Media Informatika, Vol. 6, No. 1, Juni 2008, 39-55
ISSN: 0854-4743

PENYELESAIAN MASALAH *TRAVELING SALESMAN PROBLEM* DENGAN JARINGAN SARAF *SELF ORGANIZING*

Sukma Puspitorini
Program Studi Teknik Informatika STMIK Nurdin Hamzah Jambi
e-Mail: sukm4pit@yahoo.com

ABSTRAK

Traveling Salesman Problem *(TSP) pertama kali diperkenalkan oleh Rand pada tahun 1948, reputasi Rand membuat TSP dikenal dengan baik dan menjadi masalah yang populer. TSP merupakan persoalan yang mempunyai konsep sederhana dan mudah dipahami. Pada TSP, optimasi yang diinginkan agar ditemukan rute perjalanan terpendek untuk melewati sejumlah kota dengan jalur tertentu sehingga setiap kota hanya terlewati satu kali dan perjalanan diakhiri dengan kembali ke kota semula. Pendekatan dengan menggunakan Jaringan Saraf* Kohonen Self Organizing *memberikan solusi atau penyelesaian dalam perhitungan waktu yang lebih singkat dibandingkan dengan sejumlah algoritma lain yang diterapkan pada komputer dalam bentuk program.*

Tujuan yang ingin dicapai adalah mengaplikasikan metode kohonen untuk mensimulasikan dan menyelesaikan permasalahan traveling salesman problem untuk mendapatkan rute perjalanan terpendek.

Proses penelitian menggunakan data input berupa jumlah kota, koordinat kota, bobot jaringan dan parameter pelatihan, yang kemudian akan diolah menggunakan Jaringan Saraf Kohonen Self Organizing *untuk menentukan rute perjalanan terpendek.*

Kata kunci: Traveling Salesman Problem, Jaringan Syaraf Tiruan, Kohonen Self Organizing

1. PENDAHULUAN

Travelling Salesman Problem (TSP) dikenal sebagai salah satu permasalahan optimasi klasik yang berat untuk dipecahkan secara konvensional. Penyelesaian eksak terhadap persoalan ini akan melibatkan algoritma yang mengharuskan untuk mencari kemungkinan semua solusi yang ada. Sebagai akibatnya, kompleksitas waktu dari eksekusi algoritma ini akan menjadi *eksponensial* terhadap ukuran dari masukan yang diberikan. Permasalahan yang melibatkan algoritma demikian lebih dikenal sebagai permasalahan yang bersifat *Nondeterministic Polynomial-time Complete (NP-Complete).*

Travelling Salesman Problem melibatkan seorang *travelling salesman* yang harus melakukan kunjungan ke sejumlah kota dalam menjajakan produknya. Rangkaian kota-kota yang dikunjungi harus membentuk suatu jalur sedemikian sehingga kota-kota tersebut hanya boleh dilewati tepat satu kali dan kemudian kembali lagi ke kota awal. Penyelesaian terhadap permasalahan TSP ini adalah untuk memperoleh jalur terpendek. Penyelesaian eksak terhadap masalah TSP mengharuskan untuk melakukan perhitungan terhadap semua kemungkinan rute

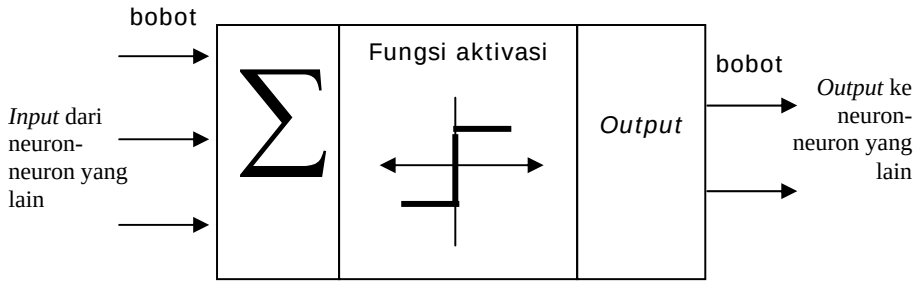
yang dapat diperoleh, kemudian memilih salah satu rute yang terpendek. Untuk itu jika terdapat *n* kota yang harus dikunjungi, maka diperlukan proses pencarian sebanyak *n!/2n* rute. Dengan cara ini waktu komputasi yang diperlukan akan jauh meningkat seiring dengan bertambahnya jumlah kota yang harus dikunjungi. Sebagai ilustrasi, untuk 10 kota saja, diperlukan proses pencarian jalur sebanyak 181.440 rute. Penjelasan ini menunjukkan bahwa solusi eksak terhadap masalah TSP sangat sulit dilakukan. Oleh karena itu dibuat suatu perangkat lunak yang melibatkan Jaringan Saraf *Self Organizing* dengan metode pembelajaran Kohonen untuk memberikan pemecahan alternatif yang lebih baik.

2. LANDASAN TEORI

2.1 Jaringan Saraf Tiruan

Jaringan Saraf Tiruan adalah merupakan salah satu representasi buatan dari otak manusia yang selalu mencoba mensimulasikan proses pembelajaran pada otak manusia tersebut. Istilah buatan di sini digunakan karena jaringan saraf ini diimplementasikan dengan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses pembelajaran.

Seperti halnya otak manusia yang terdiri dari sekumpulan sel saraf (*neuron*), jaringan saraf juga terdiri dari beberapa *neuron* dan terdapat hubungan antara *neuron-neuron* tersebut. *Neuron-neuron* tersebut akan memindahkan informasi yang diterima melalui sambungan keluarnya menuju *neuron-neuron* yang lain. Pada jaringan saraf, hubungan ini dikenal dengan nama bobot**.** Informasi tersebut disimpan pada suatu nilai tertentu pada bobot tersebut. Gambar 1 menunjukkan struktur *neuron* pada Jaringan Saraf.



Gambar 1**.** Struktur *neuron* jaringan saraf

Pada neuron jaringan syaraf tiruan, informasi (disebut pula dengan *input*) akan dikirim ke *neuron* dengan bobot kedatangan tertentu. *Input* ini akan diproses oleh suatu fungsi perambatan yang akan menjumlahkan nilai-nilai semua bobot yang datang. Hasil penjumlahan ini kemudian akan dibandingkan dengan suatu nilai ambang (*threshold*) tertentu melalui fungsi aktivasi setiap *neuron*. Apabila *input* tersebut melewati suatu nilai ambang tertentu, maka *neuron* tersebut akan diaktifkan. Apabila *neuron* tersebut diaktifkan, maka *neuron* tersebut akan mengirimkan keluaran (disebut dengan *output*) melalui bobot-bobot *output* nya ke semua *neuron* yang berhubungan dengannya. Demikian seterusnya.

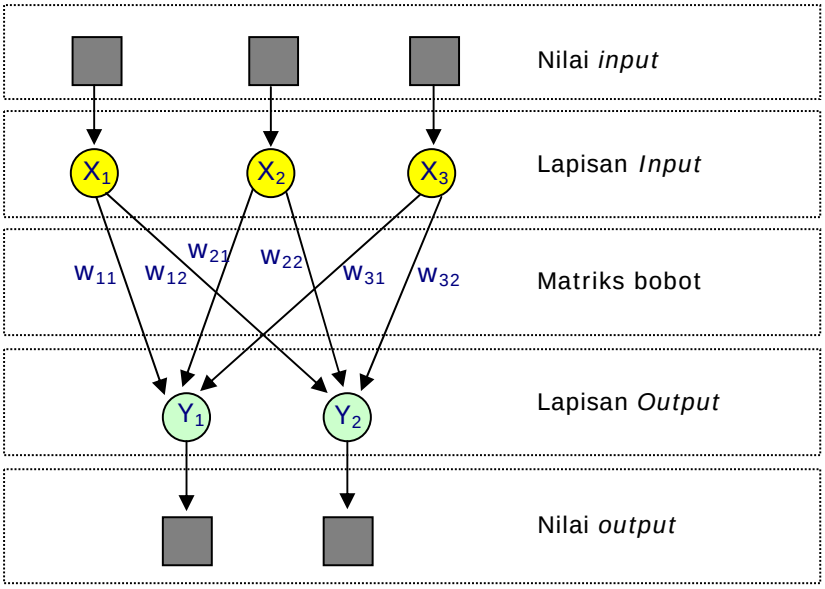
Pada Jaringan Saraf, *neuron-neuron* akan dikumpulkan dalam lapisan-lapisan (*layers*) yang disebut dengan lapisan *neuron* (*neuron layers*). Biasanya *neuron-neuron* pada satu lapisan akan dihubungkan dengan lapisan-lapisan sebelum dan sesudahnya (kecuali lapisan *input* dan *output*). Informasi yang diberikan pada Pada Jaringan Saraf akan dirambatkan lapisan ke lapisan, mulai dari lapisan *input* sampai ke lapisan *output* melalui lapisan yang lainnya yang sering dikenal dengan nama lapisan tersembunyi (*hidden layers*). Tergantung pada algoritma pembelajarannya, bisa jadi informasi tersebut dirambatkan secara mundur pada jaringan.

2.2 Arsitektur Jaringan Saraf Tiruan

Ada beberapa arsitektur jaringan saraf tiruan (Kusumadewi, 2003), antara lain:

2.2.1 Jaringan dengan lapisan tunggal (*single layer net*)

Jaringan dengan lapisan tunggal hanya memiliki satu lapisan dengan bobot-bobot terhubung. Jaringan ini hanya menerima *input* kemudian secara langsung akan mengolahnya menjadi *output* tanpa harus melalui lapisan tersembunyi, seperti yang terlihat pada Gambar 2.

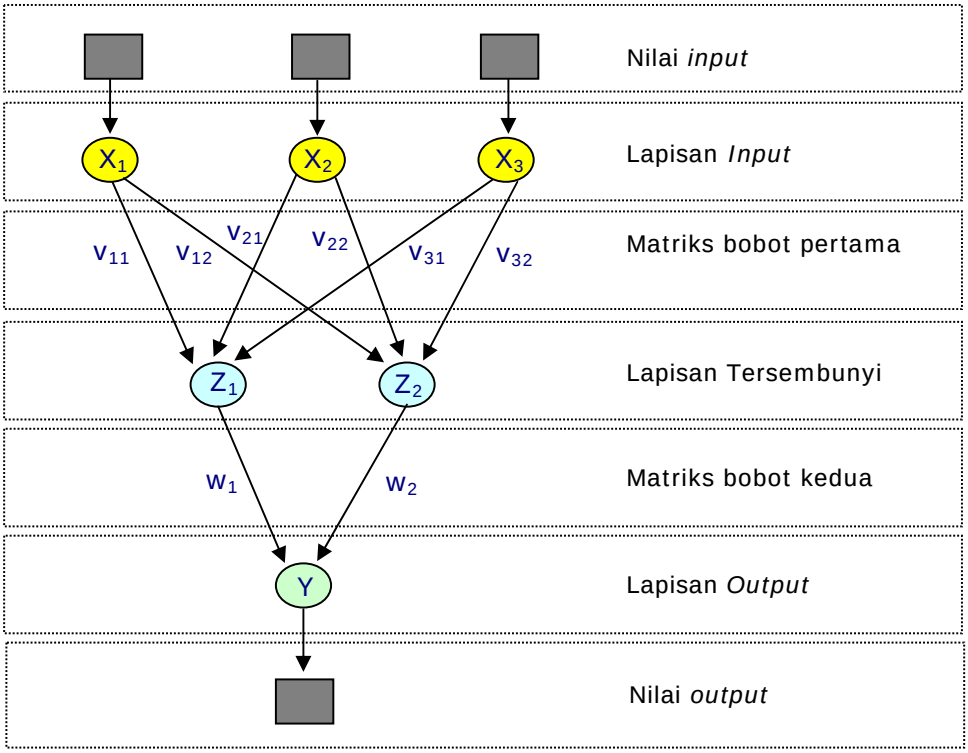


Gambar 2. Jaringan saraf dengan lapisan tunggal

Pada Gambar 2 tersebut, lapisan *input* memiliki 3 neuron, yaitu X_1, X_2 dan X_3. Sedangkan pada lapisan *output* memiliki 2 neuron yaitu Y_1 dan Y_2. *Neuron-neuron* pada kedua lapisan saling berhubungan. Seberapa besar hubungan antara 2 *neuron* ditentukan oleh bobot yang bersesuaian. Semua unit *input* akan dihubungkan dengan setiap unit *output*.

2.2.2 Jaringan dengan banyak lapisan (*multilayer net*)

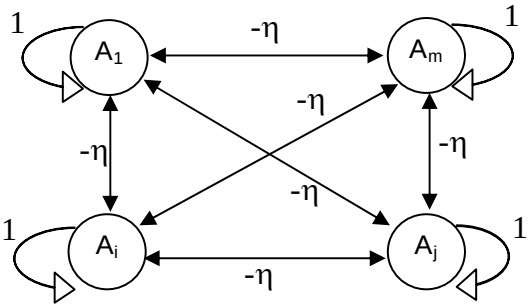
Jaringan dengan banyak lapisan memiliki 1 atau lebih lapisan yang terletak diantara lapisan *input* dan lapisan *output* (memiliki 1 atau lebih lapisan tersembunyi). Umumnya, ada lapisan bobot-bobot yang terletak antara 2 lapisan yang bersebelahan. Setiap nilai yang di*input*kan akan dikalikan dengan bobot yang terhubung ke tiap *neuron* pada lapisan tersembunyi, lalu dijumlah. Hasil penjumlahannya di*input*kan pada fungsi aktivasi yang berlaku pada *neuron* lapisan tersembunyi tersebut untuk mendapatkan hasilnya. Kemudian, nilai hasil dari tiap *neuron* lapisan tersembunyi dikalikan dengan bobot yang terhubung ke masing-masing *neuron* pada sisi *output*. Hasil penjumlahannya dimasukkan pada fungsi aktivasi yang berlaku untuk mendapatkan nilai keluarannya. Jaringan dengan banyak lapisan ini dapat menyelesaikan permasalahan yang lebih sulit daripada lapisan dengan lapisan tunggal, tentu saja dengan pembelajaran yang lebih rumit.



Gambar 3. Jaringan saraf dengan banyak lapisan

2.2.3 Jaringan dengan lapisan kompetitif (*competitive layer net*)

Merupakan jenis jaringan saraf yang memiliki bobot yang telah ditetapkan dan tidak memiliki proses pelatihan. Digunakan untuk mengetahui neuron pemenang dari sejumlah neuron yang ada. Nilai bobot untuk diri sendiri tiap neuron adalah 1, sedangkan untuk neuron lain adalah bobot random negatif. Gambar 4 menunjukkan salah satu contoh arsitektur jaringan dengan lapisan kompetitif yang memiliki bobot - η.



Gambar 4. Jaringan saraf dengan lapisan kompetitif

2.3 Fungsi aktivasi

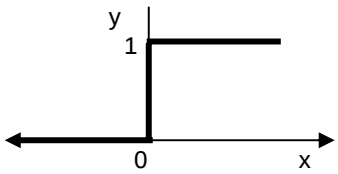
Ada beberapa fungsi aktivasi yang sering digunakan dalam jaringan saraf tiruan, antara lain:

2.3.1 Fungsi Undak Biner (*Hard Limit*)

Jaringan dengan lapisan tunggal sering menggunakan fungsi undak (*step function*) untuk mengkonversikan *input* dari suatu variabel yang bernilai kontinu ke suatu *output* biner (0 atau 1). (Gambar 5).

Fungsi undak biner (*hard limit*) dirumuskan sebagai berikut:

$$y = \begin{cases} 0, & jika\ x \le 0 \\ 1, & jika\ x > 0 \end{cases} \quad (1)$$

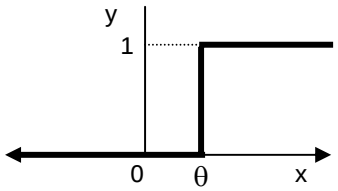


Gambar 5. Fungsi aktivasi: Undak Biner (*hard limit*)

2.3.2 Fungsi Undak Biner (*Threshold*)

Fungsi undak biner dengan menggunakan nilai ambang sering juga disebut dengan nama fungsi nilai ambang (*threshold*) atau fungsi Heaviside. Fungsi undak biner (dengan nilai ambang θ) dirumuskan sebagai berikut:

$$y = \begin{cases} 0, & jika\ x < \theta \\ 1, & jika\ x \ge \theta \end{cases} \quad (2)$$

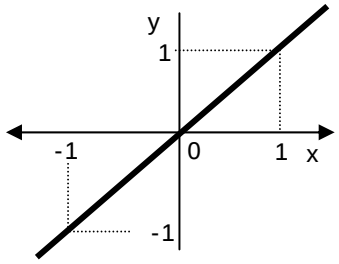


Gambar 6. Fungsi aktivasi: Undak Biner (*threshold*)

2.3.3 Fungsi Linear (identitas)

Fungsi linear memiliki nilai *output* yang sama dengan nilai *input*nya (Gambar 7). Fungsi linear dirumuskan sebagai:

$$y = x \tag{3}$$



Gambar 7. Fungsi aktivasi: Linear (identitas)

2.4 Proses Pembelajaran

Proses pembelajaran terhadap perubahan bobot dalam Jaringan Saraf Tiruan ada 2, yaitu:

2.4.1 Pembelajaran terawasi (*supervised learning*)

Metode pembelajaran pada jaringan saraf disebut terawasi jika *output* yang diharapkan telah diketahui sebelumnya. Pada proses pembelajaran, satu pola *input* akan diberikan ke satu neuron pada lapisan *input*. Pola ini akan dirambatkan di sepanjang jaringan saraf hingga sampai ke neuron pada lapisan *output*. Lapisan *output* ini akan membangkitkan pola *output* yang nantinya akan dicocokkan dengan pola *output* targetnya. Apabila terjadi perbedaan antara pola *output* hasil pembelajaran dengan pola target, maka disini akan muncul error. Apabila nilai error ini masih cukup besar, mengindikasikan bahwa masih perlu dilakukan lebih banyak pembelajaran lagi.

2.4.2 Pembelajaran tak terawasi (*unsupervised learning*)

Pada metode pembelajaran yang tak terawasi ini tidak memerlukan target *output*. Pada metode ini, tidak dapat ditentukan hasil yang seperti apakah yang diharapkan selama proses pembelajaran. Selama proses pembelajaran, nilai bobot disusun dalam suatu range tertentu tergantung pada nilai *input* yang diberikan. Tujuan pembelajaran ini adalah mengelompokkan unit-unit yang hampir sama dalam suatu area tertentu.

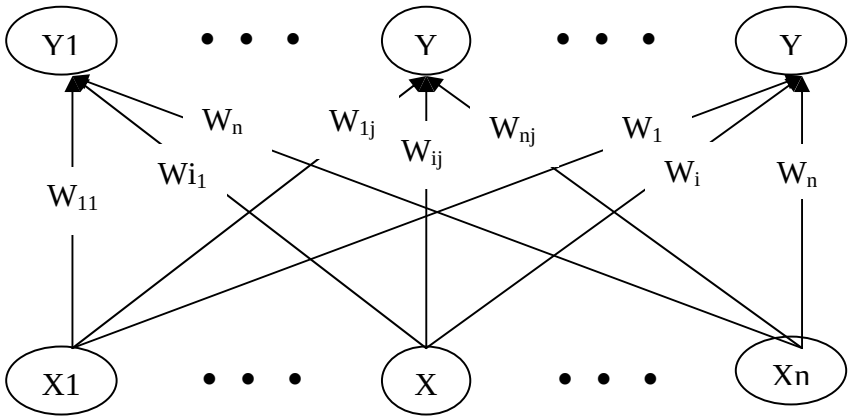
2.5 Jaringan Saraf Kohonen Self Organizing

Jaringan Saraf Kohonen *Self Organizing* termasuk dalam pembelajaran tak terawasi (*unsupervised learning*). Pertama kali diperkenalkan oleh Prof. Teuvo Kohonen pada tahun 1982. Pada jaringan ini, suatu lapisan yang berisi neuron-neuron akan menyusun dirinya sendiri berdasarkan *input* nilai tertentu dalam suatu kelompok yang dikenal dengan istilah cluster. Selama proses penyusunan diri, cluster yang memiliki vektor bobot paling cocok dengan pola *input* (memiliki

jarak yang paling dekat) akan terpilih sebagai pemenang. Neuron yang menjadi pemenang beserta neuron-neuron tetangganya akan memperbaiki bobot-bobotnya.

Tujuan pembelajaran dari algoritma ini adalah untuk mentransformasikan suatu pola sinyal masukan yang mempunyai dimensi yang berubah-ubah ke suatu peta yang berdimensi satu atau dua. Sifat pemetaan yang dimiliki Jaringan Saraf Kohonen *Self Organizing* meniru pada otak manusia yang tidak terdapat pada Jaringan Saraf Tiruan lainnya.

Terdapat *m* unit kelompok yang tersusun dalam arsitektur satu atau dua dimensi dan sinyal-sinyal masukan sejumlah *n*. vektor bobot untuk suatu unit kelompok disediakan satu *eksemplar* dari pola-pola masukan yang tergabung dengan kelompok tersebut. Selama proses pengorganisasian sendiri, unit kelompok yang memiliki vektor bobot paling cocok dengan pola masukan (ditandai dengan jarak *Euclidean* paling minimum) dipilih sebagai pemenang. Unit pemenang dan unit tetangganya diperbaharui bobotnya. Untuk susunan unit kelompok linier, tetangga dengan radius R di sekitar unit kelompok jaringan terdiri atas semua unit jaringan yang memenuhi maksimal fungsi Aj : (1, J-R < = j,= min (J+R, m) (Fausset, 1994). Arsitektur Jaringan Saraf Kohonen ditunjukkan pada Gambar 8.



Gambar 8. Arsitektur Jaringan Kohonen *Self Organizing*

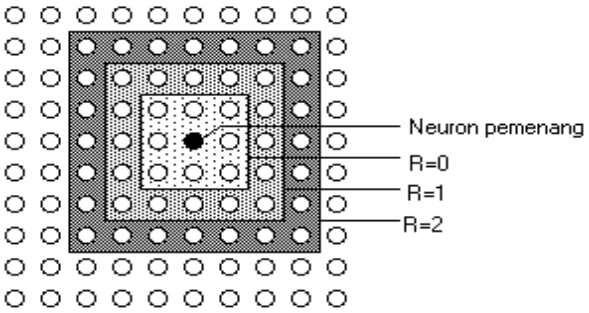
Tetangga dari unit I yang dinotasikan sebagai (#) dengan R=2, 1 dan 0 dalam topologi satu dimensi dengan 10 unit kelompok ditunjukkan pada Gambar 9 (Fausset, 1994). Pada ilustrasi unit pemenang ditandai dengan "#" dan unit-unit lainnya dengan simbol "*".

* * * { * (* { # } *) * } * * *

{ } R=2
( ) R=1
[ ] R=0

Gambar 9. Susunan linear unit-unit kelompok

Dalam topologi dua dimensi, unit-unit tetangga dengan R=2, 1 dan 0 ditunjukkan dalam bentuk lingkup segi empat pada Gambar 10. (Fausset, 1994)



Gambar 10. Tetangga-tetangga untuk lingkup segi empat

Proses pemetaan dimulai dengan menginisialisasikan bobot-bobot Wij (pemilihan bobot dapat diambil secara acak), pengesetan topologi paramater tetangga dan parameter angka pembelajaran untuk semua vektor masukan dipilih neuron yang mempunyai jarak Euclidean minimum dengan persamaan:

$$d(j) = \sum (W_{ij} - X_i)^2 \quad (4)$$

Kemudian untuk neuron pemenang beserta tetangga-tetangganya mengalami perubahan sesuai dengan persamaan berikut:

$$dW_j / dt = \alpha [X - W_{ij}] \quad (5)$$

neuron-neuron yang berada di luar lingkup tetangga tidak akan mengalami perubahan. Besar bobot baru selanjutnya dapat ditulis dengan persamaan:

$$W_{j(n+1)} = \begin{cases} W_{j(n)} + \alpha [X_{(n)} - W_{j(n)}] \\ W_{j(n)} \end{cases} \quad (6)$$

Angka pembelajaran α berkurang dengan perlahan, radius tetangga tidak akan mengalami perubahan secara perlahan. Proses dilakukan sampai tidak ada perubahan yang berarti dalam pemetaan.

2.6 Optimasi *Travelling Salesman Problem* dengan Jaringan Saraf Kohonen *Self Organizing*

Travelling Salesman Problem (TSP) pertama kali diperkenalkan oleh Rand pada tahun 1948. Pendekatan dengan menggunakan Jaringan Saraf Kohonen *Self Organizing* memberikan solusi atau penyelesaian dalam perhitungan waktu yang lebih singkat dibandingkan dengan sejumlah algoritma lain yang diterapkan pada komputer dalam bentuk program.

TSP merupakan persoalan yang mempunyai konsep sederhana dan mudah dipahami. Optimasi pada suatu penyelesaian fungsi berarti penentuan jarak lokasi minimum atau maksimum dari fungsi tersebut. Pada TSP, optimasi diinginkan agar ditemukan rute perjalanan terpendek untuk melewati sejumlah kota dengan jalur tertentu sehingga setiap kota hanya terlewati satu kali dan perjalanan diakhiri

dengan kembali ke kota semula. Terdapat dua macam bentuk persoalan TSP ditinjau dari jumlah pelaku perjalanan, bentuk pertama dengan satu orang yang berkeliling dan dengan lebih dari satu (banyak) orang yang berkeliling sebagai bentuk kedua. Dalam penelitian ini hanya akan dibahas bentuk pertama.

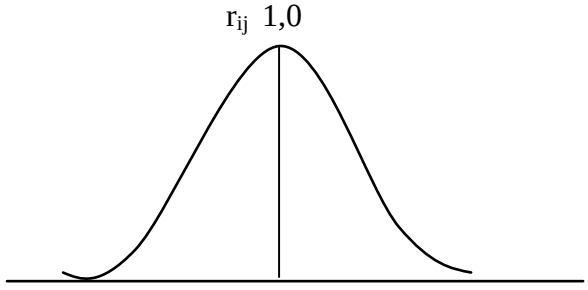
Ide pokok dari Jaringan Saraf Kohonen adalah pengorganisasian sendiri oleh jaringan. Untuk ini harus ditampilkan pola secara kontinyu dan acak sampai suatu keadaan stabil tercapai. Pada suatu sudut pandang *Neurobiological* terdapat fakta adanya hubungan *neuron-neuron*, maka untuk aplikasi permasalahan *Travelling Salesman Problem* jaringan terdiri dari dua kelompok *neuron*. Pada kelompok pertam, masing-masing *neuron* dengan neuron yang ada pada kelompok tersebut (termasuk dengan dirinya sendiri) mempunyai hubungan dengan nilai bobot tergantung pada jarak antar *neuron.*

Bobot r (i,j) antar *neuron* i dan j diberikan dengan:

$r_{ij} = e^{[(-nD(i,j)\text{^}2) / 2(\square\text{^}2)]}$ (7)

dimana D(i,j) adalah jarak euclidean antara *neuron* i dan j.

Untuk memasukan data pada jaringan dibutuhkan lapisan pada *neuron* lain (kelompok *neuron* yang kedua). Lapisan ini tidak memiliki aturan topologi yang sama dengan jaringan sebelumnya. Masing-masing *neuron* dari kelompok ini dihubungkan dengan masing-masing *neuron* dari kelompok pertama. Bobot dari hubungan dua kelompok akan dinotasikan dengan W. Bobot antar *neuron* inilah yang menentukan topologi tetangga. Topologi tetangga maksimum pada saat D_{ij} = 0, dengan kata lain pencapaian nilai maksimum pada *neuron* pemenang j dimana jarak D_{ij} = 0. Amplitudo dari topologi tetangga r (i,j) menurun dengan adanya pertambahan jarak $D_{ij,}$, mendekati nol untuk $A_{ij} \rightarrow \infty$. Illustrasi fungsi tetangga *Gaussian* ini dapat dilihat pada Gambar 11.



Gambar 11. Ilustrasi fungsi tetangga *Gaussian*

2.7 Algoritma *Travelling Salesman Problem*

Untuk selengkapnya, algoritma TSP dapat dijabarkan (Kusumadewi, 2003), sebagai berikut:

1. Tetapkan parameter-parameter:
   a. Maksimum epoh (MaxEpoh).
   b. *Learning rate* untuk perubahan bobot antar neuron (θ)
   c. *Learning rate* untuk perubahan bobot antara koordinat kota dengan neuron (φ).

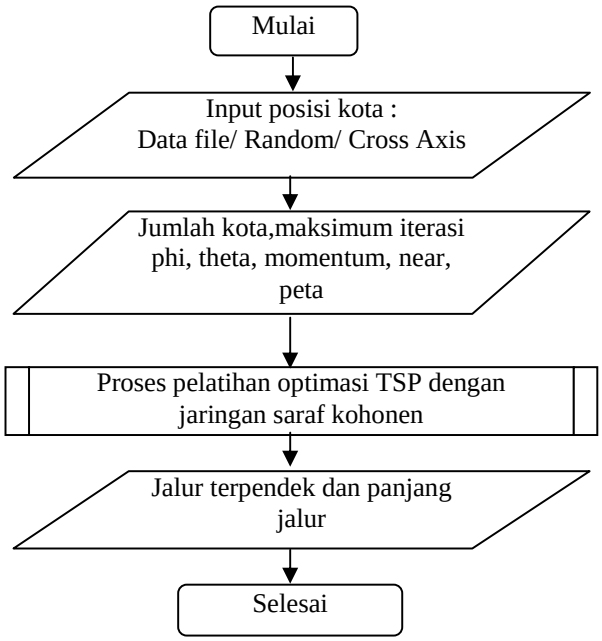
   Nilai θ dan φ bisa dibuat sama.

d. Pengurangan *learning rate* (momentum).
e. Faktor pengali untuk menuju koordinat kota (near).

2. Masukkan koordinat kota (X_i,Y_i), dengan i=1,2,…,N.
3. Cari jarak antar setiap kota ke-i dengan kota ke-j, D_{ij}, dengan i,j=1,2,…,N.
4. Tetapkan:
   a. Jumlah neuron (Q), dengan $Q \geq N$.
   b. Tetapkan koordinat awal setiap neuron (nX_i, nY_i), sedemikian hingga neuron-neuron membentuk lingkaran.
      Misal: $nX_i = 0.5 \cos(\alpha_i)$ (8)
      $nY_i = 0.5 \sin(\alpha_i)$ (9)
      dengan i=1,2,…,Q; $\alpha_1 = 0$, dan $\alpha_i = \alpha_{i-1} + 2\pi/Q$, untuk i>1.
5. Tetapkan bobot awal antara koordinat kota (x,y) dengan setiap neuron, sebut sebagai wX_i dan wY_i secara acak, antara 0 sampai 1; dengan i=1,2,…,Q.
6. Tetapkan bobot awal antar neuron, sebut sebagai r_{ij}, dengan i,j = 1,2,…,Q, dengan cara:
   a. Cari jarak setiap antar neuron, nD_{ij}, dengan i,j = 1,2,…,Q.
   b. $$r_{ij} = e^{\left[\frac{-nD_{ij}^2}{2\theta^2}\right]}$$ (10)
7. Set epoh = 0.
8. Kerjakan selama epoh < MaxEpoh:
   a. epoh = epoh+1
   b. Pilih sembarang kota secara random, misal kota terpilih adalah kota ke-idx.
   c. Set koordinat lokasi yang akan didekati (tX,tY):
      i. Bangkitkan bilangan satu random r antara 0 sampai 1.
      ii. $tX = X_{idx} + r*Near - Near/2$ ……(2.10)
      iii. $tY = Y_{idx} + r*Near - Near/2$ ……...(2.11)
   d. Cari jarak minimum antara (tX,tY) dengan bobot antara koordinat kota dan neuron (wX_i,wY_i), i=1,2,…,Q. Misalkan jarak minimumnya jatuh pada jarak antara (tX,tY) dengan bobot ke-j (wX_j,wY_j).
   e. Perbaiki bobot antara koordinat kota dan setiap neuron (wX_i,wY_i), i=1,2,…,Q:
      i. $wX_i = wX_i + \varphi * r_{ij} * (tX - wX_i)$ (11)
      ii. $wY_i = wY_i + \varphi * r_{ij} * (tY - wY_i)$ (12)
      dengan j adalah indeks terpilih pada (d).
   f. Perbaiki nilai θ dan φ:
      i. $\theta = \theta * momentum$ (13)
      ii. $\varphi = \varphi * momentum$ (14)
   g. Perbaiki bobot antar neuron (r_{ij}), dengan i=1,2,…,Q; dan j adalah indeks terpilih pada (d):
      $$r_{ij} = e^{\left[\frac{-nD_{ij}^2}{2\theta^2}\right]}$$ (15)

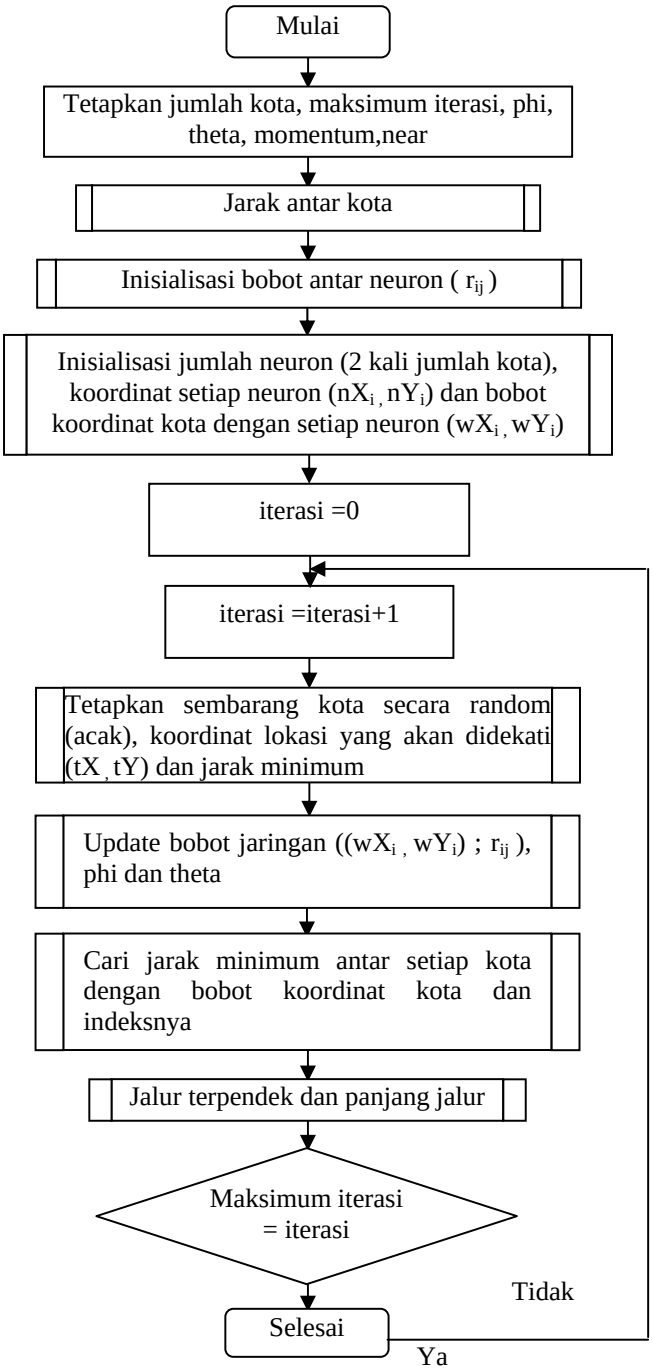
9. Cari jarak minimum antara setiap kota ke-i dengan setiap bobot koordinat kota ke neuron ke-j, sebut sebagai mJ_i, dan indeksnya sebut sebagai L_i dengan i=1,2,…,N:
   a. mJ_i=minimum $\sqrt{\left(X_i - wX_j\right)^2 + \left(Y_i - wY_j\right)^2}$; dengan j=1,2,…,Q (16)
   b. L_i=j, sedemikian hingga $\sqrt{\left(X_i - wX_j\right)^2 + \left(Y_i - wY_j\right)^2}$ =mJ_i (17)
   c. Bentuk matriks P berukuran Nx2 dengan kolom pertama adalah mJ_i dan kolom kedua berisi L_i.
10. Urutkan naik matriks P, berdasarkan kolom pertama.
11. Jalur terpendek adalah matriks P terurut kolom kedua. Cari panjang jalur terpendek.

3. DIAGRAM ALIR SISTEM



Gambar 12. Diagram Alir Perangkat Lunak

Pada penginputan melalui *data file*, dapat dilakukan pemanggilan (*load data*) atau pemasukan data (*create data*) berupa koordinat kota. Pada penginputan melalui *random*, data koordinat kota akan diplot secara acak. Sedangkan pada penginputan melalui *cross axis*, data koordinat kota di masukkan melalui pointer. Dari Diagram alir perangkat lunak diatas akan dijabarkan pemprosesan pelatihan optimasi TSP dengan jaringan saraf kohonen seperti pada Gambar 13.



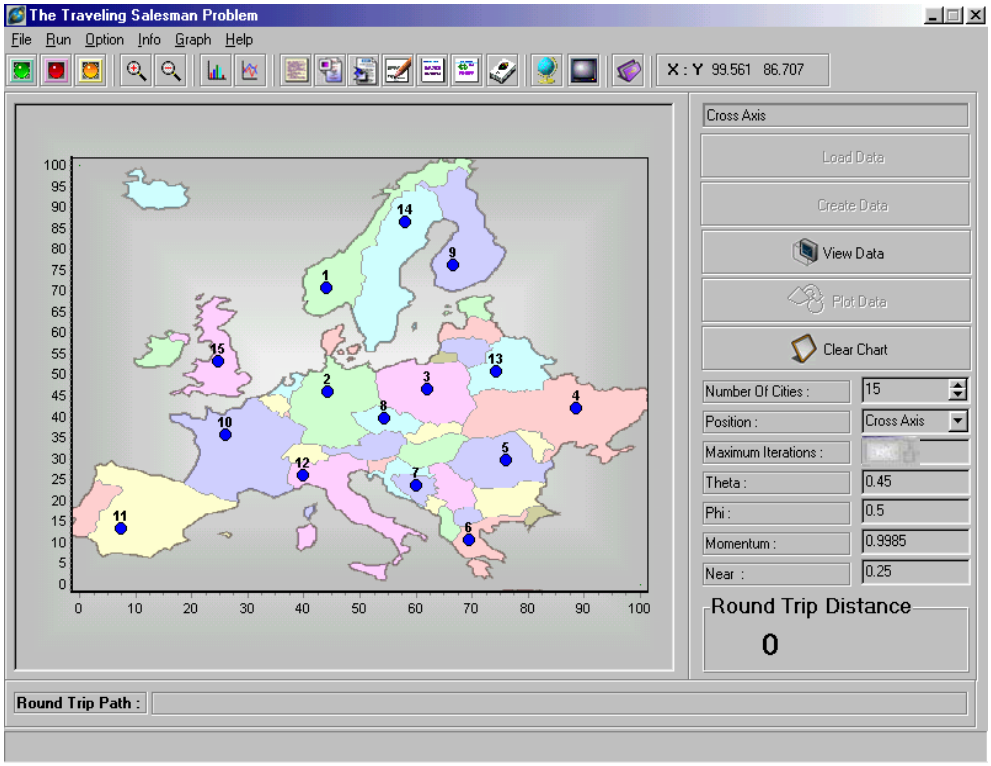
Gambar 13. Diagram Alir Perangkat Lunak Jaringan Saraf Kohonen

4. IMPLEMENTASI DAN PENGUJIAN

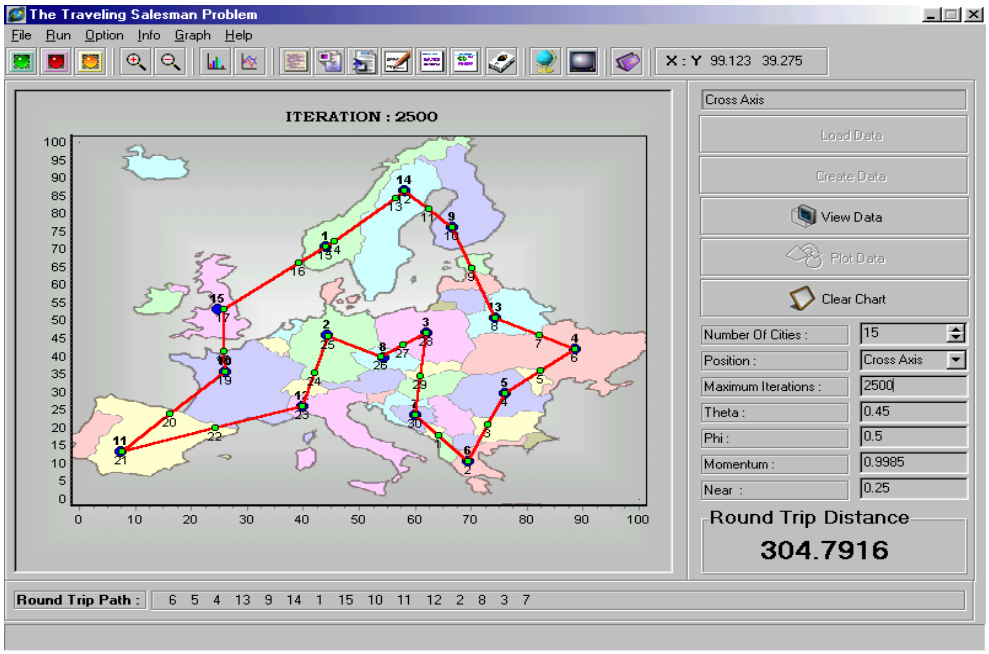
Tujuan dari tahap implementasi ini untuk memastikan perangkat lunak yang di buat dapat bekerja secara efektif dan efesien sesuai yang diinginkan. Sistem perangkat lunak yang akan dibuat adalah sebuah perangkat lunak yang berisi banyak perhitungan, peraturan logika, pengembangan visual, pola desain dan pemakaian pada lingkup kerjanya, dan fleksibilitas. Maka dibutuhkan sebuah bahasa pemrograman yang terstruktur, handal, praktis, mudah digunakan dan juga mendukung batasan sistem operasi secara umum. Borland Delphi 6.0 adalah salah satu bahasa pemrograman yang memiliki semua persyaratan yang dibutuhkan dalam proses pembuatan perangkat lunak ini.

a. Kasus-1

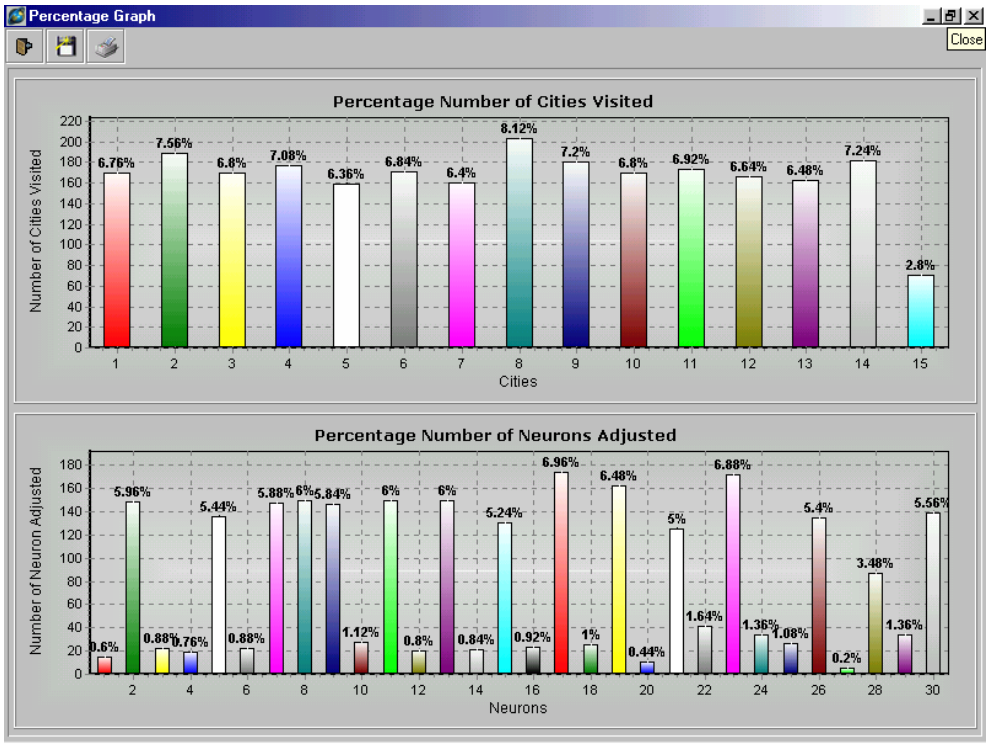
Pada uji coba kasus-1 dari beberapa kali proses pelatihan yang dilakukan, didapat panjang jalur dengan lintasan terpendek: 304.7916 satuan jarak, dengan persentase 8.12% untuk kota ke-8 yang paling banyak dikunjungi dan 2.8% untuk kota ke-15 yang paling sedikit dikunjungi. Presentase neuron yang paling sering diupdate sebesar 6.96% yaitu pada neuron ke-17 dan 0.2% untuk neuron ke-27 yang paling sedikit di-*update*. Gambar 14 sampai 16 berurutan merupakan antarmuka hasil uji perangkat lunak untuk kasus-1.



Gambar 14. Tampilan Awal Kasus-1



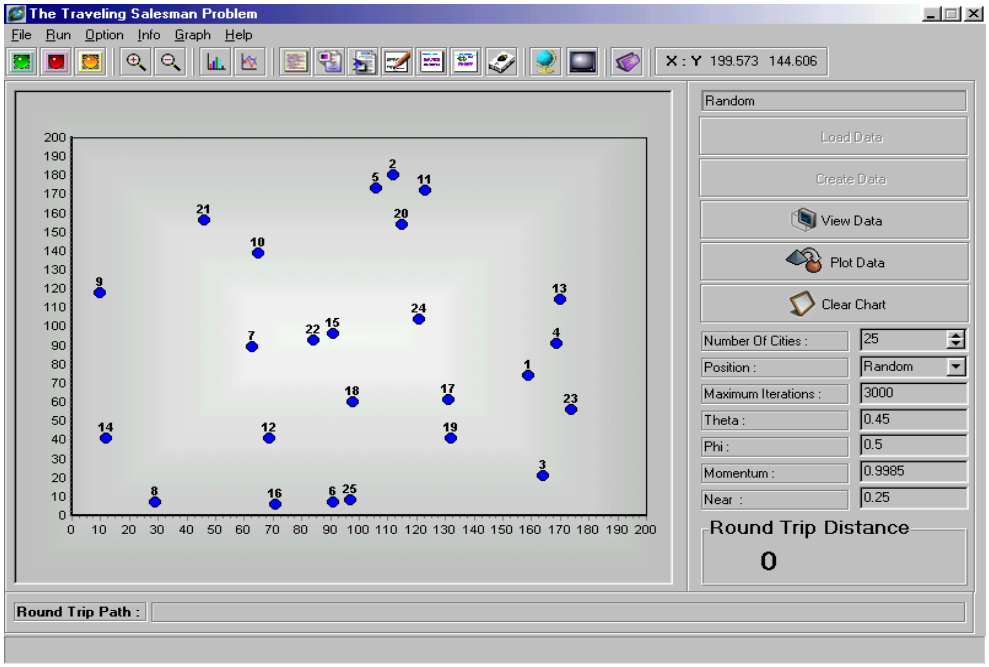
Gambar 15. Tampilan Akhir Kasus-1



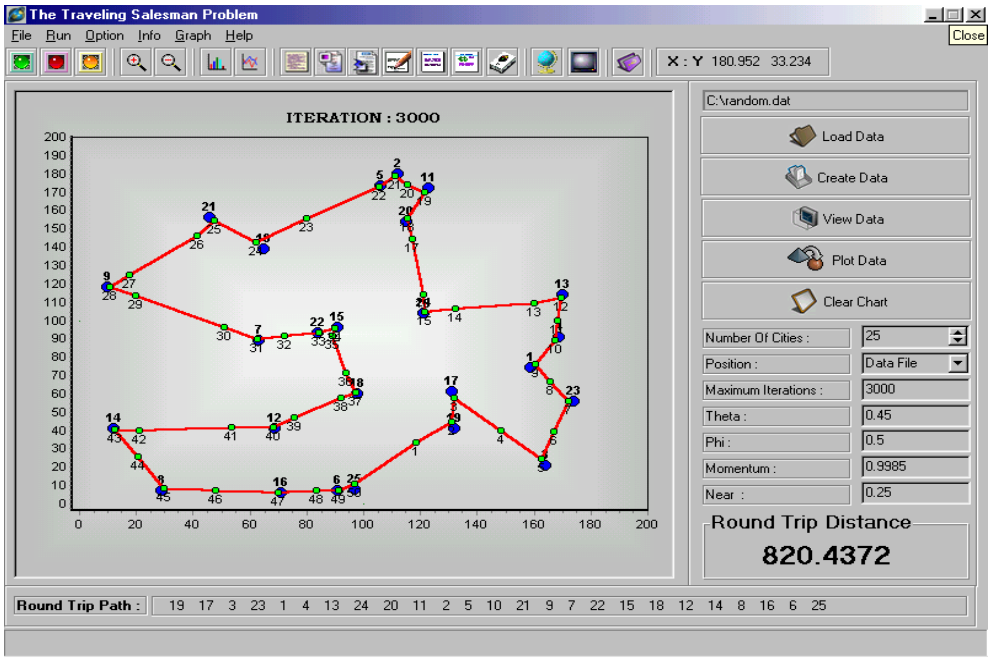
Gambar 16. Tampilan Grafik Presentase Kasus-1

b. Kasus-2

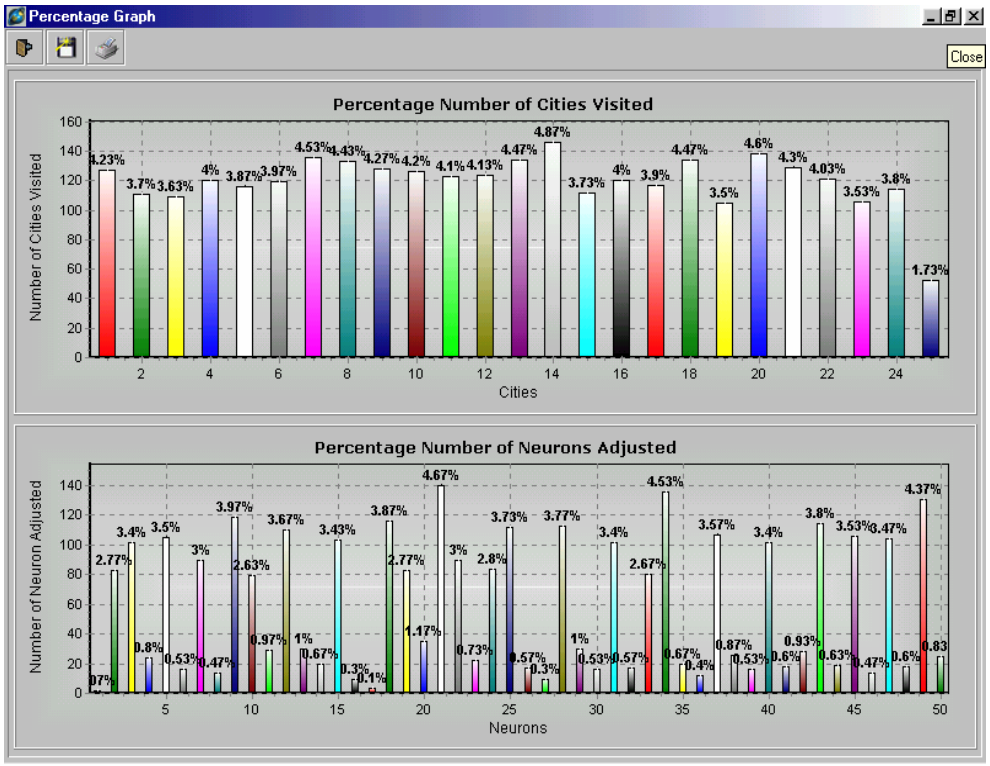
Pada uji coba kasus-2 dari beberapa kali proses pelatihan yang dilakukan, didapat panjang jalur dengan lintasan terpendek: 820.4327 satuan jarak, dengan persentase 4.87% untuk kota ke-14 yang paling banyak dikunjungi dan 1.73% untuk kota ke-25 yang paling sedikit dikunjungi. Presentase neuron yang paling sering diupdate sebesar 4.67% yaitu pada neuron ke-21 dan 0.07% untuk neuron ke-1 yang paling sedikit diupdate. Gambar 17 sampai 19 berurutan merupakan antarmuka hasil uji perangkat lunak untuk kasus-2.



Gambar 17. Tampilan Awal Kasus-2



Gambar 18. Tampilan Akhir Kasus-2



Gambar 19. Tampilan Grafik Presentase Kasus-2

5. SIMPULAN DAN SARAN

Penyelesaian masalah *travelling salesman problem* menggunakan metode jaringan saraf kohonen untuk mendapatkan rute perjalanan terpendek sangat dipengaruhi oleh parameter pelatihan seperti phi, theta, momentum dan near, dan keadaan lainnya. Sehingga jika proses pelatihan dilakukan beberapa kali dengan data koordinat kota yang sama akan memungkinkan mendapatkan jalur dan panjang jalur perjalanan yang berbeda. Hal ini mengasumsikan bahwa untuk mendapatkan hasil yang baik perlu dilakukan beberapa kali pelatihan sebagai perbandingan untuk mendapatkan rute perjalanan terpendek.

Pada proses pelatihan dengan jumlah kota yang besar akan membutuhkam memori yang cukup besar dan pelatihan akan berjalan lambat, hal ini memungkinkan pengembangan terhadap perangkat lunak yang dibuat sehingga dapat efesien dalam penggunaan memori, dan pengembangan pada visualisasinya supaya lebih baik.

PUSTAKA

Alam, M. A. J. (1999). *Belajar Sendiri Borland Delphi 5.0*. Jakarta: Elex Media Komputindo.

Fausset, L. (1994). *Fundamental of Neural Network: Architecture, Algorithm, and Application*. New Jersey: Prentice-Hall.

Kung, S. Y. (1993). *Digital Neural Network*. New Jersey: Prentice-Hall.

Kusumadewi, S. (2004). *Membangun Jaringan Syaraf Tiruan (menggunakan MATLAB & Excel Link)*. Yogyakarta: Graha Ilmu.

Setiawan, K. (2003). *Paradigma Sistem Cerdas*. Surabaya: Bayumedia Publishing.