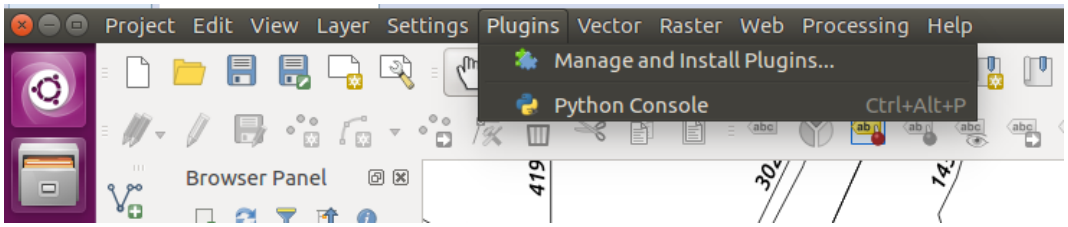
BAB 5 PERANCANGAN ALGORITME

5.1 Permasalahan pada Tahap *Pre-Processing*

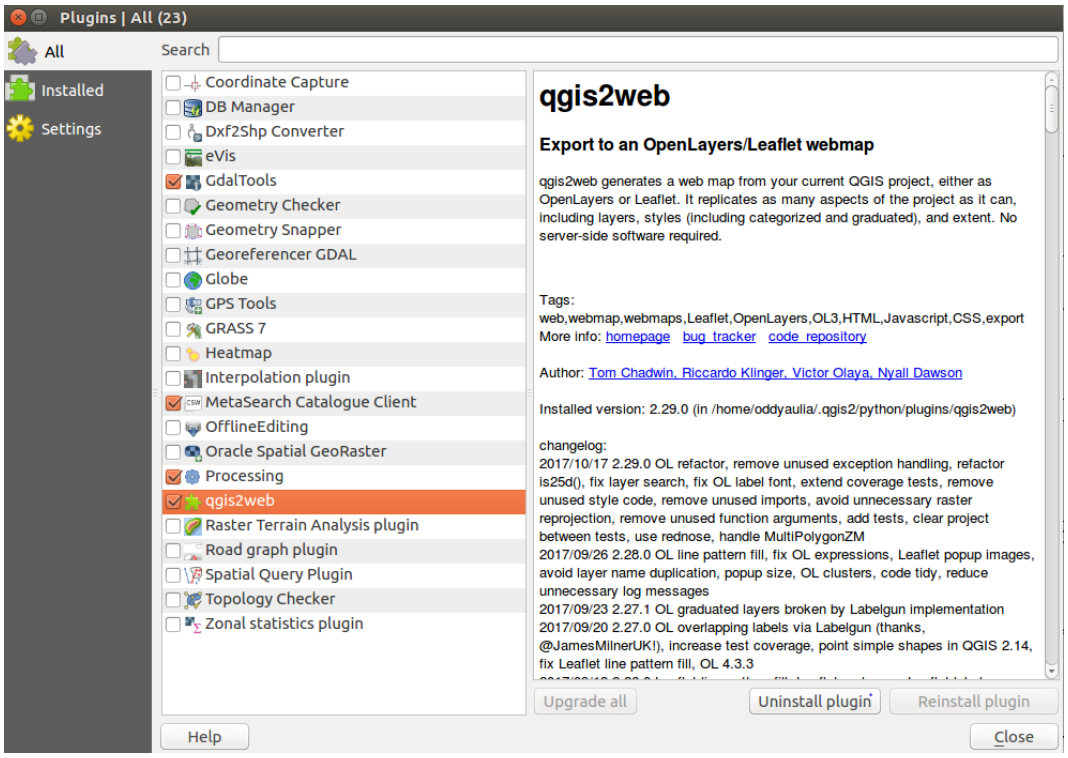
Pada tahap ini, dilakukan analisis terhadap permasalahan yang ditemukan sebelum perancangan algoritme dilakukan. Permasalahan yang dihadapi adalah memastikan hasil dari pengolahan data dapat digunakan pada algoritme yang diusulkan. Setelah melakukan beberapa analisis pada data hasil *pre-processing* ditemukan kendala sebagai berikut :

- Data *geom* (*geometry* jaringan jalan) pada data hasil *pre-processing*, memiliki format *geometry* (berbentuk kumpulan dari indeks *MultiLineString*) yang tersimpan di dalam *database* PostGIS, sehingga saat akan dibandingkan dengan *interest point* yang menggunakan koordinat *longitude* dan *latitude* perbandingan tidak dapat dilakukan karena perbedaan format data.
- Data Kecamatan yang pada mulanya diusulkan sebagai data uji memiliki cakupan yang telalu luas pada satu wilayah, sehingga pembentukan poligon Voronoi menjadi besar.
- Penentuan POI atau *interest point* untuk diagram Voronoi belum secara efisien dalam mencakup seluruh jaringan jalan sehingga terdapat *split nodes* di daerah tertentu yang memiliki jalur yang terputus.
- Permasalahan lain yang ditemukan akan di bahas lebih dalam pada sub-bab berikutnya.

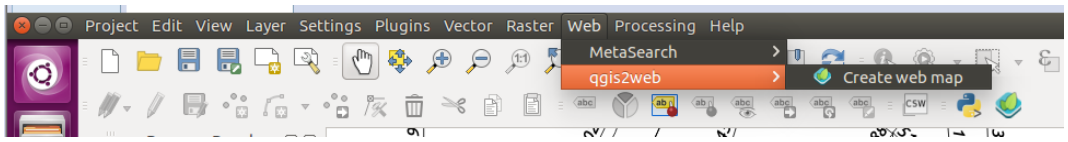
Pada permasalahan bentuk data *geom* dilakukan perubahan data yang memiliki format *geom* menjadi bentuk koordinat *longitude* dan *latitude* sebagai solusi yang digunakan. Perubahan yang dilakukan adalah dengan memanfaatkan *geo-json* untuk memisahkan data *LineString* menjadi koordinat dan memisahkan koordinat menjadi kumpulan dari *node* menggunakan kode program PHP dan *javascript*. Untuk mendapatkan *geo-json* yang dapat diolah dengan kode program PHP dan *javascript* menggunakan salah satu *plugins* dari QGIS yaitu *qgis2web. Plugin qgis2web* berfungsi untuk mengubah hasil pengolahan di QGIS menjadi berbentuk *web-map.* Untuk melakukan instalasi *plugin qgis2web* dapat dilakukan dengan mengakses *Manage and Install Plugins* yang ada di QGIS dan kemudian mencari *plugin qgis2web* kemudian melakukan instalasi seperti pada Gambar 5.1. dan Gambar 5.2. Dengan menggunakan *plugin qgis2web* data *shapefile* hasil *pre-processing* yang dimiliki akan di ekspor ke dalam bentuk *web-map* beserta *geo-json*. Pada Gambar 5.3 menggunakan *plugin qgis2 web* dan pada Gambar 5.4. sebelum dilakukan ekspor data terdapat pengaturan untuk memilih data yang akan di ekspor yang kemudian dilakukan *preview* untuk memastikan data yang di ekspor sesuai dengan kebutuhan. Hal ini dilakukan untuk menentukan hanya data yang digunakan yang akan di ekspor, sehingga tidak seluruh data yang pernah diolah atau dibuat di QGIS yang di ekspor dalam bentuk *web-map*.



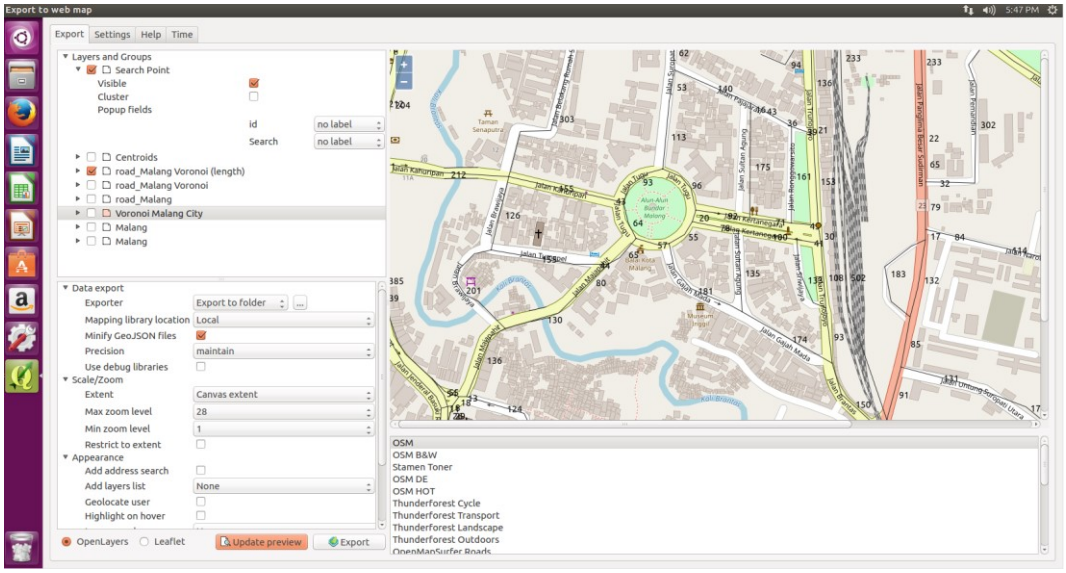
Gambar 5.1 Proses Instalasi *qgis2web*



Gambar 5.2 Instalasi *qgis2web* Berhasil



Gambar 5.3 Membuat *Web Map* dengan *qgis2web*



Gambar 5.4 *Preview Web-Map* pada *qgis2web*

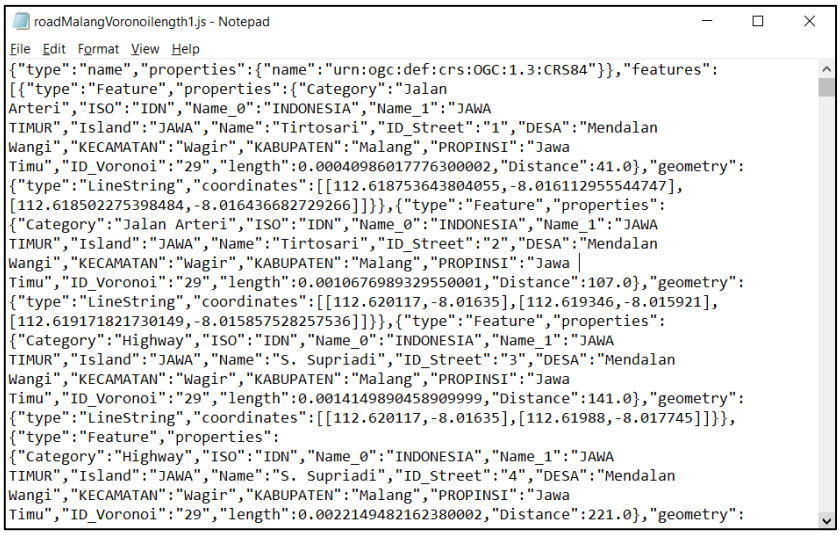
Dengan telah diekspornya data dari QGIS, maka data yang telah diolah akan menjadi seperti pada Gambar 5.5 dan Gambar 5.6 yaitu berbentuk *geo-json*. Data yang diekspor tersebut dibedakan dengan pengolahan dengan poligon Voronoi untuk nantinya digunakan oleh algoritme VCKNN sesuai dengan Gambar 5.6., yang merupakan hasil data diagram Voronoi dan untuk algoritme Dikjkstra menggunakan hasil pengolahan data jaringan jalan yang dikelompokkan berdasarkan Kecamatan/Kelurahan. Tahapan ini dapat dilakukan kembali dan berulang kali saat membutuhkan data lain yang perlu melalui pengolahan dari QGIS.

roadMalangID_0.js - Notepad

File Edit Format View Help


```
var json_roadMalangID_0 = {"type":"FeatureCollection","crs":{"type":"name","properties":
{"name":"urn:ogc:def:crs:OGC:1.3:CRS84"}},"features":[{"type":"Feature","properties":
{"Category":"Jalan Arteri","ISO":"IDN","Name_0":"INDONESIA","Name_1":"JAWA
TIMUR","Island":"JAWA","Name":"Tirtosari","SHAPE_len":0.0014,"ID_Street":"1","DESA":"Mendal
an Wangi","KECAMATAN":"Wagir","KABUPATEN":"Malang","PROPINSI":"Jawa
Timu","ID_DESA":"1","DISTANCE":"46","ID":1.0},"geometry":
{"type":"LineString","coordinates":[[112.618753643804055,-8.016112955544747],
[112.618502275398484,-8.016436682729266]]}},{"type":"Feature","properties":
{"Category":"Jalan Arteri","ISO":"IDN","Name_0":"INDONESIA","Name_1":"JAWA
TIMUR","Island":"JAWA","Name":"Tirtosari","SHAPE_len":0.00113,"ID_Street":"2","DESA":"Menda
lan Wangi","KECAMATAN":"Wagir","KABUPATEN":"Malang","PROPINSI":"Jawa
Timu","ID_DESA":"1","DISTANCE":"119","ID":2.0},"geometry":
{"type":"LineString","coordinates":[[112.620117,-8.01635],[112.619346,-8.015921],
[112.619171821730149,-8.015857528257536]]}},{"type":"Feature","properties":
{"Category":"Highway","ISO":"IDN","Name_0":"INDONESIA","Name_1":"JAWA
TIMUR","Island":"JAWA","Name":"S.
Supriadi","SHAPE_len":0.00141,"ID_Street":"3","DESA":"Mendalan
Wangi","KECAMATAN":"Wagir","KABUPATEN":"Malang","PROPINSI":"Jawa
Timu","ID_DESA":"1","DISTANCE":"158","ID":3.0},"geometry":
{"type":"LineString","coordinates":[[112.620117,-8.01635],[112.61988,-8.017745]]}},
{"type":"Feature","properties":
{"Category":"Highway","ISO":"IDN","Name_0":"INDONESIA","Name_1":"JAWA
TIMUR","Island":"JAWA","Name":"S.
Supriadi","SHAPE_len":0.00221,"ID_Street":"4","DESA":"Mendalan
```


Gambar 5.5 *Geo-Json Data Dijkstra*



Gambar 5.6 *Geo-Json Data VCKNN*

Setelah data telah berbentuk *geo-json* kemudian data akan dimasukan ke dalam *database* MySQL. *Database* MySQL digunakan karena telah memiliki hubungan langsung dengan PHP beserta *javascript.* Hal ini berbeda dengan PostGIS yang dapat terhubung langsung dengan QGIS namun tidak terhubung langsung dengan PHP, begitupula MySQL yang tidak terhubung langsung dengan QGIS. Maka dari itu, pada penelitian ini dilakukan pergeseran pemilihan *database* yaitu dengan memanfaatkan *database* MYSQL karena lebih banyak fitur yang dapat dimanfaatkan kedepannya oleh peneliti karena algoritme yang diusulkan akan dikembangkan di PHP dan menggunakan *Javascript*.

Tahapan yang dilakukan untuk memasukkan data *geo-json* adalah dengan mengambil nilai dari setiap data untuk dimasukan sesuai dengan kolom pada tabel yang berada di *database.* Pada Tabel 5.1. terdapat kode program yang memicu berjalan kode program pada Tabel 5.2. untuk menjalankan proses pemindahan data. Pada Tabel 5.1. kode program membuat akses ke *geo-json* dan menggunakan variabel yang ada di data *geo-json* untuk mengambil elemen dari *geo-json.* Kemudian akan mengirimkan elemen yang berbentuk teks tersebut ke kode program pada Tabel 5.2. untuk dimasukan ke dalam tabel.

Tabel 5.1 Kode Program Pemindahan Data *Geo-Json*

```
1  //Membuat sambungan dengan data geo-json
2  <script src="data/roadMalangVoronoilength_0"></script>
3  //Membuat sumbit button untuk memulai proses pemindahan
4         <form action="insert.php" method="POST"
5         enctype='multipart/form-data' >
6                <div id='text'> </div>
7                <button name="submit"
8                onclick="return konfirmasi();"
```

```
9                class="btn-primary form-control">Submit</button>
10       </form>
11
12 //Mengambil nilai dari geo-json dan mengirimkan ke insert.php
   <script type="text/javascript" language="JavaScript">
13       function konfirmasi(){
14         var geojsondata =
15              JSON.stringify(json_roadMalangVoronoilength_0);
16         document.getElementById("text").innerHTML =
17              "<textarea id='data' name='q'
18              style='display:none;'></textarea>";
19         document.getElementById("data").value =
20              geojsondata;
21       return true;
22 }
23 </script>
```

Pada kode program Tabel 5.2. hal pertama yang dilakukan adalah mengambil nilai yang dikirimkan pada Tabel 5.1. kemudian di *decode* dan membuat koneksi dengan *database* di MySQL. Setelah koneksi dengan *database* terbentuk data dari *geo-json* akan dimasukan menggunakan perulangan. Perulangan yang ada akan memasukkan nilai dari *geo-json* sesuai dengan tabel yang ada di *database*. Setiap data akan menjadi entitas dengan nama kolom yang disesuaikan kembali untuk mempermudah dalam mengakses kolom tersebut. Perulangan tersebut juga memisahkan data *MultiLineString* menjadi koordinat titik. Koordinat titik dimasukan kedalam tabel bersama dengan jalur yang dimiliki. Dari koordinat tersebut diambil nilai dari simpul awal dan simpul akhir untuk dijadikan *split nodes* dan sebagai titik pencarian. Penjelasan terkait kode program dapat dilihat pada Tabel 5.2.

Tabel 5.2 Kode Program Proses Pemindahan Data *Geo-Json*

```
1  //Menerima data yang dikirim dan di decode
2  $q="";
3  if(isset($_POST['q'])){
4        $q=$_POST['q'];
5        $data=json_decode($q,true);
6  //Inisalisasi koneksi database
7        $server = "localhost";
8        $username = "root";
9        $password = "";
10       $database = "skripsi";
```

```
//Membuat koneksi database
$conn = new mysqli($server, $username, $password, $database);
      if ($conn->connect_error) {
            die("Connection failed: " . $conn->connect_error);
      }
//Memasukkan data ke database
$insert='';
      $counter=1;
      $status='';
      $jumlahdata=count($data['features']);
      $iTabel="INSERT INTO `vcknn`
            (`id_street`,`category`,`ISO`,`negara`,`provinsi`,
`kepulauan`,`nama`,`kabupaten`,`kecamatan`,`kelurahan`,
`id_voronoi`,`length`,`distance`,`coordinates`,`simpul_awal`,
`simpul_akhir`) VALUES ";

      for($i=0;$i<$jumlahdata;$i++){
            if($data['features'][$i]['geometry']['type']==
            "MultiLineString"){
                  $jumlahkoordinat=
                  count($data['features'][$i]['geometry']
                  ['coordinates']);
                  for($j=0;$j<$jumlahkoordinat;$j++){
                        $insert=$insert."(
      '".$data['features'][$i]['properties']['ID_Street']."',
      '".$data['features'][$i]['properties']['Category']."',
      '".$data['features'][$i]['properties']['ISO']."',
      '".$data['features'][$i]['properties']['Name_0']."',
      '".$data['features'][$i]['properties']['Name_1']."',
      '".$data['features'][$i]['properties']['Island']."',
      '".$data['features'][$i]['properties']['Name']."',
      '".$data['features'][$i]['properties']['KABUPATEN']."',
      '".$data['features'][$i]['properties']['KECAMATAN']."',
      '".$data['features'][$i]['properties']['DESA']."',
      '".$data['features'][$i]['properties']['ID_Voronoi']."',
      '".$data['features'][$i]['properties']['length']."',
      '".$data['features'][$i]['properties']['Distance']."',
      '".json_encode($data['features'][$i]['geometry']
      ['coordinates'][$j])."',
```

```
        '".json_encode($data['features'][$i]['geometry']
        ['coordinates'][$j][0])."',
        '".json_encode($data['features'][$i]['geometry']
        ['coordinates'][$j][(count($data['features'][$i]
        ['geometry']['coordinates'][$j])-1)])."')";

                if($j<($jumlahkoordinat-1)){
                        $insert=$insert.',';
                        $counter++;
                }
        }
}
else{
$insert=$insert."(
        '".$data['features'][$i]['properties']['ID_Street']."',
        '".$data['features'][$i]['properties']['Category']."',
        '".$data['features'][$i]['properties']['ISO']."',
        '".$data['features'][$i]['properties']['Name_0']."',
        '".$data['features'][$i]['properties']['Name_1']."',
        '".$data['features'][$i]['properties']['Island']."',
        '".$data['features'][$i]['properties']['Name']."',
        '".$data['features'][$i]['properties']['KABUPATEN']."',
        '".$data['features'][$i]['properties']['KECAMATAN']."',
        '".$data['features'][$i]['properties']['DESA']."',
        '".$data['features'][$i]['properties']['ID_Voronoi']."',
        '".$data['features'][$i]['properties']['length']."',
        '".$data['features'][$i]['properties']['Distance']."',
        '".json_encode($data['features'][$i]['geometry']
        ['coordinates'])."',
        '".json_encode($data['features'][$i]['geometry']
        ['coordinates'][0])."',
        '".json_encode($data['features'][$i]['geometry']
        ['coordinates'][(count($data['features'][$i]['geometry']
        ['coordinates'])-1)])."')";
}
if($counter>=150){
        $sql=$iTabel.$insert." ";
        if ($conn->query($sql) === TRUE) {
                $insert='';
                $status=$jumlahdata.' data berhasil di inputkan';
```

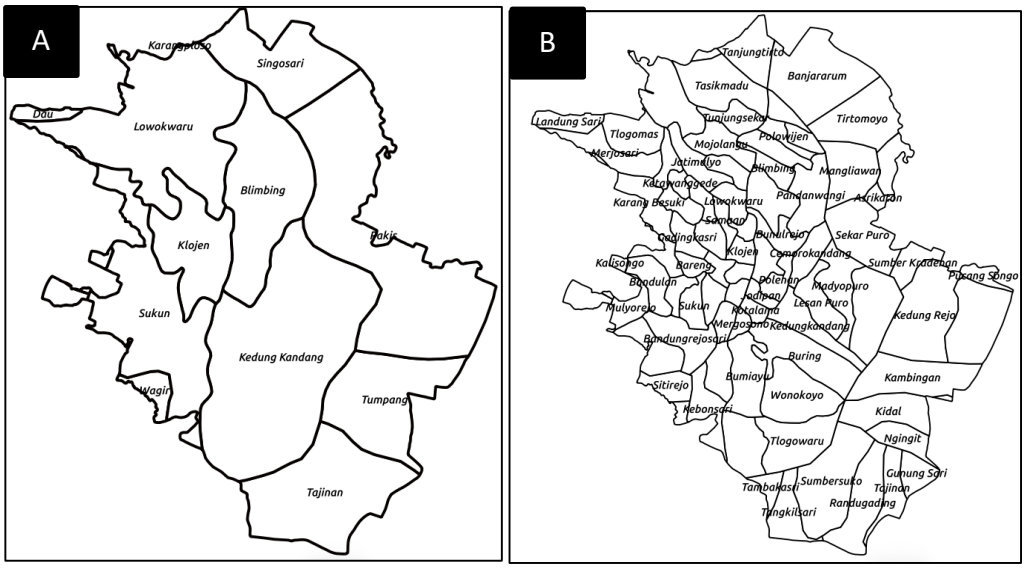
```
92                  } else {
93                  $status= "Error: " . $sql . "<br>" . $conn->error;
94                        }
95                        $counter=1;
96                  }
97          else{
98                  $insert=$insert.',';
99                  $counter++;
100                 }
101         }
102         if($insert!=''){
103                 $sql=$iTabel.substr($insert, 0, -1)." ";
104                       if ($conn->query($sql) === TRUE) {
105                             $insert='';
106                             $status=$jumlahdata.
107                             ' data berhasil di inputkan';
108                       } else {
109                             $status= "Error: " . $sql .
110                             "<br>" . $conn->error;
111
112                       }
113                 }
114 echo $status;
115 $conn->close();
116 }
```

Setelah kode program pada Tabel 5.2. telah dijalankan, maka data akan masuk ke dalam *database* dan data *MultiLineString* akan menjadi koordinat seperti pada Gambar 5.7. Setelah berhasil perlu dilakukan pengecekan kembali untuk memastikan tidak ada perulangan data yang ada di *database*. Apabila terjadi perulangan akan berpengaruh terhadap performa dari pencarian menggunakan algoritme yang diusulkan. Kode program pada Tabel 5.2. digunakan untuk memasukkan data *geo-json* untuk VCKNN dan dapat digunakan kembali untuk memasukkan data *geo-json* untuk Dijkstra dengan mengganti bagian kolom sesuai dengan data yang akan dimasukkan dan sesuai dengan kolom yang ada di tabel yang dituju. Hal ini berlaku juga untuk memasukkan data lain dengan catatan perhatikan kolom tujuan dan kolom data yang dimasukan. Dengan menerapkan penjabaran yang telah dijelaskan sebelumnya. Permasalahan pada data *MultiLineString* dapat diatasi.

coordinates	simpul_awal	simpul_akhir
[[112.69194,-7.983456001], [112.69078,-7.982769],[1...	[112.69194,-7.983456001]	[112.68962222628,-7.9821982952843]
[[112.694297,-7.984335001], [112.693077,-7.984056],...	[112.694297,-7.984335001]	[112.692108,-7.983563]

Gambar 5.7 Data *MultiLineString* Berubah Menjadi Koordinat

Setelah kode program Tabel 5.2. telah dijalankan, maka data akan masuk ke dalam *database* dan data *MultiLineString* akan menjadi koordinat seperti pada Gambar 5.7. Setelah berhasil perlu dilakukan pengecekan kembali untuk memastikan tidak ada perulangan data yang ada di *database.* Apabila terjadi perulangan akan berpengaruh terhadap performa dari pencarian menggunakan algoritme yang diusulkan. Kode program pada Tabel 5.2. digunakan untuk memasukkan data *geo-json* untuk VCKNN dan dapat digunakan kembali untuk memasukkan data *geo-json* untuk Dijkstra dengan mengganti bagian kolom sesuai dengan data yang akan dimasukan dan sesuai dengan kolom yang ada di tabel yang dituju. Hal ini berlaku juga untuk memasukkan data lain dengan catatan perhatikan kolom tujuan dan kolom data yang dimasukan. Dengan menerapkan penjabaran yang telah dijelaskan sebelumnya. Permasalahan pada data *MultiLineString* dapat diatasi.

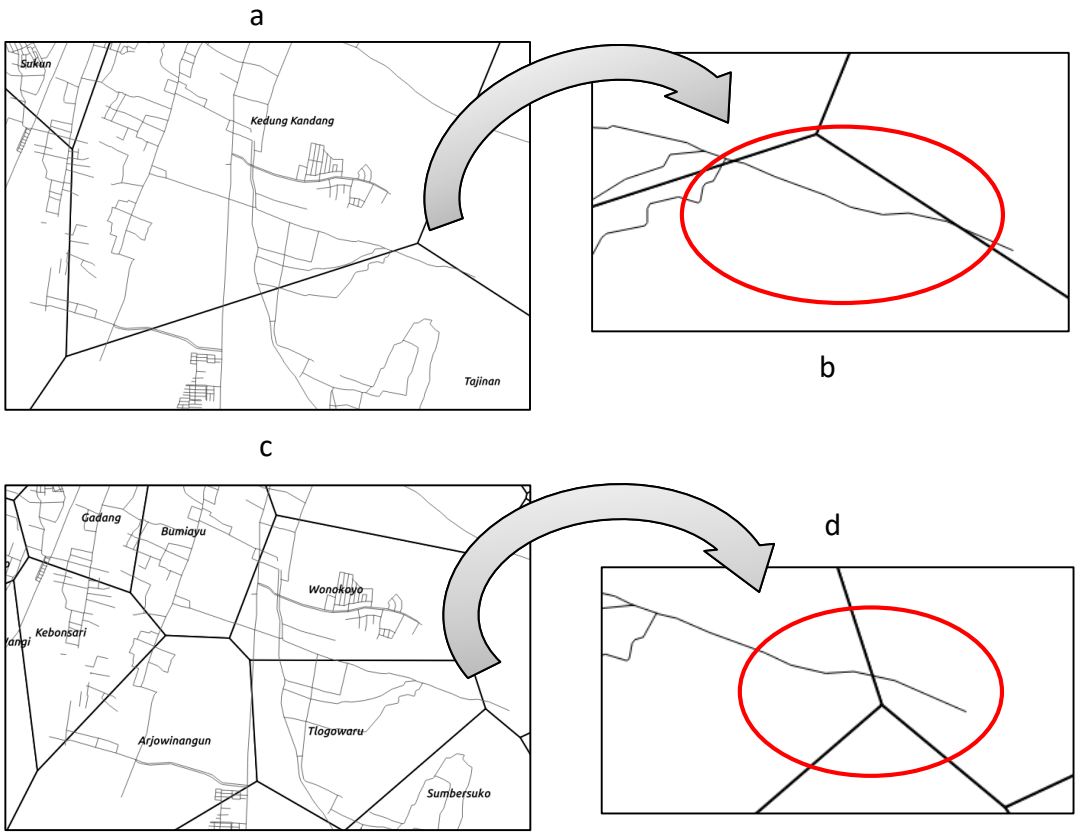


Gambar 5.8 a) Batas Administrasi Kecamatan dan b) Batas Administrasi Kelurahan

Pada permasalahan mengenai data Kecamatan yang terlalu luas dapat dilakukan perubahan dengan menggunakan data yang lebih kecil yaitu data Kelurahan atau Desa. Maksud dari data yang terlalu luas adalah Kecamatan memiliki banyak jaringan jalan dalam satu daerah Kecamatan yang akan

menyebabkan banyaknya *split nodes* dan segmen yang terbentuk. Hal ini akan membuat performa pencarian menurun apabila pencarian terjadi di dalam poligon Voronoi. Pada Gambar 5.8. merupakan ilustrasi dari pembagian daerah menggunakan data Kecamatan dan data Kelurahan terlihat jumlah yang berbeda dari kedua daerah tersebut. Pembahasan lebih lanjut akan dilakukan pada sub-bab evaluasi algoritme yang ada pada bab berikutnya.

Pada permasalahan mengenai *interest point* atau POI yang belum secara efisien mencakup seluruh wilayah adalah melakukan pemilihan POI dengan tingkat cakupan sampai seluruh wilayah atau membuat POI bantuan untuk menarik poligon Voronoi sampai ke daerah tersebut. Banyak lokasi yang dapat dijadikan POI, namun penentuan POI harus memiliki nilai dan dapat mewakili suatu wilayah. Pada penelitian ini diasumsikan POI berdasarkan titik tengah dari Kecamatan karena Kecamatan memiliki cakupan terhadap seluruh daerah di Malang.



Gambar 5.9 a) Jaringan Jalan Kecamatan , b) Segmen pada V(P) Kecamatan, c) Jaringan Jalan Kelurahan dan d) Segmen pada V(P) Kelurahan

Pada penentuan POI berdasarkan Kecamatan menghasilkan poligon Voronoi seperti pada Gambar 5.9.a. yang memiliki poligon Voronoi yang luas sedangkan pada penentuan POI berdasarkan Kelurahan menghasilkan poligon Voronoi yang lebih kecil seperti pada Gambar 5.9.c. Pada kedua hasil yang didapat dan diambil contoh berupa lokasi yang sama seperti pada Gambar 5.9.b. dan 5.9.d. yang merupakan jalan yang terpotong dengan poligon Voronoi. Pada lingkaran merah di Gambar 5.9.b. mengambil contoh garis yang melalui 3 poligon Voronoi untuk satu jalur dan menghasilkan 3 segmen yang berbeda sedangkan pada lingkaran merah di Gambar 5.9.d. merupakan lokasi contoh yang sama namun terdapat perbedaan berupa jalur yang sama tetapi hanya melalui 2 poligon Voronoi dan menghasilkan 2 segmen yang berbeda.

Dari ilustrasi pada Gambar 5.9. dapat diketahui adanya pengaruh pemilihan POI terhadap jumlah segmen dan *split nodes*. Pengaruh terhadap segmen dan *split nodes* yang terjadi akibat dari perpotongan jalan dan poligon Voronoi yang berbeda. Pada hasil di atas terlihat dengan menggunakan data Kecamatan menghasilkan segmen yang lebih banyak dibanding dengan data Kelurahan, namun hal ini bukan hasil mutlak data yang lebih besar akan menghasilkan segmen yang lebih banyak bisa jadi data yang terlalu kecil juga menghasilkan segmen yang lebih banyak pula. Maka dari itu dibutuhkan penelitan tersendiri untuk menentukan kriteria POI yang baik dan efisien.

Dengan telah dijabarkan permasalahan yang dialami setelah tahap *pre-processing* dan telah dilakukan penyelesaian terhadap permasalahan tersebut, maka akan dilanjutkan ke tahap berikutnya yaitu perancangan algoritme. Pada tahapan ini algoritme yang diusulkan akan dijabarkan dari mulai awal penerapan sampai kode program yang digunakan.

5.2 Perancangan Algoritme Dijkstra

Pada sub-bab ini akan membahas mengenai perancangan algoritme Dijkstra. Pada Gambar 5.10. merupakan struktur data yang akan dikelola oleh algoritme Dijkstra. Struktur data tersebut memiliki informasi mengenai jalan beserta dengan koordinat dan *node* dari jalan tersebut.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id_jalan	int(11)			No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
2	id	int(11)			No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
3	category	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
4	ISO	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
5	negara	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
6	provinsi	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
7	kepulauan	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
8	nama	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
9	kabupaten	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
10	kecamatan	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
11	kelurahan	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
12	id_desa	int(11)			No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
13	distance	int(11)			No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
14	coordinates	text	latin1_swedish_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
15	simpul_awal	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
16	simpul_akhir	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More

Gambar 5.10 Struktur Data Algoritme Dijkstra

Tahapan yang dilakukan berikutnya adalah menjalankan algoritme Dijkstra dengan kode program pada Tabel 5.3. Pada algoritme Dijkstra akan dilakukan pencarian antara dua titik yaitu :

Koordinat Titik Awal : 112.63681,-7.977426001

Koordinat Titik Tujuan : 112.646278,-7.977362

Kedua titik di atas merupakan titik pencarian yang dimasukkan. Berikut ini merupakan penjabaran dari algoritme Dijkstra pada Tabel 5.3.

Tabel 5.3 Algoritme Dijkstra

```
1   //Menghitung waktu eksekusi
2   $time_start = microtime(true);
3
4   //Insialisasi koneksi database
5   $server = "localhost";
6   $username = "root";
7   $password = "";
8   $db2 = "skripsi";
9
10
11  //Inisialisasi simpul awal
12  $nodeAwal='[112.63681,-7.977426001]';
13  //Inisialisasi simpul tujuan
14  $nodeTujuan='[112.646278,-7.977362]';
15          //Membuat koneksi database
16          $conn = new mysqli($server, $username, $password, $db2);
17          //S merupakan jalur yang sudah dikunjungi
18                  $S=array();
19          //B merupakan jalur belum dikunjungi
20                  $B=array();
21          //J merupakan jalur dilalui
22                  $J=array();
23          //C simpul yang saat ini, Inialisasi berdasarkan simpul awal,
24           saat perulangan berubah menjadi simpul yang dikunjungi
25                  $C=array($nodeAwal,0);
26          //MC merupakan iterasi minimum yang dilakukan
27                  $MC=array();
28          //TM merupakan jalur sementara yang dipilih,
29          //data sementara untuk dibandingkan
30                  $TM=array();
31                  $dataJalan=array();
```

```
//Mengambil nilai dari jalan
$getJalan = $conn->query("SELECT `simpul_awal` as simpul_awal,
       `simpul_akhir` as simpul_akhir, '1' as segmen FROM `road`
       ORDER BY simpul_awal ");

       if ($getJalan->num_rows > 0) {
              while($barisJalan = $getJalan->fetch_assoc()) {

              //Mengatur data berdasarkan arah
              //T = nilai tak hingga

       $dataJalan[$barisJalan['simpul_awal']][]=array($barisJalan
              ['simpul_akhir'],$barisJalan['segmen']);
       $dataJalan[$barisJalan['simpul_akhir']][]=array($barisJalan
              ['simpul_awal'],$barisJalan['segmen']);
              if(in_array($barisJalan['simpul_awal'],$B)==false){
                     $B[]=$barisJalan['simpul_awal'];
                     $J[$barisJalan['simpul_awal']]=array(0,'T');
              }
              if(in_array($barisJalan['simpul_akhir'],$B)==false){
                     $B[]=$barisJalan['simpul_akhir'];
                     $J[$barisJalan['simpul_akhir']]=array(0,'T');
              }
       }
}

echo '<pre>';
       $J[$C[0]]=array(0,0);

       //Pencarian
              while(count($B)!=0){
       //Inisialisasi MC setiap perulangan, untuk mencari nilai
       //terkecil di setiap perulangan
              $MC=array(0,0);
       //Menghapus data yang sedang dikunjungi dari data belum
       //dikunjungi
              $S[]=$C[0];
              $indeks = array_search($C[0], $B);
              unset($B[$indeks]);
```

```
72          //Mengubah perubahan nilai
73          for($i=0;$i<count($dataJalan[$C[0]]);$i++){
74          //0 merupakan node, 1 merupakan jarak (total)
75          //$dataJalan[$C[0]][$i][0] untuk mengambil nilai simpul
76          if(in_array($dataJalan[$C[0]][$i][0],$B)==true){
77                  $tempSegmen=$dataJalan[$C[0]][$i][1]+$C[1];
78                  //Jika jalur tercepat belum ditemukan, maka isi
79                  //nilainya
80          if($J[$dataJalan[$C[0]][$i][0]][1] === 'T'){
81          $J[$dataJalan[$C[0]][$i][0]]=array($C[0],$tempSegmen);
82          }
83   //Jika jalur tercepat ditemukan, dan jarak yang baru
84   //lebih kecil dari jarak sebelumnya, maka nilai akan di ganti
85          else if ($tempSegmen<$J[$dataJalan[$C[0]][$i][0]][1]){
86          $J[$dataJalan[$C[0]][$i][0]]=array($C[0],$tempSegmen);
87                  }
88          }
89  }
90                  //Mencari simpul berikutnya
91                  foreach ($B as $tempB) {
92                          if($J[$tempB][1] !== 'T'){
93                  //Jika nilai masih belum di isi, maka isi nilai MC
94                  //dengan data pertama
95                          if($MC[0]===0){
96                                  $MC=array($tempB,$J[$tempB][1])
97                          }
98                  //Jika nilai MC sudah di isi, dan nilai data baru
99                  //lebih kecil dari MC, maka nilai MC diganti
100                         else if($J[$tempB][1]<$MC[1]){
101                                 $MC=array($tempB,$J[$tempB][1]);
102                         }
103                 }
104         }
105
106         //Jika semua sudah dikunjungi, nilai MC tidak berubah, maka
107         //keluar dari perulangan
108                 if($MC[0]===0){
109                         break;
110                 }
111
```

```
112         //Nilai C berubah
113         $C=$MC;
114         }
115         $tempNode=$nodeTujuan;
116         $rute=array($tempNode);
117         $node=$tempNode;
118         //Tabel hasil pencarian
119         echo '<table>';
120               while($tempNode!=$nodeAwal){
121                     $temp=explode(',',substr(substr
122                     ($tempNode, 0, -1), 1));
123                     echo'<tr>';
124                     echo '<td>'.$temp[0].'</td>';
125                     echo '<td>'.$temp[1].'</td>';
126                     echo '<td>\''.$tempNode.'\'</td>';
127                     echo'</tr>';
128                     $tempNode=$J[$tempNode][0];
129                     $rute[]=$tempNode;
130                     $node=$tempNode.','.$node;
131
132               }
133                     $temp=explode(',',substr(substr
134                     ($tempNode, 0, -1), 1));
135                     echo'<tr>';
136                     echo '<td>'.$temp[0].'</td>';
137                     echo '<td>'.$temp[1].'</td>';
138                     echo '<td>\''.$tempNode.'\'</td>';
139                     echo'</tr>';
140               print_r($node);
141               print_r($rute);
142               echo '</table>';
143         echo '</pre>';
144
145   //Akhir perhitungan waktu eksekusi
146   $time_end = microtime(true);
147   $time = $time_end - $time_start;
148   echo "Process Time: {$time}";
```

Pada pencarian yang dilakukan titik awal dan titik akhir dimulai dari ujung simpul awal atau ujung simpul akhir, pencarian tidak dapat dilakukan dari bagian tengah jalan karena tidak sesuai dengan kaidah topologi. Pada Tabel 5.4 merupakan hasil dari pencarian Dijkstra. Bagian yang disorot dengan warna kuning merupakan titik awal pencarian dan bagian yang disorot dengan warna merah merupakan titik tujuan pencarian. *Runtime* yang dibutuhkan adalah 3.919 detik dan menghasilkan 16 *split nodes* serta 16 segmen data.

Tabel 5.4 Hasil Pencarian Algoritme Dijkstra

Longitude	***Latitude***	**Simpul yang dilalui**
112.646278	-7.977362	'[112.646278,-7.977362]'
112.645721	-7.978134	'[112.645721,-7.978134]'
112.645568	-7.978392	'[112.645568,-7.978392]'
112.644989	-7.9781130009999	'[112.644989,-7.9781130009999]'
112.643508	-7.977405	'[112.643508,-7.977405]'
112.642478	-7.97646	'[112.642478,-7.97646]'
112.642219	-7.976224	'[112.642219,-7.976224]'
112.641449	-7.975516001	'[112.641449,-7.975516001]'
112.63970888985	-7.9743824117313	'[112.63970888985,-7.9743824117313]'
112.639648	-7.974336	'[112.639648,-7.974336]'
112.638511	-7.973499	'[112.638511,-7.973499]'
112.637164066	-7.973437776	'[112.637164066,-7.973437776]'
112.636619	-7.9734130009999	'[112.636619,-7.9734130009999]'
112.636688	-7.974336	'[112.636688,-7.974336]'
112.636726	-7.975903	'[112.636726,-7.975903]'
112.63672760367	-7.9759320760377	'[112.63672760367,-7.9759320760377]'
112.63681	-7.977426001	'[112.63681,-7.977426001]'

5.3 Perancangan Algoritme Voronoi Continuous K Nearest Neighbor

Pada sub-bab ini akan membahas mengenai perancangan algoritme VCKNN. Pada Gambar 5.11. merupakan struktur data yang akan dikelola oleh algoritme VCKNN. Struktur data tersebut memiliki informasi mengenai jalan beserta dengan koordinat dan *node* dari jalan tersebut. Perbedaan utama antara data untuk VCKNN dan data untuk Dijkstra adalah penggunaan poligon Voronoi yang membatasi suatu daerah. Pada Gambar 5.11. dapat diketahui perbedaan tersebut dengan ditemukan idv yang merupakan id Voronoi dimana lokasi jalan tersebut berada.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id_jalanv	int(11)			No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
2	id	int(11)			No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
3	category	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
4	ISO	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
5	negara	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
6	provinsi	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
7	kepulauan	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
8	nama	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
9	kabupaten	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
10	kecamatan	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
11	kelurahan	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
12	idv	int(11)			No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
13	length	double			No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
14	distance	int(11)			No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
15	coordinates	text	latin1_swedish_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
16	simpul_awal	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More
17	simpul_akhir	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values More

Gambar 5.11 Struktur Data Algoritme VCKNN

Tahapan yang dilakukan berikutnya adalah mencari tetangga dari setiap poligon Voronoi dengan kode program pada Tabel 5.5. Pencarian yang dilakukan adalah dengan mencari *border point* yang saling bersinggungan pada setiap poligon Voronoi. Berikut ini merupakan penjabaran dari pencarian tetangga dari setiap poligon Voronoi tersebut.

Tabel 5.5 Kode Program Mencari NN dari V(P)

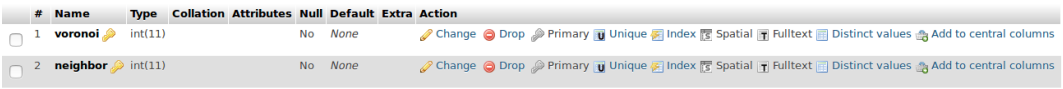
```
1   //Insialisasi koneksi database
2   $server = "localhost";
3   $username = "root";
4   $password = "";
5   $db2 = "skripsi";
6   //Membuat koneksi database
7   $conn = new mysqli($server, $username, $password, $db2);
8   //Membuat tabel voronoi
9   $buatVoronoi = $conn->query("CREATE TABLE `voronoi` (
10          `voronoi` int(11) NOT NULL,
11          `neighbor` int(11) NOT NULL,
12          PRIMARY KEY(`voronoi`,`neighbor`))");
13
14          //Mengambil id Voronoi
15          $getVoronoi =
16                  $conn->query("SELECT DISTINCT idv
17                                          FROM vcknn ORDER BY idv");
18
19          //Inisialisasi temp dan indeks
20          $temp=array();
21          $indeks=0;
```

```
22  if ($getVoronoi->num_rows > 0) {
23        while($dataVoronoi = $getVoronoi->fetch_assoc()) {
24              $getTetangga = $conn->query("SELECT DISTINCT idv
25                    FROM (SELECT id,id_jalanv,simpul_awal,
26                          simpul_akhir,idv
27                          FROM vcknn WHERE id IN
28                          (SELECT id FROM `vcknn` WHERE id IN
29                                (SELECT id FROM `vcknn`
30                                GROUP BY id HAVING COUNT(id)>1)
31                                AND
32                             idv='".$dataVoronoi['idv']."')
33                                AND
34                             idv <>'".$dataVoronoi['idv']."') v1
35                          WHERE
36  (v1.simpul_awal IN (
37        SELECT jv1.`simpul_awal` FROM `vcknn` jv1 WHERE jv1.
38        `idv` = '".$dataVoronoi['idv']."' AND
39        jv1.id_jalanv<>v1.id_jalanv
40  UNION
41        SELECT jv2.`simpul_akhir` FROM `vcknn` jv2 WHERE jv2.
42        `idv` = '".$dataVoronoi['idv']."' AND
43         jv2.id_jalanv<>v1.id_jalanv)
44
45  OR
46  v1.simpul_akhir IN (
47        SELECT jv3.`simpul_awal` FROM `vcknn` jv3 WHERE jv3.
48        `idv` = '".$dataVoronoi['idv']."' AND
49         jv3.id_jalanv<>v1.id_jalanv
50  UNION
51        SELECT jv4.`simpul_akhir` FROM `vcknn` jv4 WHERE jv4.
52        `idv` = '".$dataVoronoi['idv']."' AND
53        jv4.id_jalanv<>v1.id_jalanv))
54  AND
55  (v1.simpul_awal IN (
56        SELECT jv1.`simpul_awal` FROM `vcknn` jv1 WHERE jv1.
57        `idv` <> '".$dataVoronoi['idv']."' AND
58        jv1.id_jalanv<>v1.id_jalanv
59  UNION
60        SELECT jv2.`simpul_akhir` FROM `vcknn` jv2 WHERE jv2.
61        `idv` <> '".$dataVoronoi['idv']."' AND
62         jv2.id_jalanv<>v1.id_jalanv)
```

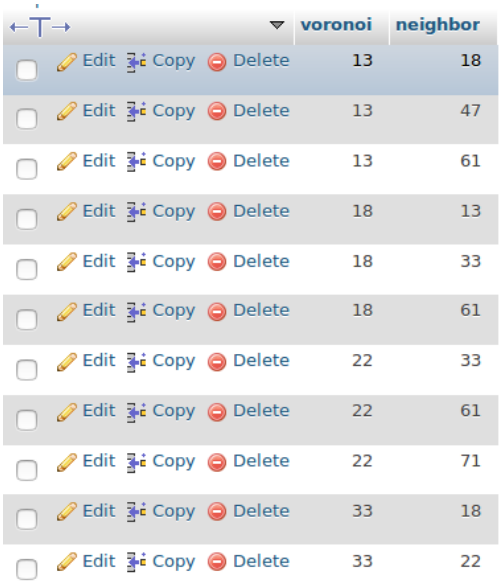
```
OR
v1.simpul_akhir IN (
       SELECT jv3.`simpul_awal` FROM `vcknn` jv3 WHERE jv3.
       `idv` <>'".$dataVoronoi['idv']."' AND
       jv3.id_jalanv<>v1.id_jalanv
UNION
       SELECT jv4.`simpul_akhir` FROM `vcknn` jv4 WHERE jv4.
       `idv` <> '".$dataVoronoi['idv']."' AND
        jv4.id_jalanv<>v1.id_jalanv))");
if ($getTetangga->num_rows > 0) {
       $temp[$indeks][0]=$dataVoronoi['idv'];
       while($dataTetanggaVoronoi = $getTetangga->fetch_assoc()) {
              $temp[$indeks][]=$dataTetanggaVoronoi['idv'];
              }
              $indeks++;
              }
       }
}
//Memasukkan data voronoi beserta tetangganya ke database
       $iTabel="INSERT INTO voronoi  VALUES ";
       $insert='';
       $counter=1;
       $status='';
       $jumlahdata=count($temp);
       for($i=0;$i<$jumlahdata;$i++){
              for($j=1;$j<count($temp[$i]);$j++){
       $insert=$insert."('".$temp[$i][0]."','".$temp[$i][$j]."')";
       if($counter>=80){
                     $sql=$iTabel.$insert." ";
                     if ($conn->query($sql) === TRUE) {
                     $insert='';
                     $status=$jumlahdata.
                            ' data berhasil di inputkan';
                     }
              else {
                     $status= "Error: " . $sql . "<br>" .
                     $conn->error;
                     }
                     $counter=1;
}
```

```
103                         else{
104                                 $insert=$insert.',';
105                                 $counter++;
106                         }
107                 }
108         }
109         if($insert!=''){
110                 $sql=$iTabel.substr($insert, 0, -1)." ";
111                         if ($conn->query($sql) === TRUE) {
112                                 $insert='';
113                                 $status=$jumlahdata.
114                                 ' data berhasil di inputkan';
115                         } else {
116                                 $status= "Error: " . $sql . "<br>" .
117                                  $conn->error;
118                         }
119                 }
120
121         echo $status;
122         $conn->close();
123         echo "</br>";
```

Dengan menjalankan kode program pada Tabel 5.5. didapatkan hasil seperti pada Gambar 5.12 yaitu tabel yang menyimpan data berupa id dari setiap poligon Voronoi beserta tetangga dari setiap poligon Voronoi tersebut. Fungsi dari membuat tabel ini adalah menyimpan pencarian terhadap tetangga terdekat karena tetangga dari setiap poligon Voronoi sudah terlebih dahulu didefinisikan, sehingga pada saat melakukan pencarian terhadap suatu lokasi hanya perlu membandingkan jarak titik awal pencarian dan poligon Voronoi yang memiliki jalur ke titik tujuan tanpa perlu melakukan pencarian terhadap setiap tetangga dari poligon Voronoi. Hal ini membuat pencarian hanya berfokus kepada poligon Voronoi terdekat. Pada Gambar 5.13. merupakan hasil dari kode program Tabel 5.5. yaitu poligon Voronoi beserta tetangganya.



Gambar 5.12 Struktur Data Voronoi Diagram dan NN



	voronoi	neighbor
Edit Copy Delete	13	18
Edit Copy Delete	13	47
Edit Copy Delete	13	61
Edit Copy Delete	18	13
Edit Copy Delete	18	33
Edit Copy Delete	18	61
Edit Copy Delete	22	33
Edit Copy Delete	22	61
Edit Copy Delete	22	71
Edit Copy Delete	33	18
Edit Copy Delete	33	22

Gambar 5.13 V(P) Beserta Tetangga

Tahapan yang dilakukan berikutnya adalah mencari *moving interval* dari setiap poligon Voronoi dengan kode program pada Tabel 5.6. Pencarian yang dilakukan adalah dengan menemukan jalur yang dilalui dari tepi poligon Voronoi satu ke tepi poligon Voronoi tetangganya. Berikut ini merupakan penjabaran dari pencarian *moving interval* dari poligon Voronoi tersebut.

Tabel 5.6 Kode Program Mencari *Moving Interval*

```
1   //Insialisasi koneksi database
2   $server = "localhost";
3   $username = "root";
4   $password = "";
5   $db2 = "skripsi";
6
7   //Pencarian MI seperti pada algoritme Dijkstra
8   //simpul = split nodes
9   function jalur($nodeAwal,$semuaNode,$idv){
10              $S=array();
11              $B=array();
12              $J=array();
13              $C=array($nodeAwal,0);
14              $MC=array();
15              $TM=array();
```

```
        $conn = new mysqli($server, $username, $password, $db2);
        $getJalan = $conn->query("SELECT
                `simpul_awal` as simpul_awal,
                `simpul_akhir` as simpul_akhir,
                '1' as segmen FROM `vcknn` where idv='".$idv."'
                ORDER BY simpul_awal ");
                $dataJalan=array();
        if ($getJalan->num_rows > 0) {
                while($barisJalan = $getJalan->fetch_assoc()) {
        $dataJalan[$barisJalan['simpul_awal']][]=
        array($barisJalan['simpul_akhir'],$barisJalan['segmen']);
        $dataJalan[$barisJalan['simpul_akhir']][]=
        array($barisJalan['simpul_awal'],$barisJalan['segmen']);
        if(in_array($barisJalan['simpul_awal'],$B)==false){
                $B[]=$barisJalan['simpul_awal'];
                $J[$barisJalan['simpul_awal']]=array(0,'T');
        }
        if(in_array($barisJalan['simpul_akhir'],$B)==false){
                $B[]=$barisJalan['simpul_akhir'];
                $J[$barisJalan['simpul_akhir']]=array(0,'T');
        }
    }
}
        $J[$C[0]]=array(0,0);
while(count($B)!=0){
        $MC=array(0,0);
        $S[]=$C[0];
        $indeks = array_search($C[0], $B);
        unset($B[$indeks]);
        for($i=0;$i<count($dataJalan[$C[0]]);$i++){
                if(in_array($dataJalan[$C[0]][$i][0],$B)==true){
        $tempSegmen=$dataJalan[$C[0]][$i][1]+$C[1];
                if($J[$dataJalan[$C[0]][$i][0]][1] === 'T'){
        $J[$dataJalan[$C[0]][$i][0]]=array($C[0],$tempSegmen);
        }
                else if
                ($tempSegmen<$J[$dataJalan[$C[0]][$i][0]][1]){
        $J[$dataJalan[$C[0]][$i][0]]=array($C[0],$tempSegmen);}
        }
}
```

```
57          foreach ($B as $tempB) {
58                  if($J[$tempB][1] !== 'T'){
59                          if($MC[0]===0){
60                                  $MC=array($tempB,$J[$tempB][1]);
61          } else if($J[$tempB][1]<$MC[1]){
62                  $MC=array($tempB,$J[$tempB][1]);
63                  }
64          }
65  }
66
67          if($MC[0]===0){
68                  break;
69          }
70  $C=$MC;
71          }
72          $hasilnya=array();
73          foreach ($semuaNode as $tempNode){
74                  $akhir=$tempNode;
75                  if($J[$akhir][0]!==0 ){
76                          $segmen=$J[$akhir][1];
77                          if($akhir != $nodeAwal){
78                                  $node=$akhir;
79                                  while($akhir!=$nodeAwal){
80                                          $akhir=$J[$akhir][0];
81                                          $node=$akhir.','.$node;
82                                          }
83          $hasilnya[]=array('awal'=>$nodeAwal,'akhir'=>$tempNode,
84                  'node'=>$node,'segmen'=>$segmen,'idv'=>$idv);
85                          }
86                  }
87          }
88          return $hasilnya;
89  }
90  //Memasukkan MI ke database
91          $insert='';
92          $counter=1;
93          $status='';
94
95          $iTabel="INSERT INTO `mi`(`awal`, `akhir`, `node`,
96          `segmen`, `idv`) VALUES ";
```

```
$getTitik = $conn->query("select simpul_awal AS simpul FROM vcknn
       WHERE idv <>'".$idv."' AND simpul_awal IN
       (SELECT simpul_awal FROM(SELECT `id_jalanv`, `id`,
       `coordinates`, `simpul_awal`, `simpul_akhir`, `idv` FROM
       `vcknn` WHERE `idv` = '".$idv."') AS v1
       WHERE
(v1.simpul_awal NOT IN (
       SELECT jv1.`simpul_awal` FROM `vcknn` jv1 WHERE jv1.
       `idv` = '".$idv."' AND jv1.id_jalanv<>v1.id_jalanv
UNION
       SELECT jv2.`simpul_akhir` FROM `vcknn` jv2 WHERE jv2.
       `idv` = '".$idv."' AND jv2.id_jalanv<>v1.id_jalanv)
OR v1.simpul_akhir NOT IN (
       SELECT jv3.`simpul_awal` FROM `vcknn` jv3 WHERE jv3.
       `idv` = '".$idv."' AND jv3.id_jalanv<>v1.id_jalanv
UNION
       SELECT jv4.`simpul_akhir` FROM `vcknn` jv4 WHERE jv4.
       `idv` = '".$idv."' AND jv4.id_jalanv<>v1.id_jalanv))

AND
(v1.simpul_awal in (
       SELECT jv1.`simpul_awal` FROM `vcknn` jv1 WHERE jv1.
       `idv` <> '".$idv."' AND jv1.id_jalanv<>v1.id_jalanv
UNION
       SELECT jv2.`simpul_akhir` FROM `vcknn` jv2 WHERE jv2.
       `idv` <> '".$idv."' AND jv2.id_jalanv<>v1.id_jalanv)
OR
v1.simpul_akhir in (
       SELECT jv3.`simpul_awal` FROM `vcknn` jv3 WHERE jv3.
       `idv` <> '".$idv."' AND jv3.id_jalanv<>v1.id_jalanv

UNION
       SELECT jv4.`simpul_akhir` FROM `vcknn` jv4 WHERE jv4.
       `idv` <> '".$idv."' AND jv4.id_jalanv<>v1.id_jalanv))
UNION
     SELECT simpul_akhir FROM (SELECT `id_jalanv`, `id`,
       `coordinates`, `simpul_awal`, `simpul_akhir`, `idv` FROM
       `vcknn` WHERE `idv` = '".$idv."') AS v1
       WHERE
(v1.simpul_awal NOT IN (
       SELECT jv1.`simpul_awal` FROM `vcknn` jv1 WHERE jv1.
       `idv` = '".$idv."' and jv1.id_jalanv<>v1.id_jalanv
```

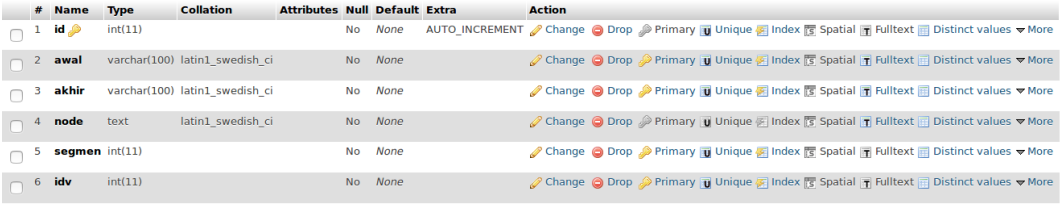
```
UNION
       SELECT jv2.`simpul_akhir` FROM `vcknn` jv2 WHERE jv2.
       `idv` <> '".$idv."' AND jv2.id_jalanv<>v1.id_jalanv)
OR v1.simpul_akhir in (
       SELECT jv3.`simpul_awal` FROM `vcknn` jv3 WHERE jv3.
       `idv` <>'".$idv."' AND jv3.id_jalanv<>v1.id_jalanv
UNION
       SELECT jv4.`simpul_akhir` FROM `vcknn` jv4 WHERE jv4.
       `idv` <> '".$idv."' AND jv4.id_jalanv<>v1.id_jalanv)))
UNION
       SELECT simpul_akhir AS simpul FROM `vcknn`
       WHERE idv <>'".$idv."' AND simpul_akhir IN
       (SELECT simpul_awal FROM(SELECT `id_jalanv`, `id`,
       `coordinates`, `simpul_awal`, `simpul_akhir`, `idv` FROM
       `vcknn` WHERE `idv` = '".$idv."') AS v1
              WHERE
(v1.simpul_awal NOT IN (
       SELECT jv1.`simpul_awal` FROM `vcknn` jv1 WHERE jv1.
       `idv` = '".$idv."' AND jv1.id_jalanv<>v1.id_jalanv
UNION
       SELECT jv2.`simpul_akhir` FROM `vcknn` jv2 WHERE jv2.
       `idv` = '".$idv."' AND jv2.id_jalanv<>v1.id_jalanv)
OR
v1.simpul_akhir NOT IN (
       SELECT jv3.`simpul_awal` FROM `vcknn` jv3 WHERE jv3.
       `idv` = '".$idv."' AND jv3.id_jalanv<>v1.id_jalanv
UNION
       SELECT jv4.`simpul_akhir` FROM `vcknn` jv4 WHERE jv4.
       `idv` = '".$idv."' AND jv4.id_jalanv<>v1.id_jalanv))
AND
(v1.simpul_awal IN (
       SELECT jv1.`simpul_awal` FROM `vcknn` jv1 WHERE jv1.
       `idv` <> '".$idv."' AND jv1.id_jalanv<>v1.id_jalanv
UNION
       SELECT jv2.`simpul_akhir` FROM `vcknn` jv2 WHERE jv2.
       `idv` <> '".$idv."' AND jv2.id_jalanv<>v1.id_jalanv)
OR
v1.simpul_akhir in (
       SELECT jv3.`simpul_awal` FROM `vcknn` jv3 WHERE jv3.
       `idv` <> '".$idv."' AND jv3.id_jalanv<>v1.id_jalanv
```

```
UNION
       SELECT jv4.`simpul_akhir` FROM `vcknn` jv4 WHERE jv4.
       `idv` <> '".$idv."' AND jv4.id_jalanv<>v1.id_jalanv))
UNION
       SELECT simpul_akhir FROM(SELECT `id_jalanv`, `id`,
       `coordinates`, `simpul_awal`, `simpul_akhir`, `idv` FROM
       `vcknn` WHERE `idv` = '".$idv."') AS v1
       WHERE
(v1.simpul_awal NOT IN (
       SELECT jv1.`simpul_awal` FROM `vcknn` jv1 WHERE jv1.
       `idv` = '".$idv."' AND jv1.id_jalanv<>v1.id_jalanv
UNION
       SELECT jv2.`simpul_akhir` FROM `vcknn` jv2 WHERE jv2.
       `idv` = '".$idv."' AND jv2.id_jalanv<>v1.id_jalanv)
OR
v1.simpul_akhir NOT IN (
       SELECT jv3.`simpul_awal` FROM `vcknn` jv3 WHERE jv3.
       `idv` = '".$idv."' AND jv3.id_jalanv<>v1.id_jalanv
UNION
       SELECT jv4.`simpul_akhir` FROM `vcknn` jv4 WHERE jv4.
       `idv` = '".$idv."' AND jv4.id_jalanv<>v1.id_jalanv))
AND
(v1.simpul_awal IN (
       SELECT jv1.`simpul_awal` FROM `vcknn` jv1 WHERE jv1.
       `idv` <> '".$idv."' AND jv1.id_jalanv<>v1.id_jalanv
UNION
       SELECT jv2.`simpul_akhir` FROM `vcknn` jv2 WHERE jv2.
       `idv` <> '".$idv."' AND jv2.id_jalanv<>v1.id_jalanv)
OR
v1.simpul_akhir IN (
       SELECT jv3.`simpul_awal` FROM `vcknn` jv3 WHERE jv3.
       `idv` <>'".$idv."' and jv3.id_jalanv<>v1.id_jalanv
UNION
       SELECT jv4.`simpul_akhir` FROM `vcknn` jv4 WHERE jv4.
       `idv` <> '".$idv."' and jv4.id_jalanv<>v1.id_jalanv)))");
if ($getTitik->num_rows > 0) {
       while($dataTitik = $getTitik->fetch_assoc()) {
       $data[$idv][]=$dataTitik['simpul'];}}
              }
}
```

```
220 //Memasukkan MI ke database
221 $moveInterval=array();
222             $jml=0;
223       foreach($data as $idv => $splitNode){
224             foreach ($splitNode as $node){
225             $tempArray=jalur($node,$splitNode,$idv);
226                   if (empty($tempArray)==false) {
227                         //Menampung nilai MI
228                         $moveInterval[]=$tempArray;
229                         //Memasukkan data MI
230       foreach ($tempArray as $tmpML){
231       $insert=$insert."('".$tmpML['awal']."','".$tmpML['akhir'].
232       "','".$tmpML['node']."','".$tmpML['segmen']."','".
233       $tmpML['idv']."')";
234       $jml++;
235       if($counter>=10){
236             $sql=$iTabel.$insert;
237             if ($conn->query($sql) === TRUE) {
238
240                   $insert='';
241                   $status=' data berhasil di inputkan';
242             } else {
243                   $status= "Error: " . $sql . "<br>" .
244                   $conn->error;
245             }
246             $counter=1;
247       } else{
248             $insert=$insert.',';
249             $counter++;
250                         }
251                   }
252             }
253       }
254 }
255       $conn->close();
```

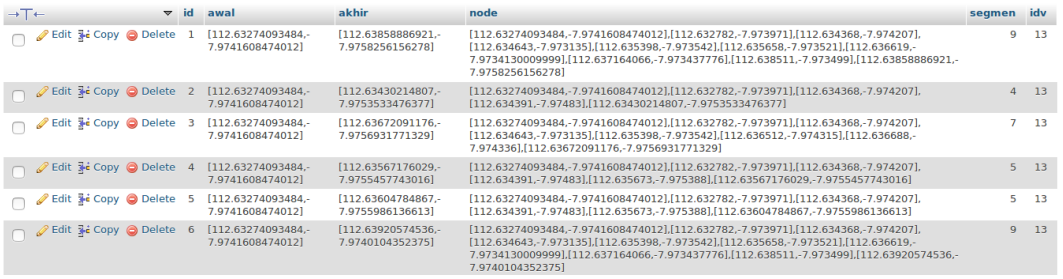
Dengan menjalankan kode program pada Tabel 5.6. didapatkan hasil seperti pada Gambar 5.14 yaitu tabel yang menyimpan data berupa simpul awal, simpul akhir. Jalur yang dilewati oleh *moving interval,* jumlah segmen yang dilalui dan poligon Voronoi tempat *moving interval* berada. Fungsi dari membuat tabel ini adalah menyimpan pencarian jalur antara tepi Voronoi satu dengan tepi Voronoi

tetangganya yang disebut *moving interval.* Pencarian ini dilakukan agar pada saat melakukan pencarian antar titik awal dan titik tujuan apabila berada pada tepi poligon Voronoi dapat langsung menggunakan hasil pencarian dengan *moving interval*. Pada Gambar 5.16. merupakan hasil dari kode program Tabel 5.6. yaitu kumpulan dari *moving interval*.



#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	id	int(11)			No	None	AUTO_INCREMENT	Change Drop Primary Unique Index Spatial Fulltext Distinct values More
2	awal	varchar(100)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values More
3	akhir	varchar(100)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values More
4	node	text	latin1_swedish_ci		No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values More
5	segmen	int(11)			No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values More
6	idv	int(11)			No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values More

Gambar 5.14 Struktur Data *Moving Interval*



	id	awal	akhir	node	segmen	idv
Edit Copy Delete	1	[112.63274093484,-7.9741608474012]	[112.63858886921,-7.9758256156278]	[112.63274093484,-7.9741608474012],[112.632782,-7.973971],[112.634368,-7.974207],[112.634643,-7.973135],[112.635398,-7.973542],[112.635658,-7.973521],[112.636619,-7.9734130009999],[112.637164066,-7.973437776],[112.638511,-7.973499],[112.63858886921,-7.9758256156278]	9	13
Edit Copy Delete	2	[112.63274093484,-7.9741608474012]	[112.63430214807,-7.97535334476377]	[112.63274093484,-7.9741608474012],[112.632782,-7.973971],[112.634368,-7.974207],[112.634391,-7.97483],[112.63430214807,-7.97535334476377]	4	13
Edit Copy Delete	3	[112.63274093484,-7.9741608474012]	[112.63672091176,-7.9756931771329]	[112.63274093484,-7.9741608474012],[112.632782,-7.973971],[112.634368,-7.974207],[112.634643,-7.973135],[112.635398,-7.973542],[112.636512,-7.974315],[112.636688,-7.974336],[112.63672091176,-7.9756931771329]	7	13
Edit Copy Delete	4	[112.63274093484,-7.9741608474012]	[112.63567176029,-7.97555457743016]	[112.63274093484,-7.9741608474012],[112.632782,-7.973971],[112.634368,-7.974207],[112.634391,-7.97483],[112.635673,-7.975388],[112.63567176029,-7.97555457743016]	5	13
Edit Copy Delete	5	[112.63274093484,-7.9741608474012]	[112.63604784867,-7.9755986136613]	[112.63274093484,-7.9741608474012],[112.632782,-7.973971],[112.634368,-7.974207],[112.634391,-7.97483],[112.635673,-7.975388],[112.63604784867,-7.9755986136613]	5	13
Edit Copy Delete	6	[112.63274093484,-7.9741608474012]	[112.63920574536,-7.9740104352375]	[112.63274093484,-7.9741608474012],[112.632782,-7.973971],[112.634368,-7.974207],[112.634643,-7.973135],[112.635398,-7.973542],[112.635658,-7.973521],[112.636619,-7.9734130009999],[112.637164066,-7.973437776],[112.638511,-7.973499],[112.63920574536,-7.9740104352375]	9	13

Gambar 5.15 Data *Moving Interval*

Dengan telah dilakukan penerapan kedua kode program pada Tabel 5.5. dan Tabel 5.6. maka didapatkan tetangga terdekat dari setiap poligon Voronoi dan *moving interval* dari setiap poligon Voronoi. Tahapan yang akan dilakukan selanjutnya adalah melakukan pencarian antara dua titik dengan algoritme VCKNN pada Tabel 5.7. Berikut ini merupakan penjabaran dari pencarian antara dua titik menggunakan algoritme VCKNN.

Tabel 5.7 Algoritme VCKNN

```
1  //Menghitung waktu eksekusi
2  $time_start = microtime(true);
3
4  //Insialisasi koneksi database
5  $server = "localhost";
6  $username = "root";
7  $password = "";
8  $db2 = "skripsi";
```

```
9  //Memasukkan simpul awal pencarian beserta V(P)
10         $nodeAwal='[112.63681,-7.977426001]';
11         $vAwal=61;
12 //Memasukkan simpul tujuan pencarian beserta V(P)
13         $nodeTujuan='[112.646278,-7.977362]';
14         $vTujuan= 33;
15 //S merupakan jalur yang sudah dicari
16 //B merupakan jalur yang belum dicari
17 //T merupakan tetangga dari V(P)
18         $tempT=array();
19         $tempS=array();
20         $tempB=array();
21
22 //Membuat koneksi database
23         $conn = new mysqli($server, $username, $password, $db2);
24
25 //Mengambil data V(P) beserta tetangga
26         $getVoronoi = $conn->query("SELECT `voronoi`, `neighbor`
27         FROM `voronoi` order by voronoi,neighbor");
28         $dataVoronoi=array();
29 if ($getVoronoi->num_rows > 0) {
30         while($barisVoronoi = $getVoronoi->fetch_assoc()) {
31         $dataVoronoi[$barisVoronoi['voronoi']][]=
32         $barisVoronoi['neighbor']; }
33 }
34
35 array_push($tempB,$vAwal);
36         $selesai=false;
37         while(count($tempB)!=0){
38                 for($i=0;$i<count($dataVoronoi[$tempB[0]]);$i++){
39                 if(in_array($tempB[0],$tempS)==false){
40                 array_push($tempB,$dataVoronoi[$tempB[0]][$i]);
41                 if($dataVoronoi[$tempB[0]][$i]!=$vAwal){
42                         if($dataVoronoi[$tempB[0]][$i]!=$vTujuan){
43         if(in_array($dataVoronoi[$tempB[0]][$i],$tempT)==false){
44         array_push($tempT,$dataVoronoi[$tempB[0]][$i]);}
45                 }
46         }
47         if ($dataVoronoi[$tempB[0]][$i]==$vTujuan){
48         $selesai=true;}
49 }
```

```
50          else {
51          break;
52          }
53  }
54          if($selesai){break;}
55          array_push($tempS,$tempB[0]);
56          array_shift($tempB);
57  }
58  //Menggunakan algoritme Dijkstra untuk pencarian di tengah V(P)
59          $S=array();
60          $B=array();
61          $J=array();
62          $C=array($nodeAwal,0);
63          $MC=array();
64          $TM=array();
65          $dataJalan=array();
66
67
68          $getJalan = $conn->query("SELECT
69          `simpul_awal` as simpul_awal,
70          `simpul_akhir` as simpul_akhir, '1' as segmen
71          FROM `vcknn` where idv in ('".$vAwal."','".$vTujuan."')
72          ORDER BY simpul_awal ");
73
74          if ($getJalan->num_rows > 0) {
75                  while($barisJalan = $getJalan->fetch_assoc()) {
76                          $dataJalan[$barisJalan['simpul_awal']][]=
77                          array($barisJalan['simpul_akhir'],
78                          $barisJalan['segmen']);
79                          $dataJalan[$barisJalan['simpul_akhir']][]=
80                          array($barisJalan['simpul_awal'],
81                          $barisJalan['segmen']);
82          if(in_array($barisJalan['simpul_awal'],$B)==false){
83                  $B[]=$barisJalan['simpul_awal'];
84                  $J[$barisJalan['simpul_awal']]=array(0,'T');
85          if(in_array($barisJalan['simpul_akhir'],$B)==false){
86                  $B[]=$barisJalan['simpul_akhir'];
87                  $J[$barisJalan['simpul_akhir']]=array(0,'T');
88                  }
89          }
90  }
```

```
//Mengambil data moving interval
        $getMoveInterval = $conn->query("SELECT `awal` as
        simpul_awal, `akhir` as simpul_akhir, `segmen` as segmen
        FROM `mi` where idv in (".join(',',$tempT).")
        ORDER BY simpul_awal ");
                if ($getMoveInterval->num_rows > 0) {
                while($barisMI = $getMoveInterval->fetch_assoc()) {
                        //Mengatur data berdasarkan arah
                        //T = nilai tak hingga
        $dataJalan[$barisMI['simpul_awal']][]=
        array($barisMI['simpul_akhir'],$barisMI[segmen]);
        $dataJalan[$barisMI['simpul_akhir']][]=
        array($barisMI['simpul_awal'],$barisMI['waktu']);

        if(in_array($barisMI['simpul_awal'],$B)==false){
                $B[]=$barisMI['simpul_awal'];
                $J[$barisMI['simpul_awal']]=array(0,'T');
        }
        if(in_array($barisMI['simpul_akhir'],$B)==false){
                $B[]=$barisMI['simpul_akhir'];
                $J[$barisMI['simpul_akhir']]=array(0,'T');
                                }
                        }
                }
echo '<pre>';
                $J[$C[0]]=array(0,0);
                while(count($B)!=0){
                $MC=array(0,0);
                $S[]=$C[0];
                $indeks = array_search($C[0], $B);
                unset($B[$indeks]);

        for($i=0;$i<count($dataJalan[$C[0]]);$i++){
                if(in_array($dataJalan[$C[0]][$i][0],$B)==true){
                $tempSegmen=$dataJalan[$C[0]][$i][1]+$C[1];

                if($J[$dataJalan[$C[0]][$i][0]][1] === 'T'){
        $J[$dataJalan[$C[0]][$i][0]]=array($C[0],$tempSegmen);}
        else if ($tempSegmen <$J[$dataJalan[$C[0]][$i][0]][1]){
        $J[$dataJalan[$C[0]][$i][0]]=array($C[0],$ tempSegmen);}
        }
}
```

```
foreach ($B as $tempB) {
	if($J[$tempB][1] !== 'T'){
		if($MC[0]===0){
		$MC=array($tempB,$J[$tempB][1]);
}
	else if($J[$tempB][1]<$MC[1]){
		$MC=array($tempB,$J[$tempB][1]);
}
	}
}
	if($MC[0]===0){
		break;
	}
	$C=$MC;
}

	$tempNode=$nodeTujuan;
		$rute=array($tempNode);
		$node=$tempNode;
	//Tabel hasil pencarian
	echo '<table>';
		while($tempNode!=$nodeAwal){
		$temp=explode(',',substr(substr
		($tempNode, 0, -1), 1));
			echo'<tr>';
			echo '<td>'.$temp[0].'</td>';
			echo '<td>'.$temp[1].'</td>';
			echo '<td>\''.$tempNode.'\'</td>';
			echo'</tr>';

			$tempNode=$J[$tempNode][0];
			$rute[]=$tempNode;
			$node=$tempNode.','.$node;
		}
	$temp=explode(',',substr(substr($tempNode, 0, -1), 1));
			echo'<tr>';
			echo '<td>'.$temp[0].'</td>';
			echo '<td>'.$temp[1].'</td>';
			echo '<td>\''.$tempNode.'\'</td>';
			echo'</tr>';
			print_r($node);
```

```
173                     print_r($rute);
174                     echo '</table>';
175         echo '</pre>';
176
177 //Akhir perhitungan waktu eksekusi
178 $time_end = microtime(true);
179 $time = $time_end - $time_start;
180 echo "Process Time: {$time}";
```

Pada pencarian yang dilakukan titik awal dan titik akhir dimulai dari ujung simpul awal atau ujung simpul akhir, pencarian tidak dapat dilakukan dari bagian tengah jalan karena tidak sesuai dengan kaidah topologi. Pada Tabel 5.8 merupakan hasil dari pencarian VCKNN. Bagian yang disorot dengan warna kuning merupakan titik awal pencarian dan bagian yang disorot dengan warna merah merupakan titik tujuan pencarian. *Runtime* yang dibutuhkan adalah 0.009 detik dan menghasilkan 9 *split nodes* serta 9 segmen data.

Tabel 5.8 Hasil Pencarian Algoritme VCKNN

Longitude	***Latitude***	**Simpul yang dilalui**
112.646278	-7.977362	'[112.646278,-7.977362]'
112.645721	-7.978134	'[112.645721,-7.978134]'
112.645568	-7.978392	'[112.645568,-7.978392]'
112.644989	-7.9781130009999	'[112.644989,-7.9781130009999]'
112.64386099063	-7.9775737493023	'[112.64386099063,-7.9775737493023]'
112.63920574536	-7.9740104352375	'[112.63920574536,-7.9740104352375]'
112.63672091176	-7.9756931771329	'[112.63672091176,-7.9756931771329]'
112.636726	-7.975903	'[112.636726,-7.975903]'
112.63681	-7.977426001	'[112.63681,-7.977426001]'

Berdasarkan hasil pencarian yang telah dilakukan algoritme Dijkstra dan algoritme VCKNN dapat menemukan jalur dari titik awal menuju titik tujuan. Dengan telah didapatkan jalur pencarian dengan kedua algoritme yang diusulkan, maka akan dilakukan analisis terhadap kinerja dari kedua algoritme dan evaluasi berdasarkan performa dari hasil pencarian yang dilakukan. Dari kedua hasil pencarian ditemukan perbedaan yang akan diperbandingkan pada bab berikutnya untuk ditemukan algoritme yang memiliki performa lebih baik.