Techno.COM, Vol. 16, No. 2, Mei 2017 : 182-194

Penemu Jalur Optimal Untuk Rute Jalan Dengan *New Bidirectional A** Di Semarang

*Optimal Path Finder For Road Route Using New Bidirectional A * In Semarang*

Ardian Fajar Rahmanto*[1], Wijanarto[2]
[1,2]Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Dian Nuswantoro;
Jl.Imam Bonjol no.207, Semarang, 50131, (024)3517261
e-mail: *[1] fajarrahmanto@gmail.com , [2]wijanarto@dsn.dinus.ac.id

Abstrak

Lalulintas jalan merupakan masalah yang terjadi hampir di seluruh kota besar di dunia, terutama mengenai kemacetan. *Untuk menangani permasalahan kemacetan dan menguraikannya merupakan tantangan tersendiri dan dengan alat bantu kemajuan teknologi informasi dan system navigasi, nampaknya hal tersebut menjadi terobosan baru. Pencarian rute jalan sehingga didapatkan rute yang optimum diharapkan dapat membantu mengatasi dan mengurai kemacetan jalan, namun demikian seringkali juga menjadi bias karena butuh waktu tempuh yang lebih lama. Sistem Informasi Geografis atau SIG merupakan salah satu sistem yang menunjang pengetahuan rute dari sebuah peta dan informasi suatu wilayah. Paper ini menyajikan solusi penemu atau pencari rute pada SIG dengan menerapkan teknik New Bidirectional A* atau NBA* pada SIG dapat melakukan komputasi penemu jalur optimal. Aplikasi yang dihasilkan dari teknik ini dapat menampilkan jalur optimal dari lokasi awal ke tujuan. Perbandingan hasil uji perjalanan secara langsung dengan komputasi yang di lakukan sebanyak 4 kali, teknik ini terbukti valid dan sinkron sebanyak 3 kali dan menghasilkan waktu tempuh optimal dengan nilai heuristic yang di tentukan secara statis. Kedepan penentuan fungsi heuristic secara statis perlu di buat dinamis sesuai dengan lokasi pencarian dari peta digital.*

Kata kunci— *Penemu Jalur, Algoritma New Bidirectional A*, Sistem Informasi Geografis*

Abstract

*Road traffic is a problem that occurs almost in all major cities in the world, especially about congestion. To deal with congestion problems and describe them is a challenge and with the help of advanced information technology and navigation system, it seems to be a new breakthrough. Searching for road routes to get an optimum route is expected to help cope with and break down road congestion, but it is often also biased because it takes longer time to travel. Geographic Information System or GIS is one of the systems that support the route knowledge of a map and information of a region. This paper presents the inventor or route finder solution on the GIS by applying New Bidirectional A * or NBA * techniques on GIS to compute the inventor of the optimal path. Applications generated from this technique can display the optimal path from start to destination. Comparison of direct trip test results with computation done 4 times, this technique proved valid and synchronous as much as 3 times and resulted in optimal travel time with heuristic value specified in static. Future determination of static heuristic functions needs to be made dynamically according to the search location of the digital map.*

Keywords—Path Finding, *New Bidirectional A**, Geographic Information Systems

1. PENDAHULUAN

Transportasi merupakan persoalan yang berarti bagi masyarakat. Luasnya wilayah kota dan banyaknya jalan terkadang menyusahkan seseorang untuk menentukan jalur yang optimum,

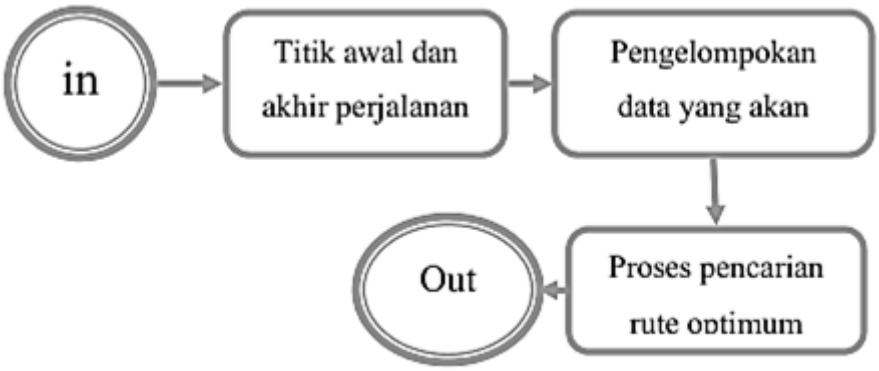
baik dari segi jarak maupun waktu yang di perlukan untuk mencapai lokasi tujuan. Hal ini di perparah dengan terjadinya kemacetan di berbagai tempat sehingga menyebabkan waktu tempuh menjadi lama [1]. Kemacetan lalu lintas tetap menjadi masalah besar di kota-kota besar terutama di negara berkembang yang mengakibatkan penundaan waktu, pemborosan bahan bakar, dan kerugian moneter. Jaringan jalan yang buruk adalah titik kemacetan yang parah [2]. Salah satu solusi yang di pilih adalah dengan menggunakan jalan alternatif, namun kurangnya informasi mengenai jalan alternatif menjadikan pilihan ini kurang efektif, sehingga di perlukan sebuah sistem yang akan memberikan informasi mengenai jaringan jalan yang benar. Sistem ini di kenal dengan sebutan Sistem Informasi Geografis (SIG) [5], yang merupakan aplikasi database spasial dengan user interface yang baik dan dibangun dalam pengolahan data, analisis dan fungsi layar. Suatu basis data spasial berisi data deskriptif dan kuantitatif seperti database biasa. Semua yang ada di database diposisikan di beberapa ruang referensi dan terkait topologi untuk item database lain berdasarkan posisi. Sedangkan untuk SIG, posisi relatif terhadap posisi yang sebenarnya pada permukaan bumi, dan mungkin sesuai dengan lintang atau bujur, koordinat peta, atau pengukuran lainnya [4]. Jan-Hendrik dan Benedikt Budig dari Universitas Wurzburg, dalam jurnal "*An Algorithm for Map Matching given Incomplete Road Data*" [6], dalam penelitiannya tersebut mereka membahas pencocokan lintasan GPS dengan satu set data jalan dimana beberapa ada jalan hilang. Untuk menentukan jalur yang di lewati maka di pilih salah satu jalur *off-road* untuk di lalui, dan melakukan percobaan menggunakan trek GPS pejalan kaki.

Penelitian lainnya di lakukan oleh Wim Pijls dan Henk Post dari Economic Institute Report, dalam penelitiannya yang berjudul "*Yet another bidirectional algorithm for shortest paths"* [7]. Dalam penelitiannya membahas menemukan jalur terpendek dalam sebuah jaringan dengan menggunakan metode *"bidirectional A*"*. *"Bidirectional A*"* di anggap paling tepat karena menggunakan perkiraan heuristic yang seimbang. Stéphanie Vanhove dan Veerle Fack dalam penelitian yang berjudul "*Applications of Graph Algorithms in GIS"* [8], membahas standar algoritma jalur terpendek, generasi rute alternatif adalah salah satunya. Penelitian tersebut bertujuan untuk mengevaluasi dan mengembangkan metode yang efisien untuk berbagai varian dari masalah routing.

Paper ini menggunakan metode penemu atau pencarian jarak yang termasuk dalam keluarga *heuristic search* yaitu New Bidirectional A* disingkat NBA*, karena kemampuannya yang hanya membebani satu kendala pada fungsi heuristic [3]. NBA* adalah algoritma pencarian baru yang hampir mirip dengan algoritma A*. Pada algoritma NBA* di asumsikan bahwa kode berjalan secara bersamaan pada dua sisi yang ada. Kedua sisi tersebut sama sama memiliki simpul *s* (awal) dan *t* (akhir) [7]. Pada proses awal melakukan pencarian, kita menggunakan graph asli dan fungsi jarak asli *D(u;v)*. Dalam proses alternatif kita menggunakan graph terbalik , dimana setiap garis asli *(u;v)* di ganti garis *(v;u)* dari jarak yang sama. Pada pencarian dua arah kita merujuk pada proses utama dan proses yang berlawanan. Pencarian dengan metode NBA* berhenti saat satu sisi berhenti [7]. Paper ini menyajikan solusi berupa aplikasi sederhana untun menemukan jalur jalan secara optimal dengan menggunakan teknik *New Bidirectional A** yang di aplikasikan pada SIG di wilayah kota Semarang.

2. METODE PENELITIAN

Metode utama yang dipakai dalam paper ini adalah menggunakan algoritma *New Bidirectional A*(NBA*)* dengan menggunakan kerangka berpikir sebagai berikut seperti pada gambar 1 di bawah,

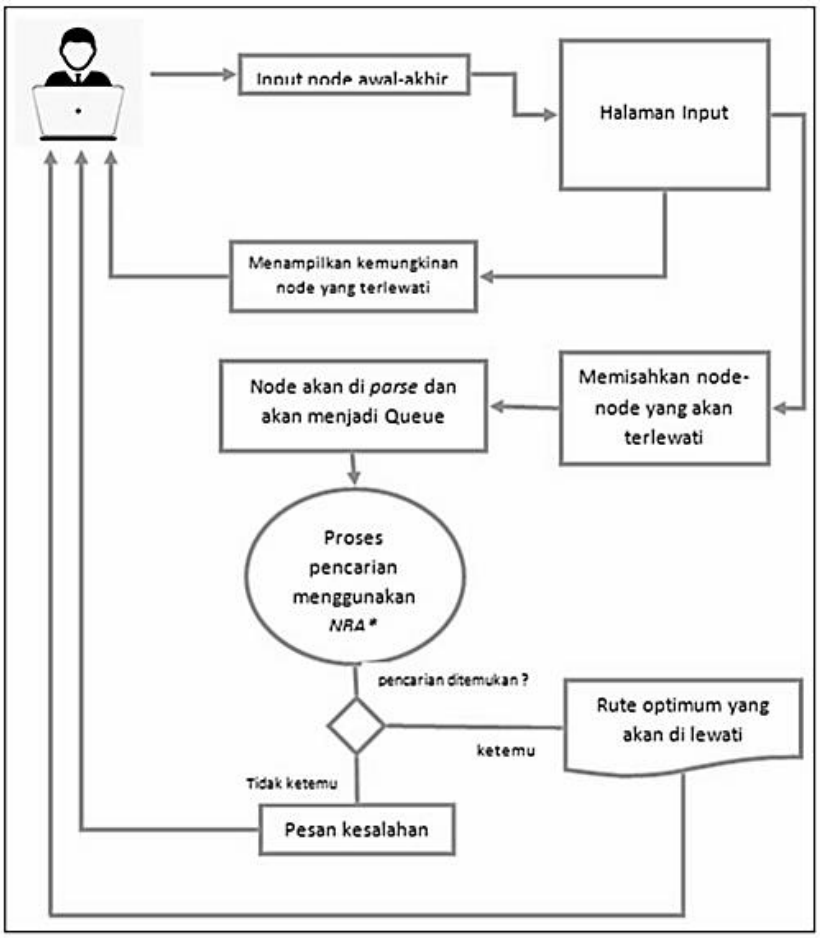


Gambar 1. Metode yang di usulkan

Seperti di tunjukan dalam gambar 1 di atas metode yang disulkan dalam paper ini dapat di jelaskan sebagai berikut :

a. Di masukkan titik awal dan akhir yang akan menjadi acuan untuk pencarian jarak.
b. Pengelompokan kemungkinan rute yang akan di lewati untuk proses pencarian.
c. Penerapan algoritma *New Bidirectional A*(NBA*)* untuk pencarian rute optimum.
d. Hasil output rute teroptimum dari proses sebelumnya.

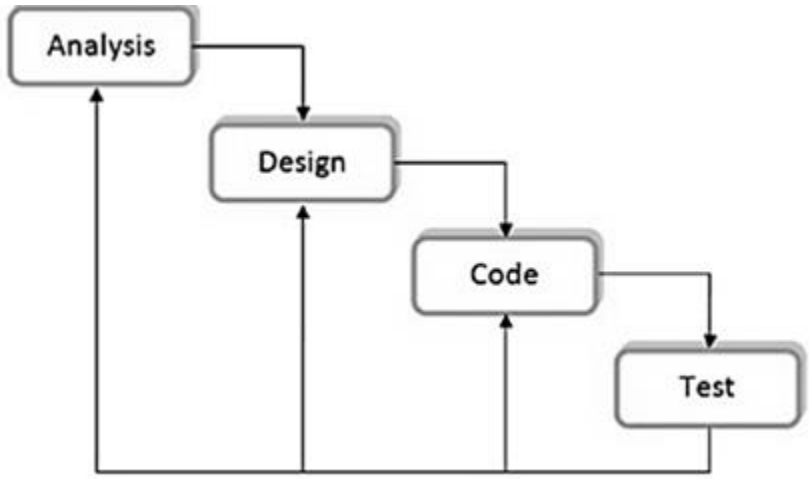
Sementara arsitektur yang akan di sajikan sebagai hasil dari penelitian ini adalah sebagai berikut pada gambar 2 di bawah ini:



Gambar 2. Arsitektur Aplikasi

Gambar 2 dapat di jelaskan sebagai berikut, setelah user menginputkan titk awal dan akhir pada halaman input, maka system akan menampilkan kemungkina jalur yang akan di lalui, pada saat yang bersamaan system juga memisahkan node dari rute yang di inputkan user untuk dilakukan parse dan di masukan ke queue. Dari data yang ada dalam queue ini akan dilakukan pencarian dengan NBA*, dan system akan memberkan informasi apakah jalur yang di cari ketemu atau tidak.

Sementara pengembangan system dilakukan dengan metode model *Waterfall*. Model ini termasuk model sekuensial linier yang sistematis, menggunakan pendekatan sekuensial untuk pengembangan perangkat lunak melalui analisis, desain, implementasi, pengujian, dan dukungan. Berikut gambar 3 adalah langkah-langkah pengembangan sistem model *Waterfall* menurut Roger S.pressman[14].



Gambar 3. Model Waterfall

a. Analysis

Pada tahap analysis, yang di lakukan pertama kali adalah analisis kebutuhan sistem (*system requirement).* Pada tahap ini dibuat beberapa daftar kebutuhan fungsional aplikasi yang akan di buat.

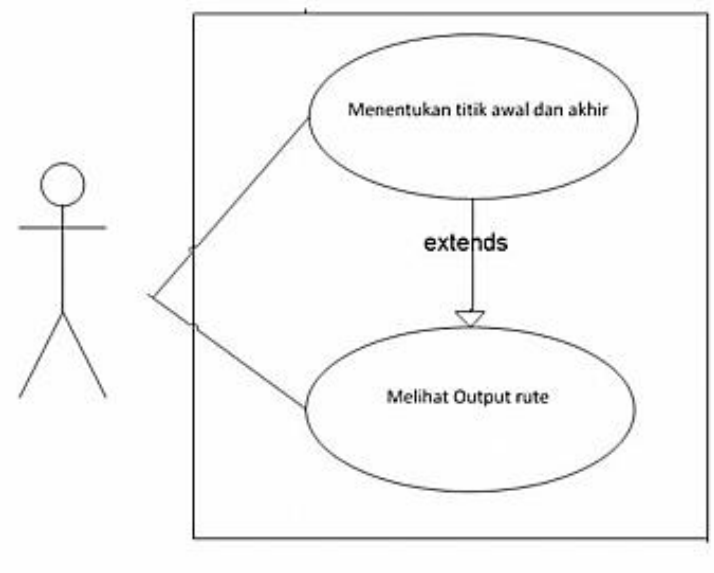
1. Aplikasi dapat menerima masukan node awal dan akhir.
2. Aplikasi dapat melakukan p*arse* node node yang kemungkinan akan dilalui.
3. Aplikasi dapat melakukan proses pencarian menggunakan algoritma *NBA*(New Bidirectional A*)*
4. Aplikasi dapat menghasilkan rute optimum rute perjalan dari peta digital.

b. Design

Di dalam tahap design, dibuat rancangan atau pemodelan dari aplikasi dengan memanfaatkan pendekata object oriented yang di implementasikan dengan UML (Unified Modelling Language) sebagai pemodelannya yang terdiri dari sebagai berikut :

1. Usecase Diagram

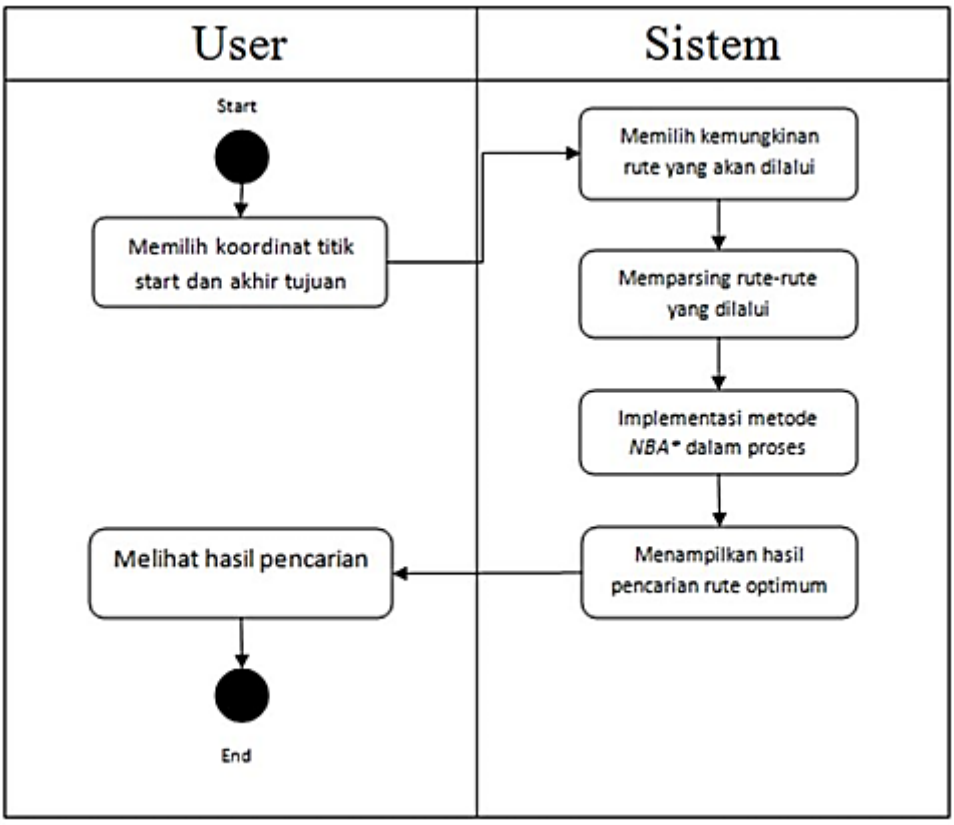
Aktor yang berperan adalah (*user)* pengguna aplikasi. Peran *user* dalam aplikasi adalah menginputkan titik awal node dan akhir tujuan untuk mengetahui rute optimum sebagai outputnya, seperti terlihat pada gambar 4 di bawah ini.



Gambar 4. Use Case Diagram

2. Activity Diagram

Dalam diagram berikut akan di modelkan alur kerja dari proses sebuah sistem dan sebagai urutan aktivitas suatu proses, yaitu proses mencari rute optimum dengan mengimplementasikan algoritma *NBA**.

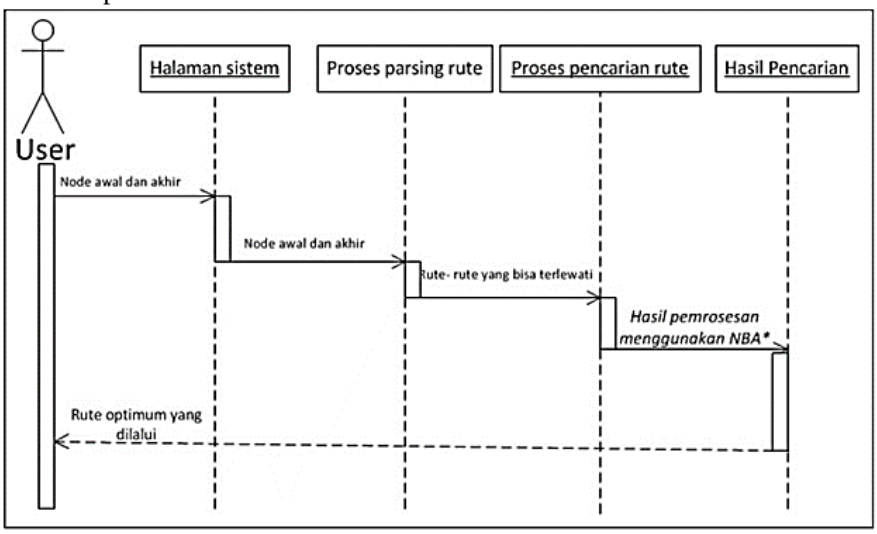


Gambar 5. Activity Diagram

Gambar 5 dapat di jelaskan sbeagai berikut, user akan menentukan koordinat awal, aplikasi akan memilih kemungkinan rute yang akan di laluii, lalu melakukan parse terhadap rute tersebut dan menjalankan NBA* untuk menghasilkan rute optimal bagi user.

3. Sequence Diagram

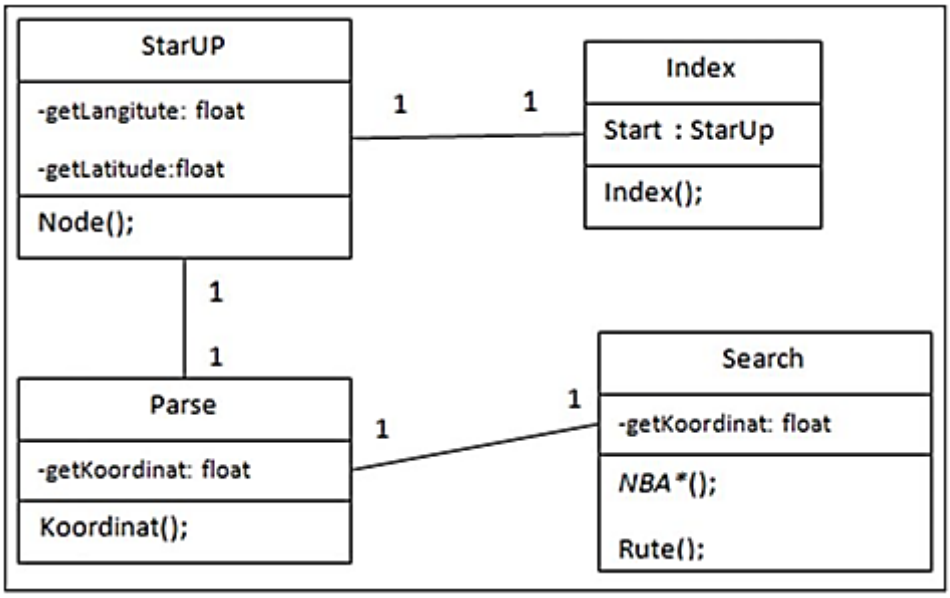
Menggambarkan tahap-tahap dimana mulai menginputkan node awal dan tujuan, proses pencarian dengan menggunkan algoritma *New Bidirectional A** serta hasil output rute optimum. Dari gambar 6 di bawah terlihat terdapat 4 tahap yang akan dikerjakan oleh aplikasi, yaitu menerima data dari user, melakukan proses parse dari input user, melakukan pencarian rute dan menampilkan hasil pencarian.



Gambar 6. Sequence Diagram

4. Class Diagram

Memodelkan beberapa *class* yang terlibat di dalam sistem beserta hubungan dan relasinya. *Class* yang di gunakan antara lain, Index, StarUp, Parse dan Search.



Gambar 7. Class Diagram

Start-Up dalah class pertama yang akan menyimpan node awal dan akhir dari data yang di inputan user, berupa padsangan data longitude dan latitude, class Start-up berkorelasi 1:1 dengan class Parse dan Index, dimana Index akan melakukan penyimpanan data kemungkinan yang terindex dari input user, kemudian Parse akan melakukan parse terhadap koordinat input

dan terakhir class Search akan melakukan pencarian atas koordinat yang di inputkan oleh user dan menyimpannya dalam rute.

c. Code

Setelah rancangan sistem selesai dibuat, proses selanjutnya adalah mengimplementasi (*coding)* aplikasi dengan *tools* PHP menggunakan editor teks berdasarkan rancangan yang sudah di modelkan sebelumnya.

d. Eksperimen dan Cara Pengujian Metode

Eksperimen dalam paper ini dilakukan dengan 2 macam pengujian yaitu :

1. *Black-Box Testing*

Pengujian dilakukan secara fungsionalitas program yang dibuat, tanpa melihat *source code.* Pengujian juga dilakukan secara urut dengan menjalankan semua fasilitas yang apa pada aplikasi dan memeriksa kesesuaian antara *input* dan *output*.

2. Uji Aplikasi berbasis algoritma *New Bidirectional A**

Pengujian dilakukan dengan mencoba melakukan perbandingan routing jalan secara nyata dan komputasi melalui aplikasi kemudian menyimpulkan hasilnya.

3. HASIL DAN PEMBAHASAN

Hasil dari paper ini adalah sebuah aplikasi penemu rute optimal, yang dapat di sajikan dalam 2 macam yaitu hasil implementasi disain dan uji terhadap aplikasi. Inti dari aplikasi yang di hasilkan adalah ada pada class diagram yang di disain sebelumnya berikut hasil dalam bentuk notasi:

a. Class StartUp

Class ini setidaknya terdiri dari atribut getLatitude dan getLongitude untuk menyimpan data awal dan akhir dari input user, dengan method utama **Node()** seperti notasi pada gambar 8 yang akan menghasilkan titik koordinat yang di simpan dalam database.

Function Node (getLangitude : Float , getLatitude Float) (input : Langitude , latitude Proses : penyimpanan query ke dalam database Output : Koordinat data yang tersimpan pada database)
Kamus: getLangitude : Float getLatitude : Float
Algoritma : **if**(getLangitude = null && getLatitude=null)**then** ➔ ”” **else** ➔ ”Insert into koordinat, values(getLangitude,getLatitude)”;

Gambar 8. Notasi Class StartUp

b. Class Parse

Class ini akan memproses data output dari class StartUp, yang terdiri dari atribut getKoordinat dan method utama **Koordinat()** yang berfungsi menampilkan koordinat sebuah titik, seperti ditunjukan notasi pada gambar 9 di bawah ini.

Function Koordinat (getKoordinat : Array)→float (input : Langitude , latitude Proses : pemanggilan query database Output : Koordinat data yang tersimpan pada database akan di panggil

untuk dilakukan proses hitung)

Kamus:
getLangitude : Float
getLatitude : Float

Algoritma :
if(getLangitude = null && getLatitude=null)**then**
➔ ” ”
else
➔ ”SELECT langitude , latitude FROM koordinat”;

Gambar 9. Notasi Class Koordinat

c. Class Search

Class ini merupakan jantung dari aplikasi, karena tempat dilakukan komputasi dengan algoritma NBA*, yang terdiri dari atribut getKoordinat yang mengambil data dari class Parse dan 2 method NBA() dan Rute(), yang di sajikan dalam notasi gambar 10 dan 11 di bawah ini.

Method NBA, notasi gambar 10, akan melakukan komputasi terhadap jarak dan waktu optimal dari koordinat awal ke koordinat akhir.

Function NBA(getKoordinatfrom , getKoordinatto)→float
(input : Langitude , latitude
Proses : menghitung jarak dan waktu optimum
Output : Jarak tempuh dan waktu yang terpilih optimum)

Kamus:
getLangitudeFrom : Float
getLatitudeTo : Float
Dlat : float
Dlang : float
h : float;

Algoritma :
Dlat = latitude.akhir – latitude.awal;
Dlong = langitude.akhir-langitude.awal;
h= D*(Dlat+Dlong);
while(node != null){
f=g+h
Uo= min{g+h}
}

Gambar 10. Notasi Class NBA

Sedangkan method Rute akan mengolah data koordinat dan menampillkan jalan yang di temukan paling optimal melalui perhitungan yang secara otomatis dan terupdate, seperti pada notasi gambar 11 di bawah ini.

Function Rute(getKoordinatfrom , getKoordinatto)→float
(input : Langitude , latitude
Proses : menampilkan rute pada peta google maps

Output : Garis rute yang menampilkan jarak dari titik awal sampai akhir)

Kamus:
getLangitudeFrom : Float
getLatitudeTo : Float

Algoritma :

```
// Menampilkan peta pada posisi tengah
var myOptions = { zoom:12, center: from,
   mapTypeId: google.maps.MapTypeId.ROADMAP
};
// Gammbar peta
var mapObject = new
google.maps.Map(document.getElementById("map"), myOptions);
var directionsService = new
google.maps.DirectionsService();
var directionsRequest = { origin: from, destination: to,
 travelMode:
 google.maps.DirectionsTravelMode.DRIVING,
 unitSystem: google.maps.UnitSystem.METRIC
};
directionsService.route(
directionsRequest, function(response, status){
 if (status == google.maps.DirectionsStatus.OK) {
    new google.maps.DirectionsRenderer({
    map: mapObject,
    directions: response });
 }else
   $("#error").append("Unable to retrieve your route<br
/>");
}
);
```

Gambar 11. Notasi Class Rute

d. Class Index

Class Index akan menampilkan hasil routing yang sudah di lakukan komputasi pada class Search, dengan atribut getKoordinat dan method utama Index(), seperti pada notasi gambar 12 di bawah.

Function Index(getKoordinatfrom,getKoordinatto)→float
(input : Langitude , latitude
Proses : Menampilkan GUI dan semua proses
Output : tampilan program dengan GUI)

Kamus:
getLangitudeFrom : Float
getLatitudeTo : Float

Algoritma :

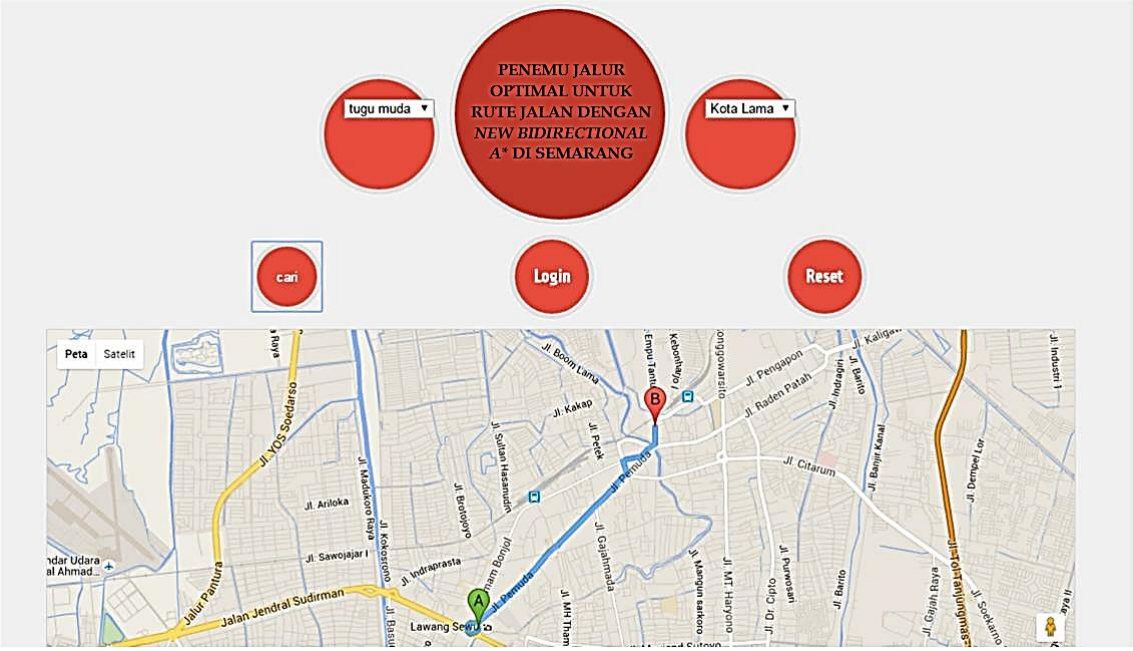
```
$(document).ready(function() {
if (typeof navigator.geolocation == "undefined") then
    output("unsupport browser with Geolocation API");
```

```
    return;

$("#from-link, #to-link").click(function(event){
   event.preventDefault();
var addressId = this.id.substring(0, this.id.indexOf("-"));
navigator.geolocation.getCurrentPosition(function(position) {
   var geocoder = new google.maps.Geocoder();
geocoder.geocode({"location": new
                google.maps.LatLng(position.coords.latitude,
                    position.coords.longitude)
                }, function(results, status) {
 if (status == google.maps.GeocoderStatus.OK)
    $("#" + addressId).val(results[0].formatted_address);
 else
    $("#error").append("Unable to retrieve your address<br />");
});
},
function(positionError){
$("#error").append("Error: " + positionError.message +
"<br />");
},
{enableHighAccuracy: true, timeout: 10 * 1000 // 10 seconds
});
});
$("#search").click(function(event) { event.preventDefault();
  var point_start = $("#start").val();
  var start = point_start.split(",");
  var mark_start = new
  google.maps.LatLng(start[0],start[1]);
  var point_finish = $("#finish").val();
  var finish = point_finish.split(",");
  var mark_finish = new
  google.maps.LatLng(finish[0],finish[1]);
  calculateRoute(mark_start, mark_finish);
});
});
```

Gambar12. Notasi Class StartUp

Berikut antarmuka aplikasi yang di hasilkan, sesuai dengan disain yang di inginkan dan berjalan sesuai fungsi, seperti ditunjukan pada gambar 13 di bawah ini.



Gambar 13. Antar Muka Aplikasi

Sementara hasil uji Black Box, yang di maksudkan untuk mengetahui fungsionalitas aplikasi seperti di sajikan pada table 1 di bawah ini dan sebagai test case untuk mengetahui berjalan atau tidaknya fungsi yang d buat ditunjukan pada table 1:

Tabel 1. Test Case Class Index

Class Uji	Butir Uji
Index	Memilih lokasi awal mulai dan akhir tujuan
Index	Menekan tombol klik dan menampilkan peta

Uji class dilakukan dengan memasukan input titik koordinat awal dan akhir dan melihat rute yang di hasilkan dan hasilnya seperti pada table 2 berikut di bawah ini.

Tabel 2. Uji class Index

Input Data	Output	Pengamatan	Kesimpulan
Memilih koordinat tujuan yang telah di sediakan	Menerima inputan untuk di proses kedalam sistem	Data koordinat berhasil di tampung dan di proses	[v] diterima [ ] ditolak

Dan terakhir uji performa aplikasi dilakukan dengan membandingkan perjalanan manual atau riil dengan komputasi yang di lakukan oleh aplikasi dimana uji performa aplikasi dilakukan sebanyak 4 trip perjalanan, seperti di sajikan pada table 3 berikut di bawah ini.

Tabel 3. Uji Perforam Aplikasi

No	Rute	NBA*	Manual	Status	Interpretasi
1	Tugu Muda – Kota ama	Jalur Pemuda	Jalur Pemuda	[v]Sinkron	Rute yang dilalui manual

					lebih cepat melewati pemuda dan pada NBA* sama dengan memilih jalan pemuda
2	Kota Lama – Simpang Lima	Jalan MH Thamrin	Jalan MH Thamrin	[v]Sinkron	Rute yang dilalui manual lebih cepat melalui jalan Mh Thamrin
3	Kota Lama – MAJT	Jalan Citarum – Jalan Ahmad Yani	Jalan Tol Tanjung Mas – Srondol	[ ] Tidak Sinkron	Rute yang dilalui manual melewati jalan tol karena pada saat jam sibuk jalan citarum padat dilalui kendaraan bermotor
4	Simpang Lima – Tugu Muda	Jalan Pandanaran	Jalan Pandanaran	[v]Sinkron	Rute yang di lewati manual dan NBA* sama

4. KESIMPULAN

Dari hasil dan pengujian sederhana, paper ini menghasilkan kesimpulan sementara sebagai berikut:

1. Aplikasi Optimum NBA* dapat melakukan perhitungan dan memberi solusi untuk menentukan rute teroptimum.
2. Aplikasi Optimum NBA* dapat menampilkan rute dari awal lokasi ke tempat tujuan
3. Berdasarkan penelitian yang telah di laksanakan maka dapat disimpulkan fungsi-fungsi berjalan sesuai yang di inginkan
4. Berdasarkan pengujian sebanyak 4 kali, 3 hasil perhitungan waktu optimum valid dengan nilai heuristic yang sudah di tentukan secara statis.

5. SARAN

Berikut merupakan beberapa hal yang perlu diperhatikan untuk melakukan penelitian lebih lanjut tentang aplikasi Optimum NBA*:

1. Adanya vitur *check in location* untuk mengetahui keberadaan pengguna.
2. Aplikasi Optimum NBA* yang masih statis bisa di buat dengan secara dinamis.
3. Sensitivitas aplikasi Optimum NBA* perlu di kembangkan secara proporsional
4. Perhitungan otomatis NBA* akan lebih baik apabila menampilkan tutorial *parsing tree* .
5. Aplikasi Optimasi NBA* akan lebih baik di aplikasikan dengan *mobile android* maupun *IOS* pada *Iphone*

DAFTAR PUSTAKA

[1] Tim Williams, "Developing A Transdisciplinary Approach To Improve Urban Traffic Congestion Based On Product Ecosystem Theory", Vol. 191, P. 11, 2014.

[2] Ashlesh Sharma, Lakshminarayanan Vipin Jain, "Road Traffic Congestion In The Developing World," March 2012.

[3] Luiz Chaimowicz, Luis Henrique, Oliveira Rios, "PNBA* : A Parallel Bidirectional Heuristic Search Algorithm," 2010.

[4] Wendy Zhang And Theresa Beaubouef, "Geographic Information Systems: Real World Applications For Computer Science," Vol. 40, 2008.

[5] Ega Julia Fajarsari, Kartini Halief, Nuryanto, Haryanto, And Dewi Agushinta, "Rancangan Sistem Informasi Geografis Pemilihan Jalan Alternatif Di Jakarta Berbasis Android ," February 2013.

[6] Jan-Henrik Haunert And Benedikt Budig, "An Algorithm For Map Matching Given Incomplete Road Data," November 2012.

[7] Wim Pijls And Henk Post, "Yet Another Bidirectional Algorithm For Shortest Paths", June 2009.

[8] Stéphanie Vanhove And Veerle Fack, "Applications Of Graph Algorithms In GIS", November 2010.

[9] K. Endro Sariyono And Muhammad Nursa'Ban, *Kartografi Dasar*. Yogyakarta, 2010.

[10] Deny Wiria Nugraha, "Perancangan Sistem Informasi Geografis Menggunakan Peta Digital," Maret 2012.

[11] Ferouw R. I. Ratu, Kaunang, And Arie S.M. Lumenta, "Peta Digital Kota Bitung," 2007.

[12] Yongke Yoswara, "Penerapan Algoritma Simplified-Memory-Bounded A* Dan Algoritma Greedy-Best First Search Dalam Pencarian Lintasan Terpendek Dan Efisiensi Tarif Perjalanan Antar Kota ," 2011.

[13] Rudy Adipranata, Andreas Handojo, And Happy Setiawan, "Aplikasi Pencari Rute Optimum Pada Peta Guna Meningkatkan Efisiensi Waktu Tempuh Pengguna Jalan Dengan Metode A* Dan Best First Search," Maret 2007.

[14] Roger S. Pressman, *Software Engineering ,7th Edition.*, 2010.