# Aplikasi Desktop Pencarian Rute Jalan dengan Algoritma Simulated Annealling

**Andi Wahju Rahardjo Emanuel, Allen F. Aritonang**

Jurusan Teknik Informatika
Fakultas Teknologi Informasi, Universitas Kriten Maranatha
Jl. Prof. Drg. Suria Sumantri no. 65 Bandung 40164
Email : andi.wre@eng.maranatha.edu, allen.aritonang@gmail.com

*Abstract*

*Map is a tool to help people on finding a location. Map contains symbols that represented certain area or location and also contains lines that connected from one location with another. Integrating technologies in map application is more useful than conventional map. The goal of this application is to help map user to find location or streets. Another goal is to know the performance of Simulated Annealing algorithm comparing with another search algorithm. So, Simulated Annealing algorithm was implemented in this application. This application was build using Java Technologies that running in Windows platform and the programming language is Java. This map application data acquired from OpenStreetMap and subsequently the data was converted to PostGIS database. From all perfomance test, Simulated Annealing algorithm have an accurary is about 50%. Even though this algorithm haven't much accuration than the others, but this algorithm is fit to be implemented in any map application.*

*Keywords : Simulated Annealing, OpenStreetMap, PostGIS.*

## 1. Pendahuluan

Kendala yang dimiliki oleh peta konvensional adalah tidak dapat diperbaharui dalam waktu singkat dan memiliki waktu kadaluwarsa. Hal ini mengakibatkan pengguna peta tidak dapat mengetahui hal-hal seperti perubahan nama wilayah, perubahan nama kota, penambahan wilayah-wilayah baru secara *real-time*. Kendala lainnya adalah untuk mencari jalur terpendek dari kota asal ke kota tujuan pada peta masih menggunakan cara manual, yaitu dengan memperhitungan skala pada peta. Cara ini tentu saja sangat merepotkan, jika efisiensi waktu menjadi tuntutan pengguna peta.
Untuk mengatasi kendala-kendala pada peta konvensional, peta dibuat pada media elektronik seperti pada *GPS*, *PDA* ataupun pada *PC*. Saat ini peta dibuat semirip mungkin dengan keadaan geografis yang sesungguhnya, yaitu dengan citra satelit.

Agar interaktif peta dibuat dengan menggunakan bahasa pemrograman sehingga pengguna peta dapat mengetahui informasi wilayah atau lokasi hanya dengan mengeklik satu tombol saja. Dengan algoritma yang tepat guna jalur terpendek menuju suatu kota dapat diketahui dengan tepat.
Saat ini banyak algoritma yang digunakan pada peta, khususnya dalam pencarian jalur terpendek. Algoritma-algoritma ini mencari jalur terpendek (*the shorttest*

*path*) dalam peta yang direpresentasikan ke dalam graf berbobot. Dengan mengintegrasikan algoritma pencarian dan sistem informasi geografis ke dalam aplikasi peta, maka kendala-kendala pada peta konvensional dapat diatasi.

## 2. Landasan Teori Algoritma Simulated Annealing

Algoritma *SA* diperkenalkan oleh Metropolis *et al*. Pada tahun 1953, dan aplikasinya dalam masalah optimasi dilakukan pertama kali oleh Kirkpatrick *et al*. Tahun 1983. Algoritma ini beranalogi dengan proses *annealing* (pendinginan) yang diterapkan dalam pembuatan material *glassy* (terdiri dari butir kristal). Dari sisi ilmu fisika, tujuan sistem ini adalah untuk meminimasi energi potensial. Fluktuasi kinematika acak menghalangi sistem untuk mencapai energi potensial yang minimum global, sehingga sistem dapat terperangkap dalam sebuah keadaan minimum lokal. Dengan menurunkan temperatur sistem, diharapkan energi dapat dikurangi ke suatu level yang relatif rendah. Semakin lambat laju pendinginan ini, semakin rendah pula energi yang dapat dicapai oleh sistem pada akhirnya. Algoritma Metropolis yang kemudian berkembang sebagai algoritma *Simulated Annealing* adalah sebagai berikut.

```
Function Metropolis( current_state, temperature )
        T ← temperature
        E ← getEnergy( current_state )
        E'← Random( E )
        ΔE ← E' – E

        If ( ΔE ≤ 0 )
                Return getState( E' )
        Else
                If( ΔE > 0 )
                        If (ΔE = e^(−ΔE/T) )
                                Return getState( ΔE )
                        End If
                End If
        End If
End Function
```

### 2.1 Penerapan Algoritma SA pada Aplikasi Pencarian Rute

Berdasarkan teori yang telah dijelaskan pada bagian sebelumnya, penerapan algoritma *SA* dapat diaplikasikan pada masalah pencarian rute terpendek dalam graf berbobot (*weighted graph*). Hal-hal yang harus diperhatikan dalam mengaplikasikan algoritma *SA* yaitu temperatur awal, pembangkitan bilangan acak dan vertex tetangga (*adjacent/neigbour*).

Sebelum menerapkan algoritma *SA*, lingkungan pencarian harus dibuat terlebih dahulu. Lingkungan pencarian berarti peta yang terdapat kota-kota yang akan dicari rute terpendeknya. Peta tersebut harus direpresentasikan ke dalam suatu data yang terstruktur seperti graf berbobot. Penggunaan graf berbobot paling cocok digunakan untuk melakukan pencarian rute, hal ini dikarenakan bobot yang

terdapat pada *edge* yang menghubungkan *vertex* dapat digunakan untuk mencari rute terpendek.

Dari algoritma *SA* yang telah dijelaskan pada bagian sebelumnya, algoritma tersebut dapat dimodifikasi seperti berikut.

```
Function SIMULATED-ANNEALING ( problem, schedule )
                                                    returns a solution state
inputs : problem, a problem
         schedule, a mapping from time to "temperature"
local variables :
              current, a node
              next, a node
               T, a "temperature" controlling the probability of downward steps
   current ← MAKE-NODE( INITIAL-STATE[ problem ] )
   for t ← 1 to INFINITY do
   T ← schedule[ t ]
   if T = 0 then return current
       next ← a randomly selected successor of current
       ΔE ← VALUE[ next ] – VALUE[ current ]
       if ΔE > 0 then
                current ← next
       else
                current ← next only with probablity e^(-ΔE/T)
```

Setelah algoritma *SA* menemukan solusi pada setiap iterasi, maka solusi-solusi (berupa *vertex*) tersebut disimpan pada daftar atau *list* yang digunakan untuk mengetahui *path* yang telah dilalui. *List* yang berisi *path* atau jalur terpendek, dapat digunakan untuk menggambar jalur pada peta.

## 2.2 Simulasi Penerapan Algoritma SA pada Aplikasi Pencarian Rute

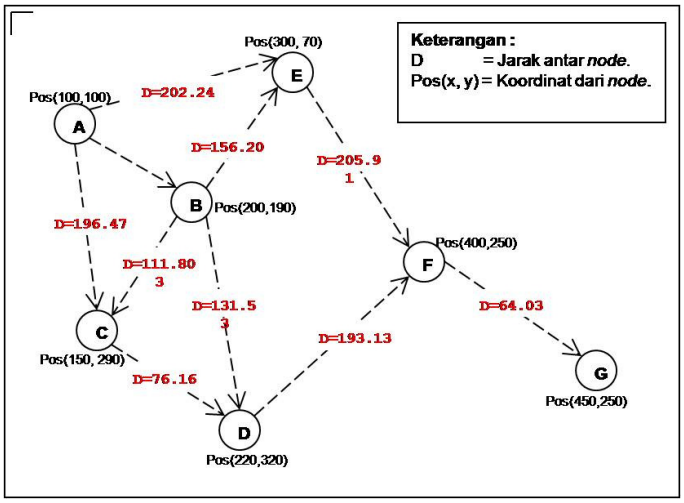
Untuk menerapkan algoritma *SA* ke dalam aplikasi pencarian rute, perlu dilakukan simulasi algoritma tersebut dengan contoh kasus yang sederhana. Maksud dan tujuan dari simulasi ini agar cara kerja algoritma ini dapat diketahui secara mendetail. Simulasi akan dilakukan dengan menggunakan lingkungan pencarian berupa graf berarah dengan arah tunggal dengan konfigurasi sebagai berikut.

***Tabel 1** Konfigurasi Graf Berarah*

| **Lokasi / Node** | **Koordinat Posisi** | |
|---|---|---|
| | **X** | **Y** |
| A | 100 | 100 |
| B | 200 | 190 |
| C | 150 | 290 |
| D | 220 | 320 |
| E | 300 | 70 |

| Lokasi / Node | Koordinat Posisi | |
|---|---|---|
| | X | Y |
| F | 400 | 250 |
| G | 450 | 250 |

Tabel 1 merupakan daftar dari *node-node* yang terdapat dalam graf. Node-node tersebut memiliki koordinat posisi masing-masing. Sistem koordinat yang digunakan adalah sistem koordinat monitor (bukan sistem koordinat kartesian).




**Gambar 1** Konfigurasi Graf Berarah

Gambar 1 memperlihatkan konfigurasi graf berarah yang digunakan sebagai lingkungan pencarian dari algoritma *SA*. Setiap *vertex* memiliki *adjacent* atau tetangga. Tetangga ditentukan oleh arah dari garis panah. Asumsi titik awal pencarian dimulai dari *node/vertex* A dan titik tujuan pecarian adalah *vertex G*. Jarak antar graf berarah pada gambar diatas didapat dari fungsi persamaan berikut ini:

$$f(x,y,x',y') = \sqrt{|x'-x|^2 + |y'-y|^2}$$

## a). Daftar Istilah

Daftar istilah ini dapat digunakan sebagai panduan untuk melakukan simulasi algoritma *SA*. Berikut ini merupakan istilah-istilah penting yang sering digunakan dalam simulasi ini.

***Tabel 2** Tabel Daftar Istilah dalam Simulasi*

| Istilah | Deskripsi | Analogi |
|---|---|---|
| Graf (*Graph*) | Representasi data yang struk-turnya terdiri dari *node/vertex* dan *edge*, *node/vertex* terhu-bung oleh *edge*. | Graf biasanya digunakan untuk merepresentasikan sebuah peta yang didalamnya terdapat kota-kota(*vertex*) dan jalan(*edge*). |

| Istilah | Deskripsi | Analogi |
|---|---|---|
| *Node / Vertex* | Merupakan bagian dari graf yang dan merupakan tempat penyimpanan data. Setiap *node/vertex* memiliki nama a-tau identitas yang unik. | *Node/Vertex* digunakan untuk merepresentasikan suatu kota atau lokasi dalam peta. Pada aplikasi ini *node/vertex* digunakan untuk menyimpan nama dan koordinat posisi. |
| *Edge* | Merupakan penghubung (*connector*) antara suatu *ver-tex* dengan *vertex* lainnya. Jumlah *edge* menentukan jumlah *adjacent* atau tetangga dari suatu *vertex*. | *Edge* digunakan untuk merepresentasikan sebuah jalan yang menghubungkan lokasi dengan lokasi lainnya dalam peta. |
| *Adjacent/ Neighbor* | Istilah untuk tetangga dari suatu *vertex.* Jumlah *adjacent* ditentukan oleh jumlah *edge* yang terkoneksi dengan suatu *vertex*. | *Adjacent* digunakan untuk me-representasikan pilihan rute tu-juan ke kota atau lokasi selan-jutnya. |
| Suhu / Temperatur Awal | Suhu awal sangat berperan penting dalam algoritma ini ka-rena suhu awal menentukan seberapa banyak langkah yang dipelukan untuk mene-mukan suatu solusi. | - |
| Persentase Penurunan Suhu | Persentase ini digunakan oleh algoritma *SA* untuk mengurangi suhu awal dengan persentase tertentu. Semakin kecil persentase pengurangan suhu maka semakin banyak langkah yang akan dilakukan oleh algoritma ini. | - |
| *Probability of Acceptance* | Rumus yang digunakan oleh algoritma *SA* untuk menentu-kan arah tujuan solusi berikut-nya. | Rumus ini dapat dianalogikan sebagai otak atau keputusan dari seseorang yang sedang menentukan arah tujuan dalam sebuah peta. |
| Energi ( E ) | Merupakan nilai yang didapat dari delta E pada vertex yang sudah dievaluasi. | Energi dalam aplikasi ini diana-logikan sebagai speedometer dari si pencari rute. Energi digu-nakan untuk menyimpan jarak yang sudah ditempuh oleh algoritma ini. |
| Delta E (ΔE ) | Selisih energi dari vertex saat ini yang sedang dievaluasi de-ngan energy *vertex* selanjut-nya yang akan dievaluasi. | Delta E dianalogikan sebagai jarak antar lokasi. |
| Distribusi Peluang Seragam (*Uniform*) | Distribusi peluang yang memi-liki peluang yang sama. Dalam algoritma ini, peluang terpilih-nya bilangan acak memiliki pe-luang yang sama dari setiap pembangkitan bilangan acak. | - |

## b). Daftar Rumus

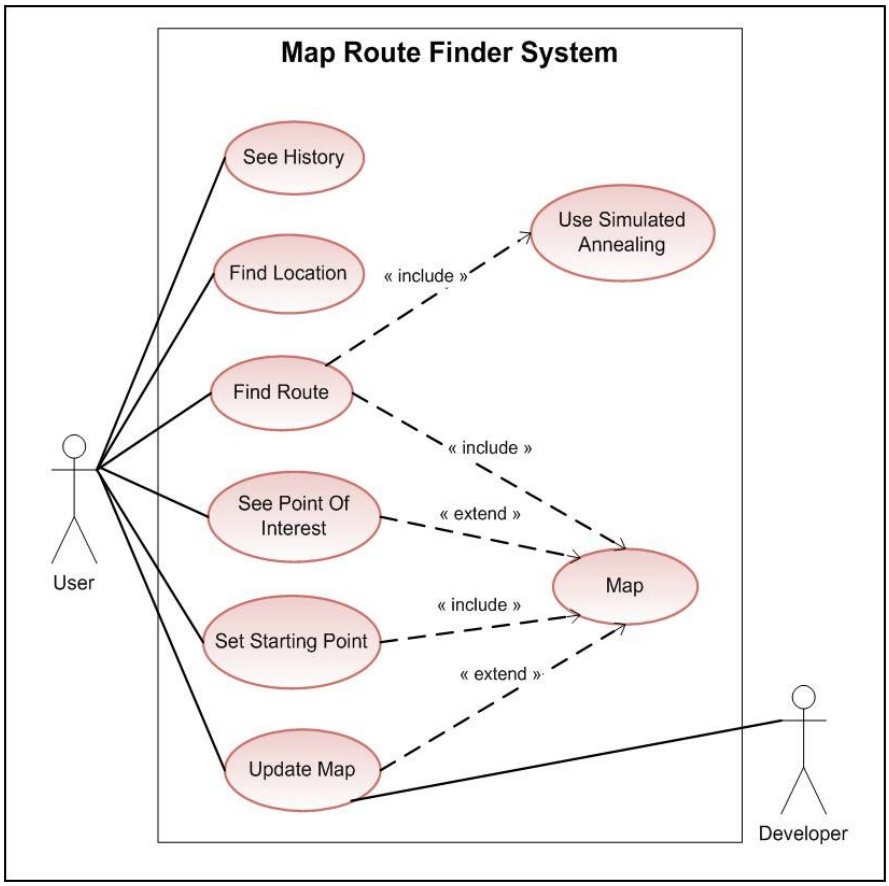
Perhitungan pada setiap iterasi dalam algoritma ini berdasarkan beberapa rumus yang telah dijelaskan dalam penjelasan teori pada bagian sebelumnya. Berikut ini adalah rumus-rumus yang digunakan dalam algoritma *SA*.

***Tabel 3*** *Tabel Rumus Algoritma* SA

| No. | Rumus | Variabel | Penjelasan |
|---|---|---|---|
| 1. | $k(n) = e^{(-p(n))}$<br>**Persamaan 3.3** Konstanta Penurunan Suhu | *n = iterasi ke n*<br>*e = eksponensial*<br>*p = persentase pengurangan suhu* | Rumus ini digunakan untuk menghitung besar penurunan suhu (proses pendinginan) pada setiap iterasi. |
| 2. | $kT = T_0 . k$<br>**Persamaan 3.4** Temperatur | $kT$ *= Temperatur saat ini*<br>$T_0$ *= Temperatur awal*<br>*k = konstanta penurunan suhu* | Rumus ini digunakan untuk menghitung suhu saat ini. |
| 3. | $\Delta E = E_{new} - E_{old}$<br>**Persamaan 3.5** Perubahan Energi | $E_{new}$ *= energi dari adjacent node*<br>$E_{old}$ *= energi dari current node* | Rumus berikut diguna-kan untuk menghitung perubahan energi pada setiap partikel yang ber-dekatan dalam hal ini di-analogikan dari verteks / node. |
| 4. | $p = e^{(-\Delta E/kT)}$<br>**Persamaan 3.6**<br>*Probability of Acceptance* | $\Delta E$ *= perubahan suhu*<br>*kT = temperatur saat ini* | Merupakan perhitungan probabilitas pada setiap iterasi dan menentukan arah evaluasi node pada setiap iterasi |

## 3. Perancangan dan Implementasi Aplikasi Desktop Pencarian Rute Jalan dengan Algoritma Simulated Annealing

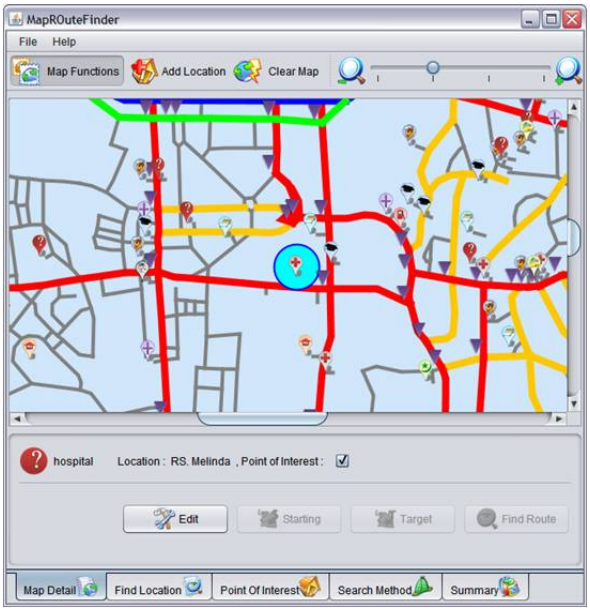
### 3.1 Use Case




**Gambar 6** *Use Case Diagram*

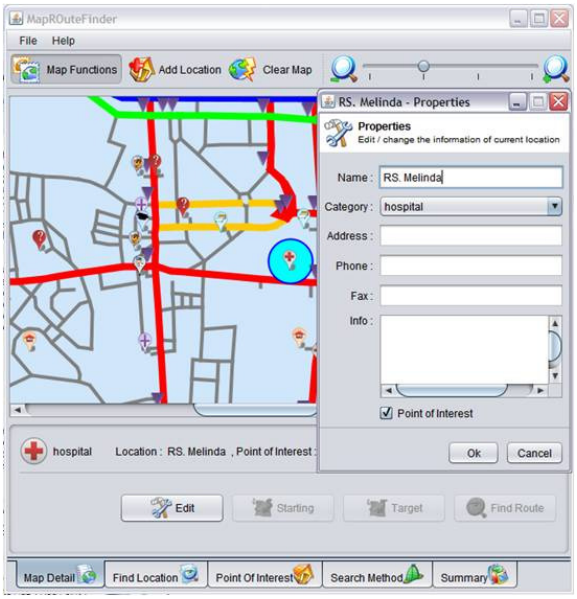
Gambar 6 memperlihatkan *use case diagram* dari aplikasi pencarian rute jalan. Aktor yang terlibat dalam sistem adalah pengguna atau *user* dan pengembang aplikasi atau *developer*. Aktivitas yang dapat dilakukan oleh *user* yaitu melihat *history* dari hasil pencarian rute yang telah dilakukan. *User* dapat mencari informasi lokasi seperti jalan atau tempat dalam peta. *User* dapat melakukan pencarian rute dari suatu titik lokasi ke titik lokasi yang dituju. Pencarian rute ini menggunakan algoritma *Simulated Annealing*. *User* juga dapat menentukan *point of interest* atau titik-titik yang paling sering dikunjungi orang seperti rumah sakit, *mall*, *supermarket*, dan lain-lain. *User* dapat melakukan aktivitas menentukan *starting point* sehingga memudahkan *user* untuk melakukan pencarian rute dan *user* tidak perlu lagi menentukan titik awal pencarian rute. Agar informasi peta dapat *up to date*, *user* dan *developer* dapat melakukan *update* informasi pada titik lokasi yang dinginkan dan dapat pula menambahkan lokasi dalam peta.

### 3.2 Implementasi User Interface Design



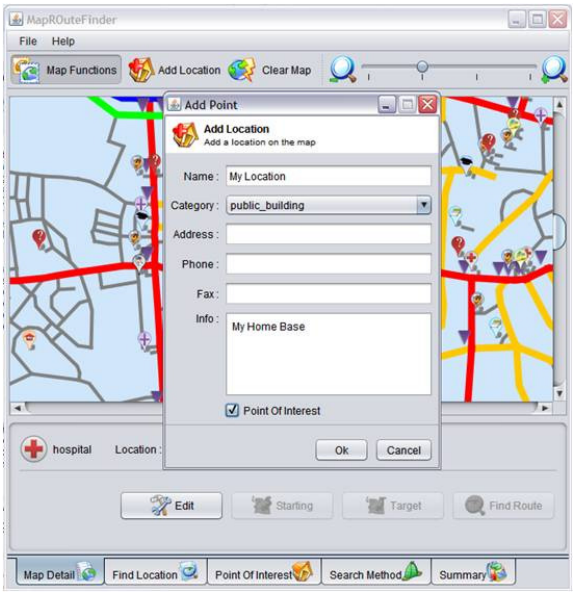

**Gambar 7** *Main Form*

Gambar 7 memperlihatkan tampilan form utama dari aplikasi ini. Form ini ditampilkan ketika proses loading component telah selesai. Pada form ini terdapat tab find location, tab ini dapat digunakan oleh pengguna untuk mencari lokasi atau jalan.



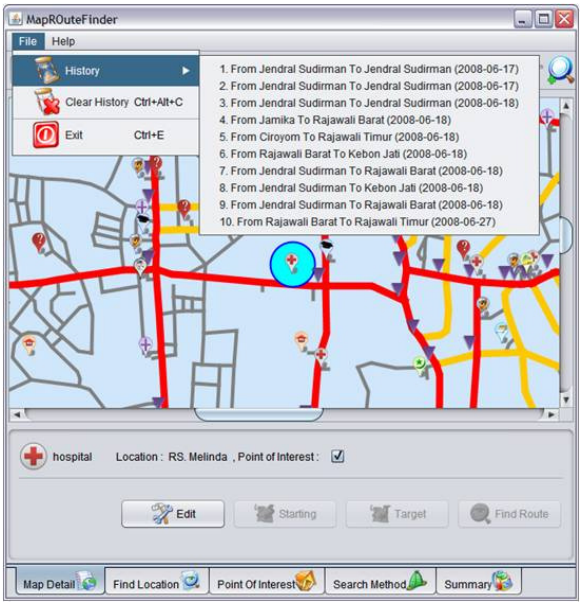

**Gambar 8** *Properties*

Gambar 8 memperlihatkan tampilan form properties dari suatu lokasi. Form ini berfungsi untuk melihat atau mengubah data dari suatu lokasi.



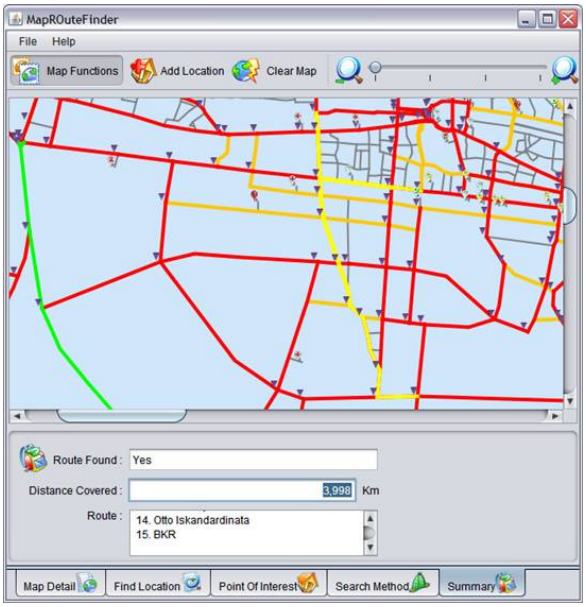

**Gambar 9** *Add Location*

Gambar 9 memperlihatkan realisasi dari perancangan form add point. Form ini dapat digunakan oleh pengguna untuk menambah lokasi dalam peta.




**Gambar 10** *History*

Gambar 10 memperlihatkan realisasi dari fitur *history*. Menu ini berguna untuk menampilkan kembali hasil pencarian yang telah dilakukan oleh pengguna




**Gambar 11** Hasil Pencarian Rute

Gambar 11 memperlihatkan hasil pencarian rute yang ditunjukan oleh garis berwarna kuning dalam peta. Kemudian status dan hasil pencarian dirangkum dan ditampilkan dalam *tab summary*.

## 4. Evaluasi

Berdasarkan hasil pengujian dan pengamatan yang telah dilakukan secara keseluruhan dimana kemungkinan penemuan solusi dari algoritma ini adalah dibawah 50% yaitu sebesar 37,62% (dari 101 kali pengujian, solusi pencarian rute yang ditemukan sebanyak 38 kali, solusi pencarian rute yang tidak ditemukan sebanyak 63 kali), dapat disimpulkan algoritma ini memiliki kekurangan dalam penemuan solusi. Sehingga pada penerapan pada aplikasi pencarian rute, Pengguna akan dihadapkan pada penemuan rute tujuan yang kadang-kadang ditemukan dan kadang-kadang tidak ditemukan. Sehingga untuk mengoptimalkan algoritma ini dapat dilakukan dengan menjalankan algoritma ini secara simultan dan mencari hasil jarak tempuh yang paling pendek.

## 5. Kesimpulan

### 5.1 Kelemahan Algoritma Simulated Annealing

Kelemahan dari algoritma ini adalah sebagai berikut.

- Algoritma ini memiliki akurasi yang buruk. Berdasarkan hasil pengujian yang telah dilakukan, algoritma *SA* memiliki akurasi sebesar 37,62% (akurasi *SA* dalam menemukan solusi / target).
- Algoritma *SA* tidak bisa menghasilkan rute terpendek jika rute jalan memiliki dua arah sehingga memungkinkan algoritma ini untuk mengunjungi kembali rute yang sudah dilalui sebagai perbaikan *state* atau kondisi dan menghiraukan hasil pencarian rute yang lebih panjang.

- Algoritma *SA* tidak dapat secara pasti dalam menentukan jarak terpendek. Walaupun rute menuju tujuan telah ditemukan namun jarak dan rute yang telah ditempuh dapat memiliki hasil yang bervariasi. Hal ini terbukti dari hasil perbandingan dengan algoritma *A Star,* Algoritma ini memiliki rute dan jarak tempuh yang lebih panjang. Fungsi heuristik serta perhitungan estimasi jarak dan jarak yang telah ditempuh membuat algoritma *A Star* lebih ungul daripada algoritma *Simulated Annealing* yang hanya menggunakan fungsi *meta-heuristic* dan bergantung pada kondisi yang acak.
- Hasil pencarian rute yang dihasilkan memiliki hasil yang bervariasi, hal ini membuat rute terpendek yang diharapkan menjadi tidak konsisten.

## 5.2 Solusi dari Kelemahan Algoritma Simulated Annealing

Kelemahan dari algoritma *Simulated Annealing* dapat diatasi dengan mengoptimisasikan algoritma ini dengan menjalankan algoritma ini secara simultan kemudian memilih algoritma mana saja yang ditemukan solusinya lalu memilih algoritma yang menghasilkan rute yang paling pendek.

## Daftar Pustaka

[Ada05] Adamson, Chris dan Marinacci, Joshua. 2005. *Swing Hacks*. California : O'Reilly.

[Car07] M. Carrano, Frank. 2007. *Data Structure and Abstraction With Java. 2nd Edition*. New Jersey : Pearson Prentice Hall.

[Goo07] Google Code (2007). *aima-java – Google Code. Retrieved* March 28th, 2007, *from* Google Code : http://code.google.com/p/aima-java/

[Knu99] Knudsen, Jonathan. 1999. *Java 2D Graphics*. California : O'Reilly.

[Kus03] Kusumadewi, Sri. 2003. *Artificial Intelligence* : Teknik dan Aplikasinya. Yogyakarta : Graha Ilmu.

[Ope07] OpenStreetMap. (2007). *Main Page - OpenStreetMap*. *Retrieved* March 28th, 2007 ,*from* OpenStreetMap Wiki: http://wiki.openstreetmap.org.

[Pan02] Panggabean, Henry. 2002. Jurnal Ilmu Komputer : Penjadwalan *Job Shop* Statik dengan Algoritma *Simulated Annealing*. Bandung : Jurusan Ilmu Komputer, FMIPA Universitas Katolik Parahyangan .

[Rus07] Russel, Stuart dan Norvig, Peter. 2007. *Artificial Intelligence : A Modern Approach*. *2nd Edition*. New Jersey : Pearson Prentice Hall.

[Wir07] Wiraguna, Adhitya. 2007. Aplikasi PDA Perencanaan Perjalanan Kota Bandung dengan Teknologi *OpenStreetMap* dan Algoritma Pencarian A*. Bandung : Fakultas Teknologi Informasi Universitas Kristen Maranatha.