KODE PROGRAM DAN MANUAL PERANGKAT LUNAK

Perangkat Lunak Algoritma Genetika dan *Google Maps API* Untuk Penjadwalan Rute Perjalanan Divisi Pemasaran STMIK El Rahma

PENCIPTA

Herdiesel Santoso S.Kom.,S.T.,M.Cs.

Rachmad Sanuri, S.E., M.Eng

DAFTAR ISI

DAFTAR GAMBAR

BAB I
KODE PROGRAM

Proses pertama adalah melakukan inisiasi destinasi yang menjadi lokasi awal dan dan menjadi lokasi tujuan perjalanan, proses ini menggunakan bantuan *google maps api*.

```
[1]     $("#inisiasi").click(function()
[2]     {
[3]             var location = $('#location').val();
[4]             var start_time = $('#start-time').val();
[5]
[6]             if(!!location && !!start_time){
[7]                 var chromosom = new Array();
[8]                 var time_windows = new Array();
[9]                 var event_data   = new Array();
[10]                var latlng = $('#latitude').val() + ', '+ $('#longitude').val();
[11]
[12]
[13]                //Membuat JSON untuk data origin
[14]                var data = '{"location":"'+ $('#location').val() + '","address":"'+
[15]                  $('#address').val() + '","latlng":"'+ latlng +
[16]                  '","time_windows":["2:30","4:20"]}';
[17]                //Push Data Sebagai Array Pertama
[18]                event_data.push(JSON.parse(data));
[19]                //Push Data Koordinat
[20]                chromosom.push(latlng);
[21]
[22]                for(i=1; i<=number_destination; i++){
[23]                    data = JSON.parse($('#body-dest #dest_'+i).val());
[24]
[25]                    event_data.push(data);
[26]                    chromosom.push(data.latlng);
[27]                    time_windows.push(data.time_windows);
[28]                }
[29]
[30]                /* Begin Calculate Distance and duration with Lat and Lng*/
[31]                var service = new google.maps.DistanceMatrixService;
[32]                service.getDistanceMatrix({
[33]                    origins: chromosom,
[34]                    destinations: chromosom,
[35]                    travelMode: google.maps.TravelMode.DRIVING,
[36]                    unitSystem: google.maps.UnitSystem.METRIC,
[37]                    avoidHighways: false,
[38]                    avoidTolls: false
[39]                }, function(response, status) {
[40]                    if (status !== google.maps.DistanceMatrixStatus.OK) {
[41]                        alert('Error was: ' + status);
[42]                    } else {
[43]
[44]                        var originList = response.originAddresses;
[45]                        var destinationList = response.destinationAddresses;
[46]                        var distanceList = new Array();
[47]                        var durationList = new Array();
[48]
[49]                        /* Begin Capture Distance and duration */
[50]                        for (var i = 0; i < originList.length; i++) {
[51]                            var results = response.rows[i].elements;
[52]                            distanceList[i] = new Array();
```

```
[53]
[54]                            for (var j = 0; j < results.length; j++) {
[55]                                if(originList[i] != destinationList[j])
[56]                                {
[57]                                    distanceList[i][j] = results[j].distance.value;
[58]                                    durationList[i][j] = results[j].duration.value;
[59]                                }
[60]                                else{
[61]                                    distanceList[i][j] = null;
[62]                                    durationList[i][j] = null;
[63]                                }
[64]                            }
[65]                        }
[66]                        $('#chromosom').val(JSON.stringify(event_data));
[67]                        $('#time_windows').val(JSON.stringify(time_windows));
[68]                        $('#distanceList').val(JSON.stringify(distanceList));
[69]                        $('#durationList').val(JSON.stringify(durationList));
[70]                        $('#form-event').submit();
[71]                    }
[72]                });
[73]            }
[74]            else{
[75]                alert('Lokasi Asal Tidak Boleh Kosong dan Harus Memilih Lebih Dari 2
[76]                  Destinasi Sekolah!!!');
[77]            }
[78]
[79]        });
```

Gambar 1.1 Kode program proses pengambilan jarak dan waktu tempuh berdasarkan lintang dan bujur menggunakan google maps api

Kode program Gambar 1.1 digunakan untuk mengambil jarak dan waktu tempuh berdasarkan koordinat lintang dan bujur yang dimasukan kedalam aplikasi.

```
[1]       private function pickRandom($dna)
[2]       {
[3]           $choices = $dna;
[4]
[5]           shuffle($choices);
[6]           $choices = implode('-', $choices);
[7]           return '1-' . $choices . '-1';
[8]       }
[9]
[10]      public function initialPopulation($dna, $numberPopulation){
[11]           $initialPopulation = array();
[12]           for ($i = 0; $i < $numberPopulation; $i++) {
[13]              $initialPopulation[$i] = $this->pickRandom($dna);
[14]               while (!in_array($initialPopulation[$i], $initialPopulation)) {
[15]                   $initialPopulation[$i] = $this->pickRandom($dna);
[16]               }
[17]           }
[18]           return $initialPopulation;
[19]       }
```

Gambar 1.2 Kode program proses inisiasi populasi

Kode program Gambar 1.2 digunakan untuk melakukan pengkodean dan inisiasi populasi populasi. Satu skema pengkodean yang biasa digunakan adalah setiap individu berisi satu kromosom yang berisi gen-gen yang merepresentasikan nomor urut lokasi event. Skema pengkodean ini dikenal sebagai *permutation encoding* (pengkodean permutasi). Inisiasi adalah membangkitkan sejumlah individu secara acak atau melalui prosedur tertentu. Dengan menggunakan fungsi random akan dipilih suatu individu dengan memilih secara acak antara bilangan 1 sampai dengan jumlah gen hingga terbentuk suatu kromosom. Kromosom yang sudah dibangkitkan akan menjadi sebuah populasi.

```
[1]     public function rate($dna)
[2]        {
[3]            $dna = explode('-', $dna);
[4]
[5]            $start_time = $this->start_time;
[6]            $penalti = 0;
[7]            $cost    = 0;
[8]
[9]            for ($i = 0; $i < count($dna) - 1; $i++) {
[10]               //Google memberikan waktu tempuh dalam detik, di konversi dalam menit
[11]                dan dibulatkan
[12]               $duration  = round($this->duration[$dna[$i] - 1][$dna[$i + 1] - 1] /
[13]                60);
[14]               $distance  = round($this->distance[$dna[$i] - 1][$dna[$i + 1] - 1] /
[15]                1000);
[16]               $arrival   = date('H:i', strtotime($start_time." +" .$duration.'
[17]                minutes'));
[18]               //Pulang ke tempat asal tidak ada waktu buka atau waktu tutup, tetapi
[19]                  cuma travel time (waktu perjalan saja) yg dicatat
[20]               if($i < (count($dna) - 2))
[21]               {
[22]                   $open  = $this->time_windows[$dna[$i+1]-2][0];
[23]                   $close = $this->time_windows[$dna[$i+1]-2][1];
[24]
[25]                   //Datang Keawalan
[26]                   if(strtotime($arrival) < strtotime($open)) {
[27]                       $penalti += strtotime($open) - strtotime($arrival);
[28]                       $arrival = $open;
[29]                   }
[30]
[31]                   //Datang Terlambat
[32]                   if(strtotime($arrival) > strtotime($close))
[33]                       $penalti += strtotime($arrival) - strtotime($close);
[34]                   else
[35]                       $start_time = date('H:i', strtotime($arrival." + 30
[36]                        minutes"));
[37]
[38]                   //Datang Tidak Terlambat Tapi Waktu Pelayanan Sudah Tutup
[39]                   if(strtotime($start_time) > strtotime($close))
[40]                       $penalti += strtotime($start_time) - strtotime($close);
[41]
[42]               }
[43]               $cost += $duration * $distance;
[44]            }
[45]            //Penalti dalam menit
[46]            $this->penalti = $penalti / 60;
[47]            $this->cost = $cost;
[48]        }
```

Gambar 1.3 Kode program proses perhitungan nilai *fittness*

Setelah terbentuk satu populasi, tahap selanjutnya adalah menghitung nilai *fitness* setiap individu. Oleh karena masalah perjalanan atau kunjungan adalah permasalahan TSP-TW dan tujuannya adalah meminimalkan total biaya, dimana nilai pinalti didapatkan dari waktu berangkat dikurangi dengan jadwal tutup apabila waktu datang tidak dalam waktu antara jadwal buka dan jadwal tutup, namun apabila waktu datang masih dalam waktu antara jadwal buka dan jadwal tutup maka dianggap 0 untuk nilai pinaltinya. Sedangkan, nilai *fitness* didapatkan dari rumus $\frac{1}{(1+pinalti)/60}$. Total biaya adalah perkalian antara jarak tempuh dan waktu tempuh. Gambar 1.3 adalah kode program untuk perhitungan nilai *fitness*.

```
[1]        public function crossover($parent1, $parent2)
[2]        {
[3]            $child1   = array();
[4]            $child2   = array();
[5]            $return   = array();
[6]            $parent1 = explode('-', $parent1);
[7]            $parent2 = explode('-', $parent2);
 [8]
[9]            $stop      = $parent1[1];
[10]           $child1[1] = $parent1[1];
[11]           $child2[1] = $parent2[1];
[12]           $x         = 1;
[13]           while ($parent2[$x] != $stop) {
[14]               $x          = array_search($parent2[$x], $parent1);
[15]               $child1[$x] = $parent1[$x];
[16]               $child2[$x] = $parent2[$x];
[17]           }
[18]
[19]           for ($i = 0; $i < count($parent1); $i++) {
[20]               if (empty($child1[$i])) {
[21]                   $child1[$i] = $parent2[$i];
[22]               }
[23]
[24]               if (empty($child2[$i])) {
[25]                   $child2[$i] = $parent1[$i];
[26]               }
[27]           }
[28]
[29]           ksort($child1, SORT_NUMERIC);
[30]           ksort($child2, SORT_NUMERIC);
[31]           $return[0] = $child1;
[32]           $return[1] = $child2;
[33]
[34]           return $return;
[35]       }
```

Gambar 1.4 Kode program proses *crossover* dengan *cycle crossover*

Metode *crossover* yang digunakan adalah metode *cycle crossover*. Parameter yang dibutuhkan untuk melakukan *crossover* adalah gen dari orang tua 1 dan orang tua 2. Implementasi kode program proses *crossover* dapat dilihat pada Gambar 1.4

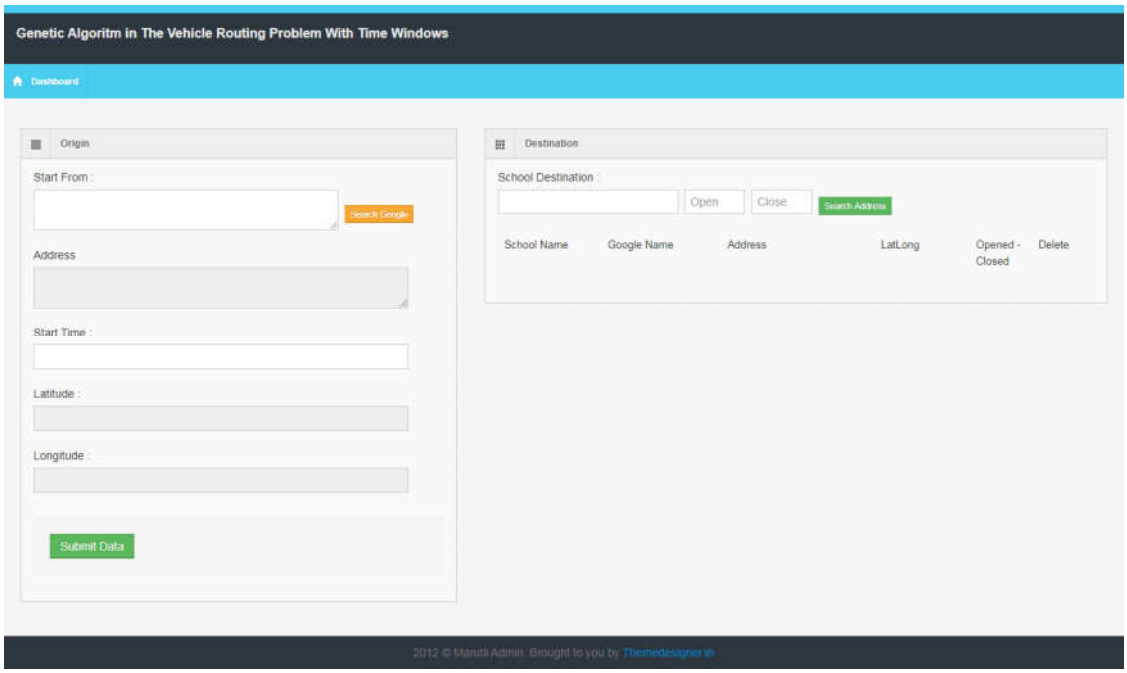
```
[1]        public function mutation($dna)
[2]         {
[3]             $dna    = explode('-', $dna);
[4]             $max    = count($dna) - 1;
[5]             $rand1  = 0;
[6]             $rand2  = 0;
[7]             while ($rand1 == $rand2) {
[8]                 $rand1 = rand(2, $max);
[9]                 $rand2 = rand(2, $max);
[10]            }
[11]            $temp            = $dna[$rand1 - 1];
[12]            $dna[$rand1 - 1] = $dna[$rand2 - 1];
[13]            $dna[$rand2 - 1] = $temp;
[14]
[15]            return $dna;
[16]        }
```

Gambar 1.5 Kode program proses *mutasi* dengan *swap mutation*

Teknik mutasi yang digunakan adalah menggunakan *Swap Mutation*. Proses mutasi dilakukan dengan cara membangkitkan bilangan random untuk setiap individu dalam *mating pool*. Jika random < Pm, maka individu tersebut mengalami mutasi Jika random > Pm, individu tidak mengalami mutasi. Gambar 1.5 adalah implemetasi kode program *swap mutation*. Proses *crossover* dan *mutasi* untuk kasus TSP adalah gen pertama dan terakhir tidak mengalami *crossover* dan *mutasi* karena kedua gen tersebut merupakan lokasi awal dan lokasi akhir perjalanan, sehingga jika ada sejumlah n gen dalam satu populasi, gen yang mengalami *crossover* atau mutasi adalah gen kedua hingga gen n-1. Sementara dari proses crossover dan mutasi dihasilkan DNA anak baru.

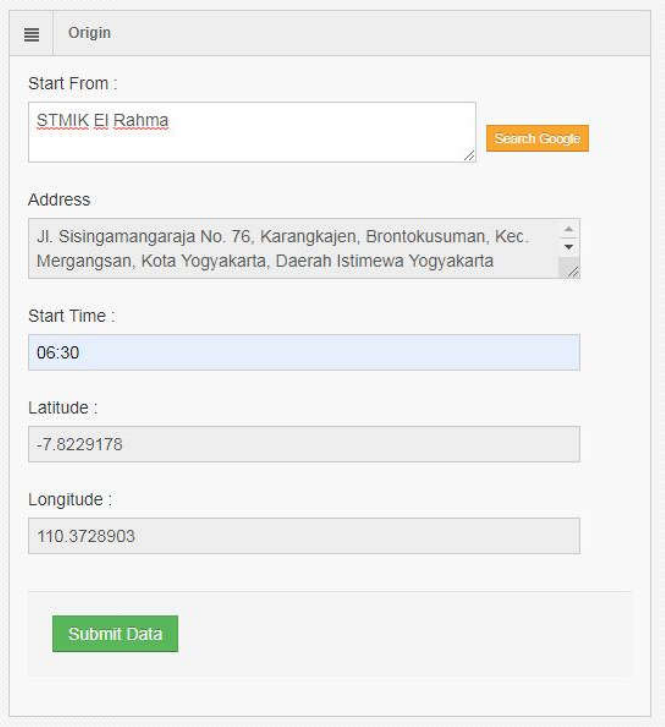
BAB II
MANUAL PENGGUNAAN PROGRAM

1. Buka alamat http://tsptw.stmikelrahma.ac.id/ menggunakan browser mozilla firefox, google chrome atau opera. Maka akan tampil aplikasi Penjadwalan Rute Perjalanan Divisi Pemasaran STMIK El Rahma.



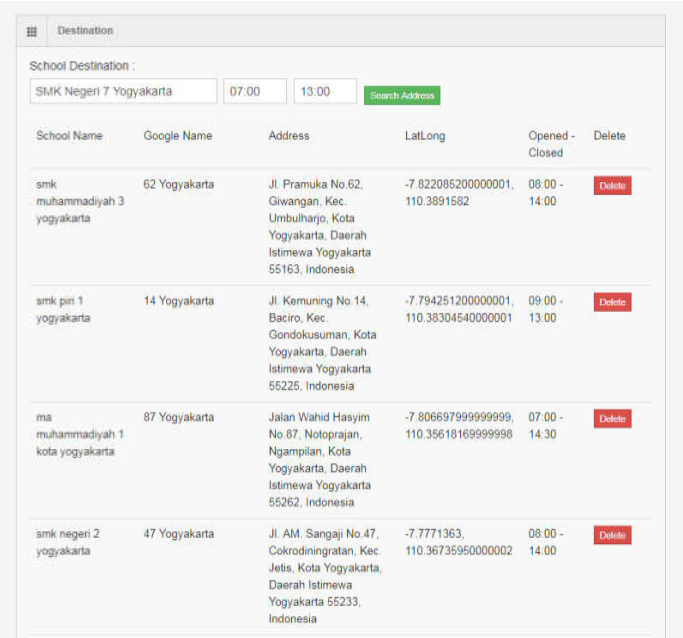
Gambar 2.1 Implementasi Halaman Depan Program Komputer

2. Halaman depan aplikasi terdiri dari menu *input origin* yaitu menu untuk memasukan lokasi awal perjalanan dan menu *input destination* yaitu menu untuk memasukan lokasi yang menjadi tujuan perjalanan dari divisi pemasaran STMIK El Rahma. Pada menu *input origin* pengguna memasukan objek yang menjadi lokasi awal perjalanan dan waktu mulai perjalanan (*start time*). Jika objek tersebut dikenali oleh aplikasi, maka akan muncul nama lokasi, alamat beserta koordinatnya. Selanjutnya pengguna memasukan waktu memulai perjalanan.



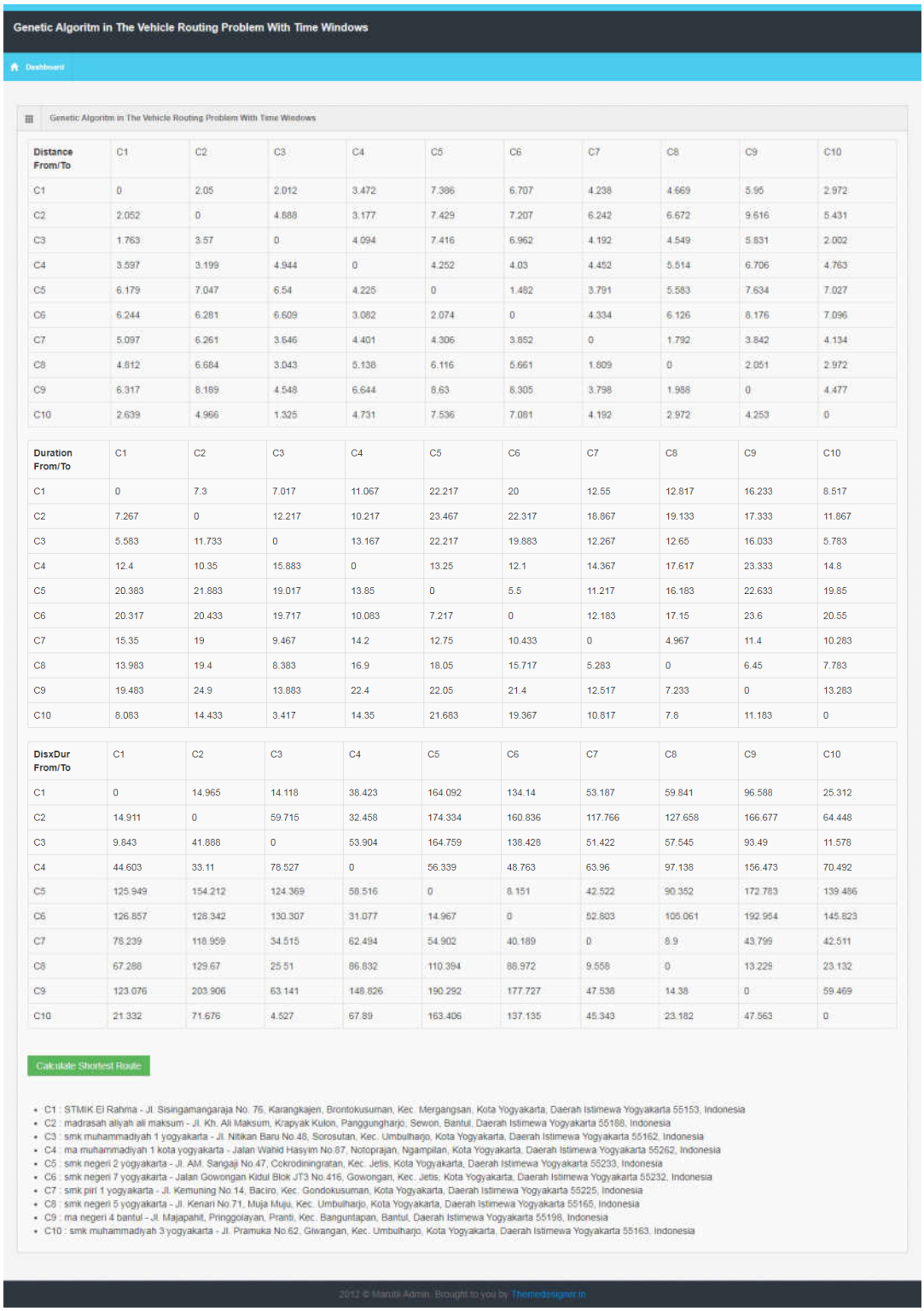
Gambar 2.2 Implementasi Menu Input Lokasi Awal

3. Pada menu *input destination* pengguna perangkat lunak memasukan objek yang menjadi tujuan perjalanan promosi serta waktu ketersediaan kunjungan. Jika objek tersebut dikenali oleh aplikasi, maka akan muncul nama lokasi, alamat beserta koordinat lintang dan bujur. Gambar 1.3 merupakan implementasi menu *input* lokasi tujuan.



Gambar 2.3 Implementasi Menu Input Lokasi Tujuan

4. Selanjutnya pengguna tinggal menekan tombol submit data. Selanjutnya palikasi dengan bantuan *Google maps API* menghitung jarak dan waktu tempuh dari lokasi awal ke masing-masing lokasi tujuan.



Gambar 2.4 Implementasi Menu Jarak dan Waktu Tempuh

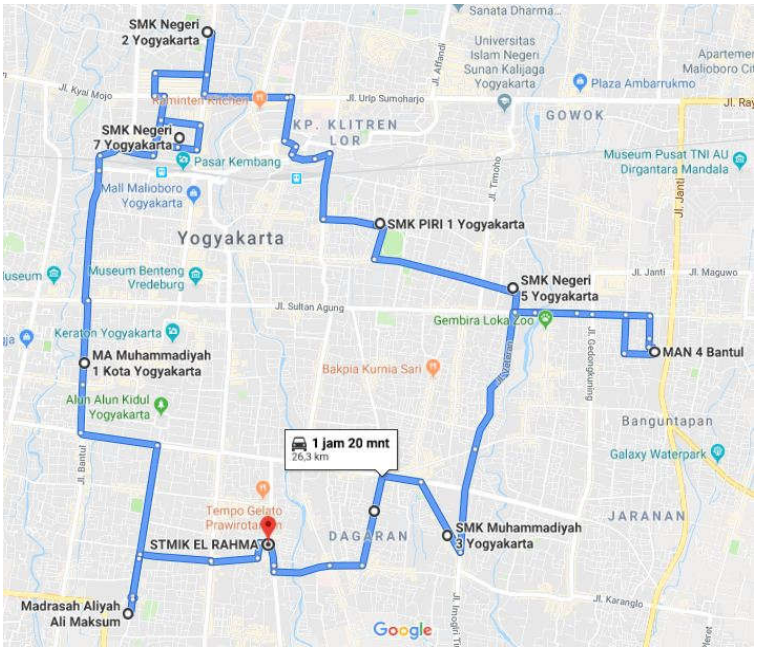
5. Selanjutnya aplikasi melakukan perhitungan menggunakan Algoritma Genetika dengan parameter waktu kedatangan, waktu ketersediaan, jarak, waktu tempuh, jumlah populasi, jumlah generasi, nilai probabilitas *crossover* (Pc) dan probabilitas mutasi (Pm) yang sudah ditentukan. Hasil perhitungan adalah solusi rute terbaik, nilai *fitness*, rekomendasi rute perjalanan lengkap dengan estimasi jarak dan waktu tempuhnya. Implementasi menu hasil perhitungan dengan Algoritma Genetika ditampilkan pada Gambar 1.5.

Solusi Terbaik Yang Ditemukan Adalah **1-8-4-10-5-3-6-9-2-7-1** dengan fitness **0.0040983606557377** dengan **165** generations.

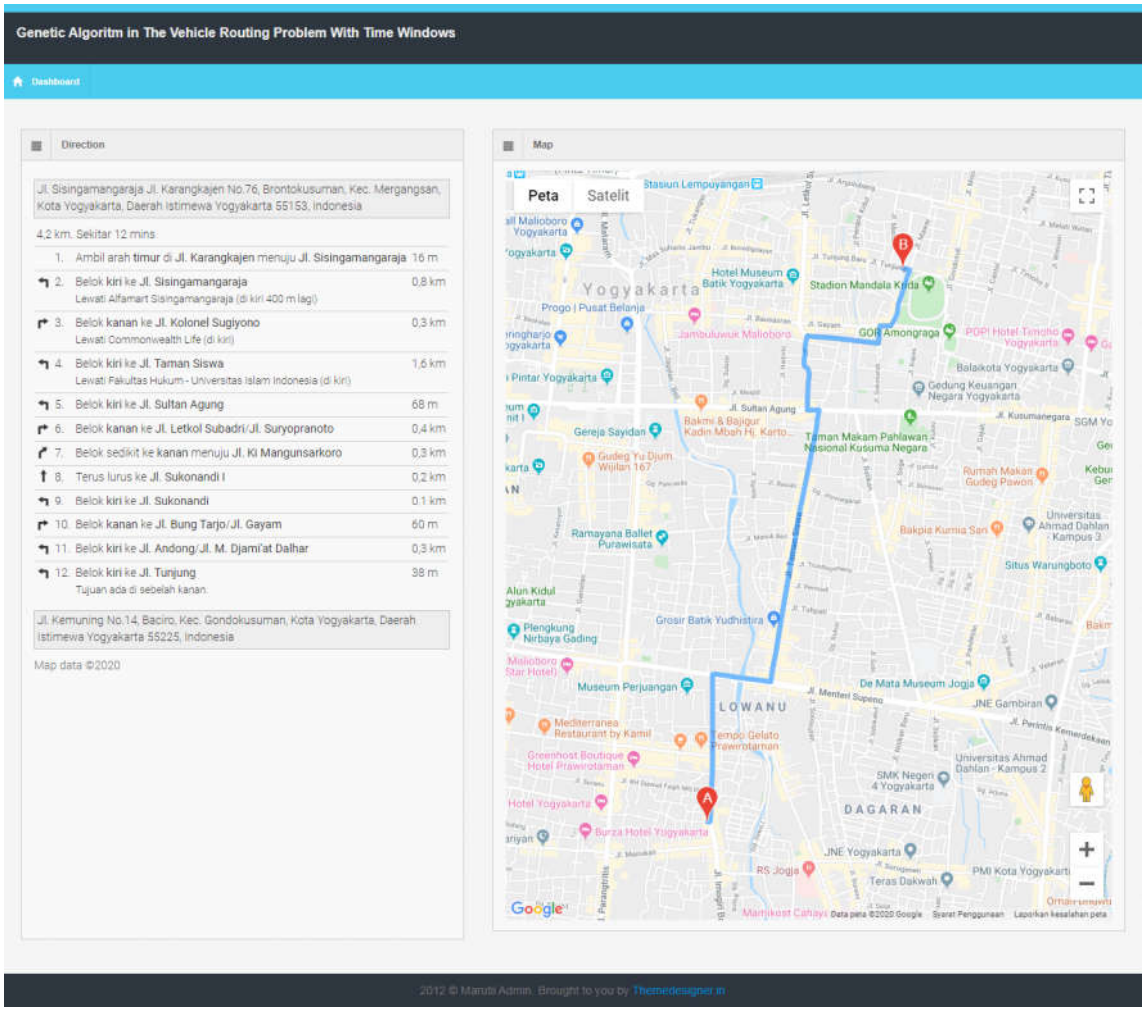
Perjalan Ke -	Rekomendasi Tujuan	Estimasi Jarak	Estimasi Waktu	Rute
1	STMIK El Rahma ke madrasah aliyah ali maksum	2.05 Km	7.3 menit	Rute
2	madrasah aliyah ali maksum ke ma muhammadiyah 1 kota yogyakarta	3.183 Km	10.18 menit	Rute
3	ma muhammadiyah 1 kota yogyakarta ke smk negeri 7 yogyakarta	4.03 Km	12.1 menit	Rute
4	smk negeri 7 yogyakarta ke smk negeri 2 yogyakarta	2.074 Km	7.22 menit	Rute
5	smk negeri 2 yogyakarta ke smk piri 1 yogyakarta	3.791 Km	11.22 menit	Rute
6	smk piri 1 yogyakarta ke smk negeri 5 yogyakarta	1.792 Km	4.97 menit	Rute
7	smk negeri 5 yogyakarta ke man 4 bantul	2.051 Km	6.45 menit	Rute
8	man 4 bantul ke smk muhammadiyah 3 yogyakarta	4.477 Km	13.28 menit	Rute
9	smk muhammadiyah 3 yogyakarta ke smk muhammadiyah 1 yogyakarta	1.322 Km	3.42 menit	Rute
10	smk muhammadiyah 1 yogyakarta ke STMIK El Rahma	1.763 Km	5.58 menit	Rute
Total		26.533 Km	81.72 menit	

Gambar 2.5 Implementasi Menu Hasil Perhitungan

6. Pengguna juga dapat melihat petunjuk arah menuju lokasi tujuan baik rute seluruh lokasi yang akan menjadi tujuan perjalanan seperti Gambar 1.6 atau secara lokasi ke lokasi seperti Gambar 1.6. Rute yang ditampilkan adalah rute tercepat saat pengguna menggunakan aplikasi berdasarkan kondisi *real time* lalu lintas yang direkomendasikan. Rute tersebut dapat berubah sesuai dengan kondisi lalulintas saat pengguna menggunakan aplikasi.



Gambar 2.6 Implementasi Menu Pentunjuk Arah Semua Lintasan



Gambar 2.7 Implementasi Menu Petunjuk Arah Antara Lokasi