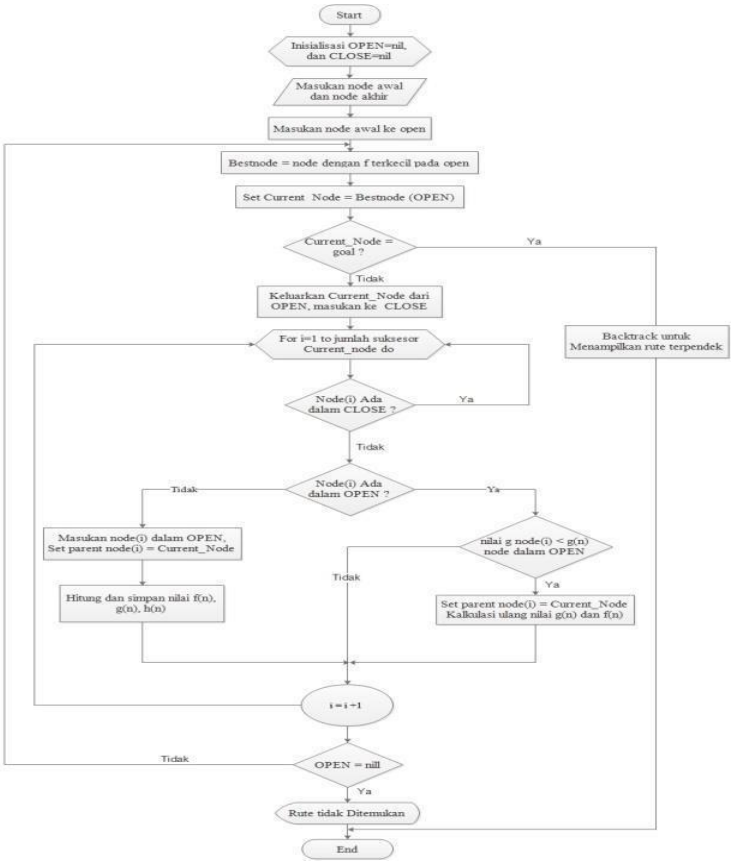
Gambar 3.1 Analisis algoritma A*

Untuk menyelesaikan permasalah pencarian jalur terpendek dengan menggunakan algoritma A* ditunjukkan pada langkah-langkah *flowchart* pada gambar 3.2.

3.2 Penentuan *cost* antara dua titik yang berhubungan

Untuk menentukan *cost* antara dua titik yang berhubungan, maka akan dicari dengan menggunakan *google maps direction* untuk mengetahui jarak dari titik s ke n. Hasil dari pengukuran nilai *cost* antara dua *node* yang saling berhubungan adalah *costnode* s ke *node* a adalah 500m, *costnode* s ke *node* c adalah 7000m, *costnode* a ke *node* b adalah 3300m, *costnode* a ke *node* d adalah 4800m, *costnode* b ke *node* c adalah 450m, *costnode* c ke *node*

f adalah 5800m, *costnode* d ke *node* e adalah 5500m, *costnode* f ke *node* g adalah 2500m, *costnode* g ke *node* h adalah 3200m, *costnode* e ke *node* g adalah 8000m, *costnode* g ke *node* p8 adalah 4800m, *costnode* i ke *node* p8 adalah 1200m.



Gambar 3.2 *Flowchart A**

3.3 Penentuan koordinat setiap *node*

Untuk menentukan titik koordinat suatu *node,* pada *google maps* dilakukan pengambilan koordinat antara dua *node* yang saling berhubungan. Setelah itu nilai dari setiap *node* dikonversi kedalam bentuk desimal dengan menggunakan *software online* dengan alamat *http://twcc.fr/#.*

Adapun hasil penentuan titik koordinat setiap *node* adalah sebagai berikut :

a. *node* s (708291.12, 9286386.4)
b. *node* a (708196.93, 9285927.06)
c. *node* b (706725.3, 9283118.76)
d. *node* c (707062.36, 9282870.09)
e. *node* d (705532.03, 9284685.68)
f. *node* e (705114.79, 9285067.69)
g. *node* f (704778.44, 9277627.68)
h. *node* g (702930.89, 9279206.15)
i. *node* h (703898.87, 9274542.7)
j. *node* i (700118.25, 9274104.33)
k. *node* p8 (700759.38, 9275091.92)

3.4 Penentuan Koordinat setiap *node*

Setelah mendapatkan nilai *cost*/biaya antara dua *node* yang berhubungan dan titik koordinat setiap *node*/titik, kemudian lakukan penghitungan nilai *f(n)* dengan langkah-langkah sebagai berikut :

Langkah 1

- Set OPEN = {}

- Set CLOSED = {}
- *Node* Awal = S
- *Node* Tujuan = p8

Langkah 2
- Masukan *node* awal ke *OPEN LIST*
- *OPEN* = {S}
- *Set current_node* = {S)

Langkah 3
- Periksa apakah *current_node* adalah *node* tujuan
- *Current_node (node S)* bukan *node* tujuan
- Karena *node* S bukan *node* tujuan, maka keluarkan *node* S dari *OPEN* dan pindahkan ke *CLOSED*
- *OPEN* = {}
- *CLOSED* = {S}

Langkah 4
- Hitung jumlah jalur yang bisa dilalui dari *current_node* (*neighbournode* S)
- Jumlah jalur yang bisa dilalui berjumlah 2 jalur yaitu *node* A dan C
- Iterasi dilakukan pada *node* A
- Periksa apakah *node* A ada di *OPEN* atau di *CLOSED* atau tidak berada di *OPEN* maupun *CLOSED*
- *Node* A tidak berada di *OPEN* maupun di *CLOSED*, sehingga *node* A dimasukan ke *OPEN*
- *OPEN* = {A}
- Hitung nilai *g*(A), *h*(A) dan *f*(A) *g*(A) = *costnode* S ke *node* A *g*(A) = 500 *meter*
  h(A) = 13142 *f*(A) = *g*(A) + *h*(A) *f*(A) = 500+13142
  f(A) = 13692
- Nilai *f*(A) adalah 13.692
- Lakukan perintah seperti iterasi 1
- *OPEN* = (C)
- Hitung nilai *g*(C), *h*(C), *f*(C) *g*(C) = 7000 *meter*
  h(C) = 10011 *f*(C) = *g*(C) + *h*(C)
  f(C) = 7000 + 10011 *f*(C) = 17011
- Nilai *f*(C) adalah 17.011

Langkah 5
- *OPEN* = {A,C}
- Bandingkan nilai *f*{A} dengan nilai *f*{C}. nilai *f*(A) adalah yang paling kecil (13.692)
- *Current_node* = {A}

Langkah 6
- *Current_note* (*node* A) bukan *node* tujuan
- Pindahkan *node* A dari *OPEN* ke *CLOSED*
- *OPEN* = {C}
- *CLOSED* = {S,A}

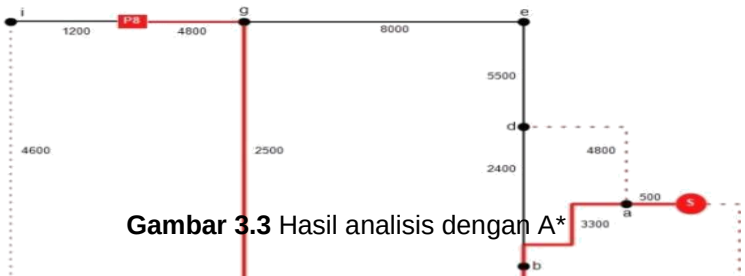
Langkah 7
- *OPEN* = {C}
- *CLOSED* = {S,A}
- Ulangi langkah-langkah diatas pada *node-node* lain dengan menggunakan aturan yang ada pada *flowchart* sampai didapatkan jalur terpendeknya.

Adapun hasil pencarian jalur terpendek menggunakan algoritma A* berdasarkan perhitungan langkah-langkah diatas ditunjukkan pada gambar 3.3. Adapun implementasi pada *smartphone* adalah sebagai berikut :

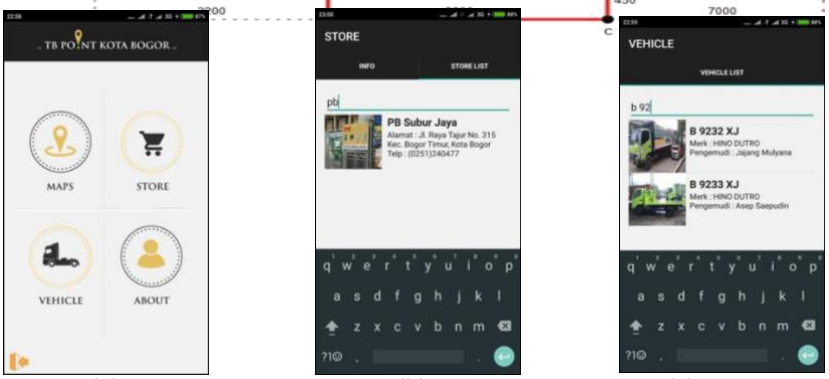
a. *layout* pada smartphone

Pada menu utama dapat dilihat pada gambar 3.4 (a), halaman ini *user* memilih menu yang akan dibuka sesuai kebutuhannya, terdapat 5 menu, yaitu : maps, store, vehicle, about dan keluar. Pada layout store list dapat dilihat pada gambar 3.4 (b), halaman ini

menampilkan informasi alamat, foto nama toko dan info toko tersebut. Pada layout vehicle list dapat dilihat pada gambar 3.4 (c), halaman ini menampilkan informasi kendaraan, nomor kendaraan, kondisi kendaraan dan nama pengemudi kendaraan tersebut.



Gambar 3.3 Hasil analisis dengan A*

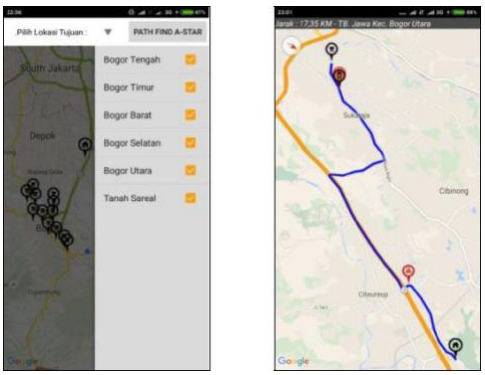


(a) (b) (c)

Gambar 3.4 Implementasi pada smartphone
(a) menu utama (b) store list (c) vehicle list

b. *layout maps*

Halaman pencarian terletak pada bagian menu *maps*, untuk melihat lokasi toko berdasarkan kecamatan geser layar dari kanan ke tengah kemudian klik salah satu untuk menampilkan pilihan lokasi. pada pilihan tujuan lokasi pilih lokasi terlebih dahulu kemudian setelah menentukan lokasi klik tombol path find a-star maka akan menampilkan rute dari PT. Tulu Atas ke lokasi toko tujuan.



Gambar 3.7 maps

c. *Pengujian Pemilihan Jalur Lintasan dengan membandingkan Algoritma A* dengan Google Map API.*

Dari hasil pengujian diperoleh nilai untuk algoritma A* adalah 17.350m dan untuk google map adalah 20.000m, ilustrasinya dapat dilihat pada tabel 1.

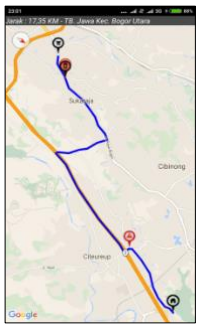
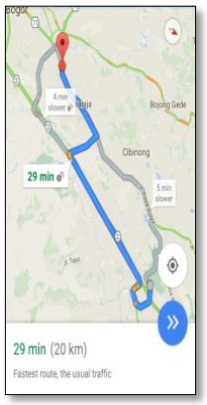
4. KESIMPULAN

Aplikasi tbPointkotaBogor merupakan aplikasi pencarian titik lokasi toko bangunan yang ada di kota Bogor, yang bertujuan mempermudah dalam proses pengiriman semen bagi perusahaan ekspedisi PT.Tulu Atas kranggan Gunung Putri Bogor.

Kesimpulan yang dapat diambil adalah sebagai berikut :

1. Untuk menentukan jalur terpendek pada pencarian dengan menggunakan algoritma A* dibutuhkan beberapa data yaitu, jarak sebenarnya antara *node* yang berhubungan *(cost* antara *node*) dan koordinat setiap *node*. Dengan data tersebut, maka pencarian jalur terpendek dapat di implementasikan.
2. Dari hasil perbandingan algoritma A* dan Google Map terlihat bahwa algoritma A* lebih mempunyai jarak tempuh paling pendek dibandingkan dengan Google Map, karena Google Map selalu memilih akses tol walaupun jarak yang ditempuh lebih jauh dibandingkan jalur bukan tol.

Tabel 1. Perbandingan Algoritma A* dengan Google Map API

Pengujian A*	Pengujian *google Map API*
	

References

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

[2] Russell, Struart.& Norvig,Peter. 2003."*Artificial Intelligence:A Modern Approach*", New Jersey: Prentice Hall.

[3] Pawitri, K., Ayu, Y. Dan Joko, P., 2007 *Implementasi Algoritma A* untuk menemukan Lintasan Terpendek, http://*journal.amikom.ac.id/index/php/SN/article/view/2075, diakses tanggal 23 Februari, 20:00 WIB.

[4] Zakaria., 2006, *Teknologi Informasi dan Komunikasi*, Arya Duta, Jakarta.

[5] N.Nilsson, Artificial Intelligence:A New Synthesis, Morgan Kaufmann Publishers, San Francisco, 1998.

[6] Siti Rachmi Wulandari, Yudha Purwanto dan Budhi Irawan., 2012,Seminar nasional aplikasi Teknologi Informasi (SNATI 2012), Yogyakarta, 15-16 Juni 2012. ISSN:1907-5022.

[7] Xiao Cui dan Ho Shi., 2011, A* based Pathfinding in Modern Computer Games, IJCSNS International Journal of Computer Science and Network Security, Vol 11 No. 1. January 2011