Jurnal Computech & Bisnis, Vol. 5, No. 2, Desember 2011, 95-102
ISSN 1978-9629

PEMANFAATAN METODE SELF ORGANIZING MAP PADA OPTIMASI MASALAH TRAVELLING SALESMAN PROBLEM

Yulistian Suryono[1], Irawan Afrianto[2]
Teknik Informatika, Fakultas Teknik dan Ilmu Komputer
Universitas Komputer Indonesia
[1]e-mail : ian_z84@yahoo.com , [2]irawan_afrianto@yahoo.com

Abstract

Travelling Salesman Problem (TSP) in development usually called classic graph problem. Salesman must a plan a course route to amount of city and visit every city once with entire distance shortly. There are several method that can give optimal solution to this problem. One of method is using computer technology that is Neural Network (NN). Self Organizing Map (SOM) Neural Network have a ability to self organizing input in to classification zone. The target of designed TSP with SOM method is to find a shortest route from the cities that must be visited. The result of experiment is to prove that SOM method could solved the TSP problem.

Keywords: *Short Route Finding, Neural Network, Self Organizing Map (SOM), Input, Cluster*

Abstrak

Travelling Salesman Problem (TSP) dalam perkembangannya sering disebut sebagai masalah graf klasik. Dimana seorang *salesman* harus merencanakan rute perjalanan ke sejumlah kota dan mengunjungi setiap kota sekali dengan jarak keseluruhan perjalanan yang sependek mungkin. Banyak metode yang diyakini dapat memberikan penyelesaian optimal terhadap permasalahan ini. Salah satunya adalah dengan memanfaatkan perkembangan teknologi komputer yaitu *Neural Network (NN). Self-Organizing Maps (SOM) Neural Network* memiliki kemampuan untuk mengorganisasi input sendiri ke dalam zona klasifikasi. Perancangan *TSP* dengan metode *SOM* ini bertujuan untuk menemukan rute terpendek dari sejumlah kota yang harus dikunjungi. Hasil pengujian terhadap rancangan membuktikan bahwa metode *SOM* ini dapat menyelesaikan permasalahan *TSP.*

Kata Kunci: Pencarian Rute Terpendek, *Neural Network, Self-Organizing Maps(SOM), Input, Cluster*

1. Pendahuluan

Teknologi yang semakin canggih dari hari ke hari telah mengantarkan manusia kepada suatu kehidupan yang lebih baik. Sebagai imbasnya, banyak orang telah beralih dari pekerjaan secara manual ke sistem yang terkomputerisasi. Meskipun demikian, masih ada beberapa masalah yang belum dapat diselesaikan secara optimal. Begitu pula halnya dengan *Travelling Salesman Problem*, dimana seorang *salesman* harus mengunjungi sejumlah kota dan merencanakan rute perjalanannya sehingga setiap kota hanya dikunjungi sekali dan seluruh perjalanannya harus ditempuh dengan jarak sependek mungkin.

Travelling Salesman Problem sering dianggap sebagai masalah graf klasik. Walaupun demikian, banyak penelitian dan metode baru yang diyakini dapat memberikan penyelesaian yang optimal terhadap masalah ini. Salah satunya adalah dengan menggunakan *Neural Network*. *Neural Network* dirancang untuk memodelkan performansi otak manusia dalam melakukan fungsinya. Hal yang penting dari *Neural Network* adalah kemampuannya untuk belajar sehingga dapat melakukan kegiatan komputasi yang luar biasa. Karena memiliki karakteristik performansi yang mirip dengan jaringan saraf biologis, maka *Neural Network* menggunakan *neuron* sebagai unit pemroses informasinya.

Prosedur yang digunakan untuk melakukan proses pembelajaran atau pelatihan disebut algoritma pembelajaran, yaitu suatu fungsi untuk memodifikasi bobot keterhubungan antar *neuron* di dalam jaringan. Ada banyak algoritma pembelajaran yang dapat digunakan dalam proses komputasi, diantaranya dikenal sebagai *Self-Organizing Map* (SOM).

2. Landasan Teori

2.1 Travelling Salesman Problem

Permasalahan ini dimodelkan sebagai graf. Graf merupakan kumpulan dari *node* yang disebut *verteks* dan garis yang menghubungkan pasangan *verteks* disebut *edge*. **Gambar 1** akan memodelkan permasalahan TSP dalam konsep teori graf. Gambar tersebut menunjukkan sistem jalan raya di Wyoming dengan seseorang yang bertanggungjawab sebagai pengawas.

Gambar 1 Contoh Permasalahan TSP

Secara spesifik, pengawas jalan raya ini harus melakukan inspeksi ke seluruh bagian jalan dan memberikan laporan tentang keadaan jalan, keberadaan rambu-rambu lalu lintas, garis jalan, dan lain sebagainya. Karena pengawas tinggal di Greybull, cara yang paling ekonomis untuk melakukan pemeriksaan adalah dengan memulai dari Greybull, kemudian mengunjungi setiap jalan yang ada sekali dan akhirnya kembali ke Greybull. Apakah hal ini mungkin? Cobalah untuk menelusuri kemungkinannya.

2.2 Neural Network

Neural Network adalah sebuah sistem pemroses informasi yang memiliki karakteristik kinerja serupa dengan jaringan saraf biologis[1]. Pusat sistem adalah otak yang diwakilkan oleh jaringan saraf yang terus menerus menerima informasi, mengolahnya dan membuat keputusan yang tepat. Cara yang tepat agar otak dapat berpikir

adalah salah satu misteri ilmu pengetahuan.

Kita tahu bahwa *neuron* atau sel saraf adalah unit fungsi dasar dari semua jaringan saraf termasuk otak. Setiap *neuron* terdiri dari tubuh sel atau *soma* yang mengandung sebuah *nucleus*. Cabang yang keluar dari tubuh sel adalah sejumlah serat yang disebut *dendrit* dan sebuah serat panjang yang merentang disebut *axon*.

Setiap*neuron* berhubungan dengan *neuron* yang lain melalui saluran penghubung dan memiliki bobot keterhubungan sendiri. Bobot itu mewakilkan informasi yang sedang digunakan. Setiap *neuron* juga melakukan fungsi aktivasi yang memproses setiap *input* yang diterima agar menghasilkan *output* yang tepat.

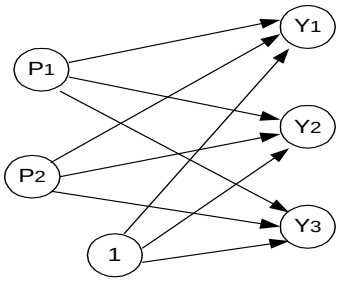
Cara kerja sistem saraf ini menjadi dasar terciptanya *Neural Network* yang memiliki karakteristik yaitu arsitektur jaringan (pola keterhubungan antar *neuron*), algoritma pelatihan atau pembelajaran (metode untuk menentukan nilai bobot hubungan), fungsi aktivasi untuk menentukan nilai keluaran berdasarkan nilai total masukan.

2.3 Self Organizing Map (SOM)

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.

2.3.1 Arsitektur SOM



Gambar 2 Arsitektur jaringan *Self Organizing Map*

Gambar 2 menunjukkan salah satu contoh arsitektur jaringan *self organizing map* dengan 2 unit pada lapisan *input* (P_1 dan P_2), serta 3 unit (*neuron*) pada lapisan *output* (Y_1, Y_2, dan Y_3). Sebagai catatan, bobot w_{ij} disini mengandung pengertian, bobot yang menghubungkan *neuron* ke-j pada lapisan *input* ke *neuron* ke-I pada lapisan *output*.

2.3.2 Algoritma Pembelajaran SOM

Algoritma pembelajaran untuk *Self-Organizing Map* adalah sebagai berikut :

0. a. Inisialisasi bobot input (w_{ij}) :

$$w_{ij} = \frac{MinP_j + MaxP_j}{2}$$

dengan w_{ij} adalah bobot antara variable input ke-j dengan *neuron* pada kelas ke-i (j = 1, 2, . . ., m ; i = 1, 2, . . ., K) ; MinPi dan MaxPi masing-masing adalah nilai terkecil pada variable input ke-i, dan nilai terbesar dari variable input ke-i.

b. Inisialisasi bobot bias (b_i) :

$$\mathbf{b_i} = e^{[\,1-\ln(1/K)\,]}$$

dengan b_i adalah bobot bias ke *neuron* ke-i, dan K adalah jumlah klas.

c. Set parameter *learning rate*.

d. Set maksimum epoh (MaxEpoh).

1. Set epoh = 0 ;
2. Kerjakan bila epoh < MaxEpoh :
   a. epoh = epoh + 1

b. Pilih data secara acak, misalkan data terpilih adalah data ke-z.
c. Cari jarak antara data ke-z dengan setiap bobot input ke-i (D_i) :

$$D_i = \sqrt{\sum_{j=1}^{m} W_{ij} - P_{jz}}^2$$

d. Hitung a : $a_i = -D_i + b_i$
e. Cari ai terbesar :
(i) MaxA = max(a_i), dengan i = 1, 2, . . ., K
(ii) Idx = i, sedemikian hingga a_i = MaxA.
f. Set *output neuron* ke-i (y_i) :
$y_i = 1$, i = Idx
$y_i = 0$, i ≠ Idx
g. *Update* bobot yang menuju ke *neuron* Idx :
$W_{Idx\text{-}j} = W_{Idx\text{-}j} + \alpha (p_{zj} - W_{Idx\text{-}j})$
h. *Update* bobot bias :
$c(i) = (1 - \alpha)\, e^{(1 - \ln(b(i)))} + \alpha\, y(i)$
$b(i) = e^{(1 - \ln(b(i)))}$

3. Analisis dan Perancangan Sistem

3.1 Analisis Masalah

Travelling Salesman Problem sering dianggap sebagai masalah graf klasik. Walaupun demikian, banyak penelitian dan metode baru yang diyakini dapat memberikan penyelesaian yang optimal terhadap masalah ini.

Pencarian jalur terpendek untuk permasalahan TSP ini proses pembelajarannya menggunakan algoritma *Self-Organizing Map* (SOM). *Self-Organizing Maps (SOM)* memiliki kemampuan untuk mengorganisasi input sendiri ke dalam zona klasifikasi. Perancangan TSP dengan metode SOM ini bertujuan untuk menemukan rute terpendek dari sejumlah kota yang harus dikunjungi. Hasil pengujian terhadap rancangan membuktikan bahwa metode SOM ini dapat menyelesaikan permasalahan TSP.

3.2 Analisis JST

Untuk membangun suatu jaringan ssaraf tiruan maka diperlukan langkah-langkah sebagai berikut :

1. Menentukan masukan
Jumlah masukan ditentukan berdasarkan kebutuhan *user* yang diinputkan kedalam sistem.
2. Menentukan parameter, yaitu maksimum epoch, *minimum learning*, *stopping condition*, *learning rate*, dan *learning reduction*.
3. Menentukan fungsi bobot yang akan digunakan berdasarkan fungsi aktivasi.
Fungsi aktivasi yang digunakan adalah fungsi bipolar sigmoid yang memiliki jangkauan nilai dari -1 sampai dengan 1 dengan fungsi sebagai berikut :

$$f(x) = \frac{2}{1 + e^{-x}} - 1$$

4. Menentukan keluaran Jumlah output ditentukan berdasarkan hasil yang diinginkan.

3.3 Analisis SOM

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*.

Pembelajaran akan dimulai dengan mencari unit pemenang dengan mengambil hasil perhitungan *Euclidean distance* yang paling minimum antara vektor *input* dan *cluster*. Setelah didapat unit pemenang, maka bobot pemenang dan tetangganya akan diperbaharui. Karena bobot awal adalah koordinat *cluster*, maka yang di*update* selanjutnya adalah koordinat *cluster*. Sehingga pada saat aplikasi dijalankan terlihat posisi *cluster* yang terus berubah.

Setelah pembelajaran dilakukan, *input* sudah mengelompok ke dalam *cluster* tertentu. Dapat dikatakan bahwa *cluster* sudah memiliki *list* kota yang

siap dipetakan TSPnya. Ada *cluster* yang terdiri dari beberapa titik *input*. Saat hal ini terjadi, ada masalah pada urutan kota yang harus dikunjungi. Oleh karena itu digunakan pendekatan lain untuk menentukan urutan kota-kota tersebut dalam TSP. Beberapa *input* yang tergabung dalam satu *cluster* akan diurutkan sehingga jaraknya dapat menjadi minimal.

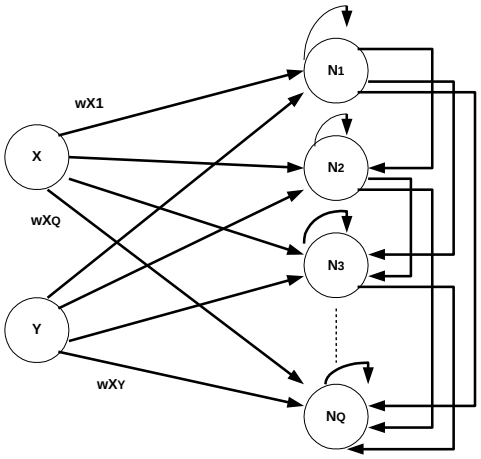
Untuk menentukan urutan ini, diperlukan kota yang menjadi awal dan akhir dalam *list* yang akan dicari urutannya. Penentuan *Start* dan *Stop* ini memerlukan bantuan dari *cluster* terdekat yang memiliki *list* kota. *Cluster* tetangga ini akan menentukan kota yang paling dekat dengan dirinya untuk dijadikan *Start* dan *Stop* pada sisi lainnya. Setelah ditemukan kota yang menjadi awal dan akhir ini, ada beberapa kombinasi urutan kota yang harus dikunjungi. Kombinasi dengan jarak yang paling pendek akan diambil untuk dimasukkan ke *path*

3.4 TSP Dalam SOM

Algoritma *Self-Organizing Map* ini digunakan untuk menyelesaikan TSP karena kemampuannya untuk mengorganisasi *input* sendiri ke dalam zona klasifikasi. Seperti telah dijelaskan sebelumnya, vektor bobot untuk sebuah unit *cluster* akan dicocokkan dengan pola *input* yang terdekat dan dipilih sebagai pemenang karena jarak *Euclidean*nya paling minimum. Setelah itu, unit pemenang dan tetangganya memperbaharui bobotnya masing-masing.

Input yang diberikan untuk pembelajaran adalah koordinat *neuron-neuron* dan setelah unit *cluster* pemenang ditemukan, bobotnya akan disesuaikan. Untuk membantu proses ini, ada beberapa nilai yang berpengaruh seperti nilai laju pembelajaran yang menurun secara lambat misalnya dengan fungsi *linear*. Selama proses pembelajaran, jarak tetangga yang digunakan juga ikut menurun.

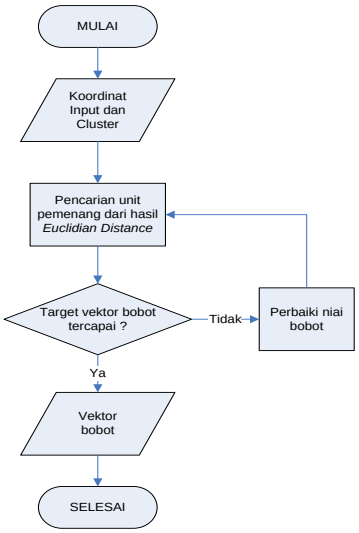
Program mencari jalur terpendek untuk permasalahan TSP ini menggunakan metode CLC yang proses pembelajarannya menggunakan algoritma *Self-Organizing Map* (SOM). Metode ini digunakan karena pendekatan yang dilakukan sistematis dan bertahap sehingga mampu mengontrol proses perancangan program. Untuk permasalahan TSP ini, arsitektur yang digunakan adalah sebagai berikut :



Gambar 3 Arsitektur *Kohonen Self Organizing Map* Untuk TSP

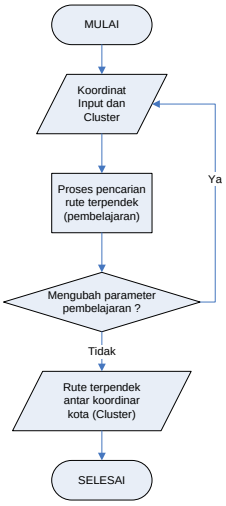
3.5 Perancangan Prosedural

Perancangan prosedural ini meliputi pembuatan Diagram Alir *(Flow Chart). Flow Chart* menggambarkan aliran proses di dalam *program* atau prosedur sistem secara logika. Diagram alir menunjukkan bagaimana proses di dalam sistem tersebut dimulai dan diakhiri, dimana masing-masing proses berada dalam urutan yang semestinya.



Gambar 4. Alur proses JST

Proses secara global ini dimulai dengan *input* koordinat *input* dan *cluster* yang telah di-*generate* hingga diperoleh rute terpendek antar koordinat.



Gambar 5 *Flow Chart* Prosedur Utama

3.6 Perancangan Antarmuka

Pada tampilan ini ada 3 menu, yaitu *File*, *View*, dan *Help*. Menu *File* terdiri atas *Open*, *Save* dan *Exit*. Hasil pencarian rute terpendek dapat disimpan dan dipanggil kembali sewaktu.-waktu dengan menggunakan submodul Save dan Exit.

Menu *View* terdiri atas *Display Setting*, *Learning Setting* dan *Learning Panel*. *User* dapat mengganti warna *link*, mengganti *boundaries layer* dan menambahkan *background* pada submodul *Display Setting*. Nilai-nilai yang berpengaruh pada pembelajaran juga dapat diubah pada *Learning Setting*. *Learning Panel* digunakan untuk mengaktifkan panel pembelajaran pada sebelah kanan *layer*.

Menu *Help* terdiri atas *About* yang berisikan data pembuat. Pada saat program dijalankan, submenu ini sudah berfungsi.

Gambar 6 Tampilan Menu Utama

4. Implementasi dan Pengujian

4.1 Implementasi

Setelah melakukan tahap perancangan sistem dan implementasi perangkat lunak, maka tahap selanjutnya yang dilakukan adalah penerapan hasil perangkat lunak tersebut. Pada bagian ini akan dilakukan penerapan hasil perancangan menjadi sebuah program aplikasi agar dapat dioperasikan sesuai dengan tujuan perancangan dan untuk mengetahui apakah program berjalan sesuai dengan fungsinya.

4.2 Pengujian

Pengujian yang dilakukan terhadap program ini menggunakan metode *Black Box Testing*. Metode ini merupakan pengujian program berdasarkan fungsi dari program. Metode ini diterapkan dengan memasukan *input* yang akan diproses oleh program dan menghasilkan *output*. Kemudian akan diuji apakah *output* program sesuai dengan fungsi dari program. Tujuan dari *Black Box Testing*

adalah untuk menemukan kesalahan fungsi pada program.

Contoh hasil pengujian terhadap kasus 50 kota dan *input* posisi kota yang sama (jarak 24887,387 dalam satuan pixel) tetapi dibedakan jumlah *cluster* dan *learning setting*nya. Jumlah *cluster*nya 50, 55, 60 dan 65.

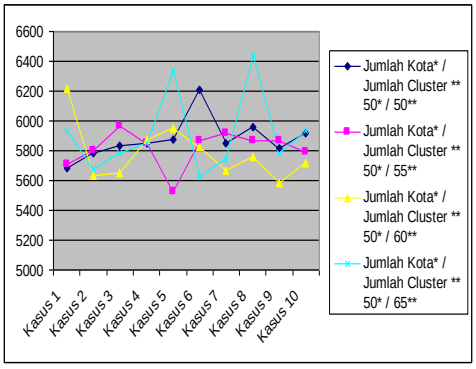
- Neighbour : 70 %
- Learning Start : 0.6
- Learning Reduction : 0.5
- Stopping Condition : 0.001

4.3 Hasil Pengujian

Berikut ini adalah hasil dari beberapa pengujian yang telah diubah-ubah parameternya seperti yang tertera di atas. Tiap pengujian dilakukan 10 percobaan (kasus) yang berbeda posisi *cluster*nya.

Tabel 1 Hasil Perhitungan Jarak dari TSP pada Pengujian ke-1

Kasus	Jumlah Kota* / Jumlah Cluster **			
	50* / 50**	50* / 55**	50* / 60**	50* / 65**
Kasus 1	5679,233	5709,403	6213,406	5931,683
Kasus 2	5785,547	5801,391	5633,309	5674,798
Kasus 3	5836,567	5966,905	5647,851	5785,78
Kasus 4	5851,429	5851,429	5878,601	5851,429
Kasus 5	5875,404	5527,266	5947,723	6340,052
Kasus 6	6211,756	5863,199	5826,135	5626,638
Kasus 7	5846,684	5920,479	5663,032	5743,344
Kasus 8	5954,685	5862,926	5754,99	6438,366
Kasus 9	5816,038	5867,855	5580,58	5781,909
Kasus 10	5916,389	5791,77	5714,624	5931,085



Gambar 7 Grafik Hasil Perhitungan Jarak TSP pada Pengujian ke-1

Hasil pengelompokan yang ideal adalah satu *cluster* untuk satu *input*. Dengan kondisi ini TSP yang akan terbentuk sudah pasti. Tetapi sering terjadi kondisi dimana satu *cluster* terdiri dari beberapa *input* dan ada beberapa *cluster* yang kosong. Dari hasil pembelajaran, letak *cluster* tersebar tidak merata, ada tempat yang banyak *input* tetapi *cluster* sedikit atau sebaliknya. Hal ini disebabkan juga karena posisi awal *cluster* yang di*random*. Saat jumlah *cluster* ditambah, kecenderungan untuk mencapai kondisi yang ideal lebih besar sehingga dengan kondisi tersebut dapat mempengaruhi *distance* yang tercipta. Dari hasil pengujian dapat dilihat bahwa penambahan jumlah *cluster* antara 10%-20% dari jumlah kota yang ada dapat meminimalkan jarak.

Perancangan ini dilengkapi dengan proses *sorting* kota-kota yang berada pada *cluster* yang sama. Penambahan proses *sorting* ini membedakan perancangan aplikasi ini dengan yang dikemukakan oleh Le Texier(1988). Kota-kota yang berada pada *cluster* yang sama perlu di*sorting* untuk menentukan urutannya dalam TSP.

Nilai-nilai yang berpengaruh dalam pembelajaran antara lain Laju Pembelajaran (*Learning Rate*). Nilai ini biasanya bernilai kecil antara 0-1. Nilai awal Laju Pembelajaran yang terlalu besar ataupun terlalu kecil tidak efektif dalam pembelajaran. *Learning rate* juga mengalami penurunan (*reduction*). Besarnya nilai *reduction* ini berpengaruh pada lama pembelajaran, semakin besar *reduction*, pembelajaran semakin lama sehingga mencapai suatu *stopping condition* yang telah ditentukan.

Nilai *Learning Rate* yang menurun semakin lama semakin kecil sehingga tidak terlalu berpengaruh terhadap pembelajaran, sedangkan hasil pembelajaran mungkin belum optimal. Oleh karena itu ditentukan suatu *Minimum Learning*. Nilai ini digunakan saat *Learning Rate* sudah mencapai di

bawah nilai *Minimum Learning* tetapi belum mencapai *Stopping Condition*.

Dalam pembelajaran ini, saat *cluster* di*update*, *neighbour* juga ikut di*update.* Hal ini dilakukan agar *neighbour* juga ikut belajar sehingga karakteristiknya semakin mirip satu dengan yang lain. Untuk nilai *Neighbour* ini, ditetapkan juga minimumnya karena tetangga yang belajar akan terus menurun. Jika tetangga yang belajar kurang dari satu, maka yang di*update* hanya *winning unit* saja. Hal ini dapat menyebabkan tetangga menjauh dan karakteristik tidak mirip.

5. Kesimpulan

Berdasarkan hasil analisa dan pembahasan yang yang telah diuraikan pada bab-bab sebelumnya, maka penulis dapat menarik kesimpulan sebagai berikut:

1. Jumlah *cluster* mempengaruhi *distance* (jarak) yang tercipta. Penambahan jumlah *cluster* antara 10% - 20% dapat meminimalkan jarak yang tercipta.
2. Nilai awal *Learning Rate* yang terlalu besar ataupun terlalu kecil tidak efektif dalam menghasilkan rute terpendek. Oleh karena itu dalam pengujian hanya digunakan nilai antara 0,5 – 0,7 untuk *Learning Rate*-nya.
3. Besarnya nilai *Learning reduction* ini berpengaruh pada lama pembelajaran, semakin besar *reduction*, pembelajaran semakin lama sehingga mencapai suatu *stopping condition* yang telah ditentukan.
4. Penyelesaian *Travelling Salesman Problem* (TSP) yang dilakukan dengan metode *Self-Organizing Map* masih ada *cross* yang terjadi. Hal ini disebabkan oleh pendekatan terhadap *sorting* kota-kota yang berada dalam *cluster* yang sama.

DAFTAR PUSTAKA

[1] Alam, M Agus. *Belajar Sendiri Borland Delphi*: Elex Media Komputindo. Jakarta, 2001.

[2] Fausette, Laurene. *Fundamentals of Neural Network*. Englewood Cliffs: Prentice Hall International, 1994.

[3] Freeman, James A. *Neural Network Algorithm, Aplication, and Programming Techniques*: Addison Westley Publishing. New York, 1992.

[4] Kusumadewi, Sri. *Artificial Intelligence*.: Graha Ilmu. Yogyakarta, 2003.

[5] Kusumadewi, Sri. *Membangun Jaringan Syaraf Tiruan*: Graha Ilmu. Yogyakarta, 2004.

[6] Schrijver's, Alexander. *History of The TSP*. http://www.tsp.gatech.edu/, 1 September 2006.

[7] Setiawan, Yudha C. *Trik & Tip Delphi*: Andi Yogyakarta, 2004.