Penyelesaian Masalah Symmetric Traveling Salesman Problem Dengan Jaringan Saraf Continuous Hopfield Net

Apul Prima S, Sri Suwarno, R. Gunawan Santosa
Fakultas Teknologi Informasi, Program Studi Teknik Informatika
Universitas Kristen Duta Wacana Yogyakarta
Email: apul.lingga@gmail.com, sswn@ukdw.ac.id, gunawan@ukdw.ac.id

Abstrak :

Makalah ini membahas Penyelesaian Masalah Symmetric Traveling Salesman Problem Dengan Jaringan Saraf Continuous Hopfield Net. Fokus permasalahan adalah seorang *salesman* harus mengunjungi semua kota sebanyak satu kali dan salesman tersebut harus mulai dari dan kembali ke kota asal. Tujuannya adalah menentukan rute perjalanan dengan jarak total ataupun biaya yang paling minimum. Salah satu bentuk permasalahan TSP adalah *symmetric* TSP yang menandakan bahwa jarak antar kota bersifat simetrik dengan setiap kota terhubung satu sama lain. Solusi dari permasalahan ini adalah dengan membangun suatu sistem aplikasi untuk menyelesaikan permasalahan *symmetric TSP*. Adapun penulis menggunakan salah satu metode jaringan saraf tiruan yaitu *Continuous Hopfield Net*

Kata Kunci : *Symmetric Traveling Salesman Problem, Continuous Hopfield Net*

1. Travelling Salesman Problem

Permasalahan optimalisasi merupakan permasalahan yang banyak dijumpai dalam kehidupan sehari-hari. Proses pengiriman barang dari suatu tempat ke tempat lain merupakan bentuk optimalisasi biaya yang dikeluarkan sehingga proses tesebut dapat menghemat pengeluaran dari suatu perusahaan. Dalam pengkajian permasalahan optimalisasi, pemodelan menggunakan komputer merupakan langkah yang tepat sehingga suatu keputusan dalam pengiriman barang dapat dilakukan secara efisien. Salah satu bentuk pemodelan yang dapat dilakukan adalah dengan menerapkan suatu permasalah optimalisasi kedalam suatu aplikasi komputer seperti permasalahan *traveling salesman problem*.

Traveling salesman problem merupakan permasalahan optimalisasi klasik yang melibat seorang *salesman* untuk menjual produknya ke beberapa kota yang telah ditentukan. Rangkaian kota yang dikunjungi akan membentuk suatu rute dengan ketentuan setiap kota hanya dapat dikunjungi tepat satu kali dan kembali ke kota awal perjalanan dimulai.

Permasalahan ini akan menjadi semakin rumit seiring bertambahnya jumlah kota yang harus dikunjungi. Kemungkinan rute yang semakin bertambah akan menyulitkan di dalam pemilihan rute dengan jarak terpendek.

Pada penelitian ini akan diterapkan metode *Continuous Hopfield Net* yang merupakan salah satu metode dalam jaringan saraf tiruan didalam menyelesaikan permasalahan *symmetric traveling salesman problem*. Metode ini bekerja dengan memanfaatkan informasi jarak antar masing-masing kota untuk selanjutnya diproses dengan parameter yang dibuat. Prinsip kerja dari *Continuous Hopfield Net* adalah dengan meminimalisasi fungsi *output* sehingga dapat diformulasikan kedalam bentuk 0 atau 1. Melalui penerapan metode *Continuous Hopfield Net* dalam permasalahan *traveling salesman problem* diharapkan dapat memberikan hasil yang lebih optimum.

2. Landasan Teori

2.1 Continuous Hopfield Net

Continuous Hopfield Net merupakan modifikasi dari diskrit Hopfield net dengan menggunakan fungsi *ouput* secara terus menerus (Laurene fausett,1994). Koneksi yang terjalin antar unit bersifat *bidirectional* sehingga bobot matriks bersifat *symmetric.* Jaringan Hopfield yang dikembangkan oleh John Hopfield dan David Tank (1982) memiliki arsitektur *single layer* dan bersifat *unsupervised learning. Single layer* merupakan jaringan yang hanya memiliki satu lapisan bobot koneksi dan terdiri dari unit-unit input yang menerima sinyal dari dunia luar dan unit output yang menghasilkan respon dari jaringan saraf tersebut.

Hopfield bersifat *unsupervised learning* (pembelajaran tak terawasi) yaitu jaringan saraf tiruan yang tidak memiliki data atau contoh pelatihan sehingga dalam prosesnya jaringan ini akan mengorganisasikan dirinya sendiri sehingga membentuk vektor-vektor input yang serupa.

Dalam pemrosesannya jaringan Hopfield bersifat *asynchronous updating,* artinya pada suatu saat hanya satu unit (neuron) yang mengupdate aktivasi.

2.2 Optimasi TSP Dengan *Continuous Hopfield Net*

Untuk menyelesaikan permasalahan optimasi seperti TSP, *Continuous Hopfield net* bekerja dengan meminimalisasi fungsi energi. Nilai energi yang terkecil akan menunjukan penyelesaian yang paling optimal. Jarak antar kota yang bersifat simetrik merupakan informasi yang menjadi input dan dianggap sebagai bobot jaringan. Berikut ini fungsi energi yang dibuat oleh John Hopfield dan David Tank (1985) yang digunakan untuk permasalahan TSP :

$$E = \frac{A}{2}\sum_{x}\sum_{i}\sum_{j\neq i} V_{x,i}V_{x,j} + \frac{B}{2}\sum_{i}\sum_{x}\sum_{y\neq x} V_{x,i}V_{y,i}$$

$$+\frac{C}{2}\left[n - \sum_{x}\sum_{i} V_{x,i}\right]^2 + \frac{D}{2}\sum_{x}\sum_{y\neq x}\sum_{i} d_{x,y}V_{x,i}\left(V_{y,i+1} + V_{y,i-1}\right)$$

Keterangan :

A,B,C,D	: parameter bernilai positif
V	: output unit (neuron)
n	: jumlah kota
i, j, y, x	: merupakan indeks posisi pada *array*
$d_{x,y}$	: jarak kota x ke kota y

Suku pertama dalam perhitungan energi merupakan penghambat baris, yang bernilai 0 jika dan hanya jika terdapat satu unit (*neuron*) yang bernilai 1 dan selebihnya bernilai 0 untuk setiap baris. Suku kedua merupakan penghambat pada kolom, yang bernilai 0 jika dan hanya jika terdapat satu unit (*neuron*) yang bernilai 1 dan selebihnya bernilai 0 untuk setiap kolom. Suku ketiga merupakan penghambat secara keseluruhan yang bernilai 0 jika dan hanya jika jumlah seluruh *neuron* yang bernilai 1 sama dengan jumlah kota(n). Suku keempat diminimalisasi sehingga jarak rute perjalanan diminimalisasi.

Pada tahap awal setiap *neuron* akan diberi nilai input inisialisasi ($X_1, X_2, ..., X_n$) dan kemudian *output* dari *neuron* tersebut akan dihitung. Pemberian nilai *input* inisialisasi dilakukan secara *random* (Zbigniew Nagórny, 2009), dengan interval sebagai berikut :

$$-0.1.\mathrm{Uo} \leq \mathrm{Inisial\ Input} \leq 0.1.\mathrm{Uo}$$

Keterangan :

Uo : konstanta bernilai 0.1

Nilai *random* untuk setiap neuron tersebut diberikan agar menghasilkan inisial ouput yang juga bernilai *random,* sehingga kondisi awal jaringan tidak menunjukkan adanya dominasi salah satu neuron yang mendekati 1 atau 0 untuk setiap baris atau kolomnya.

Proses untuk mendapatkan nilai aktivasi (*input*) yang baru dilakukan dengan menggunakan rumus perubahan aktivasi sebagai berikut :

$D\,U_{x,i}$ = Dt(Term1 + Term2 + Term3 + Term4 + Term5)

Term1 = $-U_{x,i}/1$

Term2 = $-A\sum_{j\neq i} V_{x,j}$

Term3 = $-B\sum_{y\neq x} V_{y,i}$

$$\text{Term4} = -C\left(\sum_{x}\sum_{j} V_{x,j} - n\right)$$

$$\text{Term5} = -D\sum_{y} d_{x,y}\left(V_{y,i+1} + V_{y,i-1}\right)$$

$$U_{x,i}\ (\textit{new}) = U_{x,i}\ (\textit{old}) + D\,U_{x,i}$$

Nilai konstanta A,B,C,D pada rumus perubahan bernilai sama dengan konstanta pada rumus fungsi energi. $D\,U_{x,i}$ merupakan perubahan nilai input terhadapat waktu(delta), n merupakan parameter yang bernilai positif (n>jumlah kota), Dt merupakan delta t (s*tep function*).

Perhitungan terhadap *output* dari setiap unit dilakukan dengan menggunakan rumus sebagai berikut :

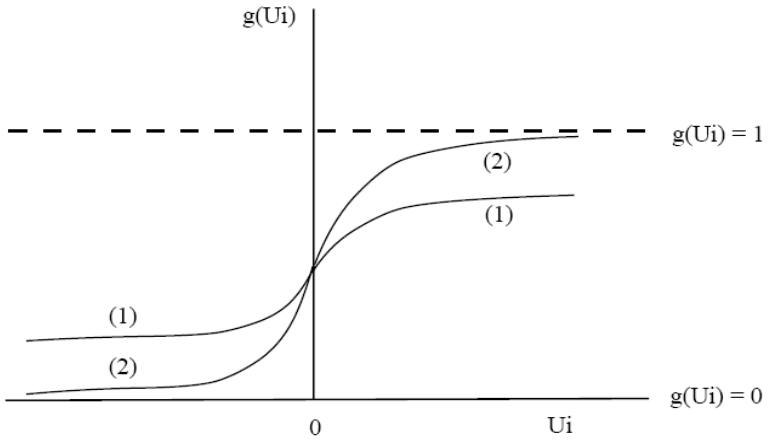
$$V_{x,i} = g(U_{x,i}) = \frac{1}{2}\left(1 + \tanh\left(\frac{U_{x,i}}{Uo}\right)\right)$$

Keterangan :

$V_{x,i}$: Ouput unit pada indeks x dan i

$U_{x,i}$: Input unit pada indeks x dan i

Hyperbolic tangen (tanh) adalah rumus matematis yang menghasilkan nilai dengan interval antara -1 sampai dengan 1. Dalam jaringan saraf tiruan fungsi *hyperbolic tangen* digunakan sebagai fungsi pengaktif (aktivasi) untuk menghasilkan *output* dari suatu *neuron*. Rumus perhitungan *output* akan mengakibatkan *output* yang dihasilkan memiliki interval 0 sampai dengan 1. Gambar 1 menunjukkan suatu skema sesuai dengan perhitungan *output*.



Gambar 1. Skema fungsi *output*

Algoritma yang digunakan untuk proses *update input neuron* dalam permasalahan *traveling salesman problem* adalah sebagai berikut:

Tahap 0. Inisialisasi aktivasi untuk semua unit(*neuron*)

Inisialisasi nilai Dt minimum (10^{-4})

Tahap 1. Jika kondisi *false,* kerjakan tahap 2 sampai tahap 6

Tahap 2. Kerjakan tahap 3 sampai tahap 6 sebanyak n^2 kali (n merupakan jumlah kota)

Tahap 3. Lakukan pemilihan unit secara *random*

Tahap 4. Proses update aktivitas unit tertentu

$$U_{x,i}(new) = U_{x,i}(old) + Dt\left(-U_{x,i}/1 - A\sum_{j\neq i} V_{x,j} - B\sum_{y\neq x} V_{y,i}\right.$$

$$\left. - C\left(\sum_{x}\sum_{j} V_{x,j} - n\right) - D\sum_{y} d_{x,y}\left(V_{y,i+1} + V_{y,i-1}\right)\right)$$

Tahap 5. Kerjakan fungsi *output*

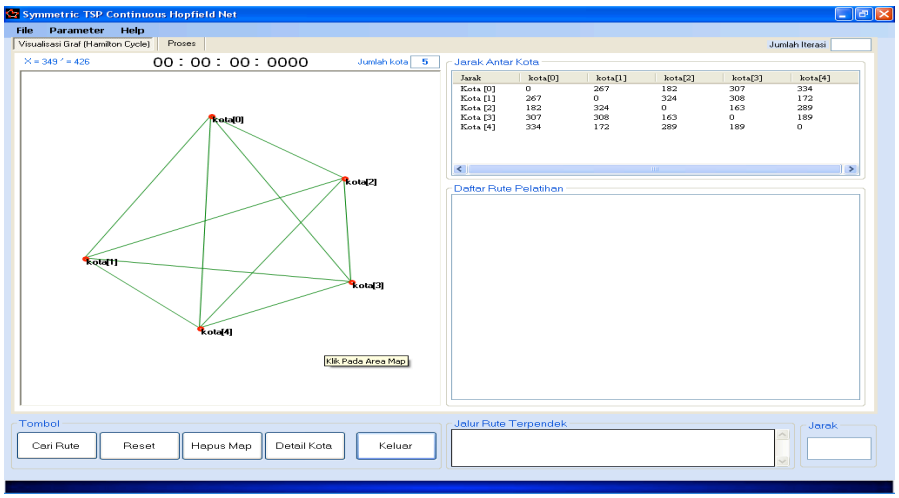
$$V_{x,i} = g(U_{x,i}) = \frac{1}{2}\left(1 + \tanh\left(\frac{U_{x,i}}{Uo}\right)\right)$$

Tahap 6. Cek kondisi perulangan (*stopping condition*)

Proses percobaan tersebut akan berhenti pada saat kondisi jaringan *konvergen* yaitu output dari setiap unit (*neuron*) membentuk matriks yang setiap baris hanya ada satu unit yang bernilai 1 dan selebihnya bernilai 0, dan setiap kolom hanya satu unit yang bernilai 1 dan selebihnya bernilai 0. Pada saat kondisi jaringan konvergen maka dilakukan perhitungan energi jaringan keseluruhan. Nilai dari energi yang terkecil akan membentuk rute dengan jarak yang paling minimum.

3. Hasil dan Pembahasan

Sistem yang dihasilkan menyajikan hasil peletakan kota dan jarak antar kota serta jarak terpendek dari rute TSP. Sistem terdiri dari tiga langkah yaitu menghitung jarak antar kota, inisialisasi unit jaringan dan penarian jarak terpendek.



Gambar 2. Papan Peletakan Kota

Secara lengkap langkah-langkah tersebut adalah sebagai berikut :

1. Menghitung jarak antar kota

   Sistem akan membuat objek kota yang baru setiap kali peletakan kota pada papan area dilakukan. Hasil peletakan kota seperti pada gambar 2. Objek tersebut akan disimpan kedalam *array* 1 dimensi yang memiliki informasi koordinat x dan y . Pencatatan koordinat dilakukan dengan mendeteksi *cursor mouse* pada lokasi papan area. Informasi koordinat masing-masing objek kota tersebut selanjutnya dipakai untuk penggambaran *visual* graf pada papan area, dan juga untuk menghitung jarak antar kota. Tabel 1 menunjukkan koordinat objek kota pada contoh kasus 5 kota diatas.

Tabel.1. Koordinat Objek Kota

Kota(indeks)	X	Y
Kota(0)	217	69
Kota(1)	72	293
Kota(2)	370	167
Kota(3)	378	330
Kota(4)	204	403

Dari informasi koordinat pada Tabel 1, selanjutnya dilakukan perhitungan jarak antar kota dengan menggunakan rumus *Phytagoras*. Hasil perhitungan jarak akan disimpan kedalam

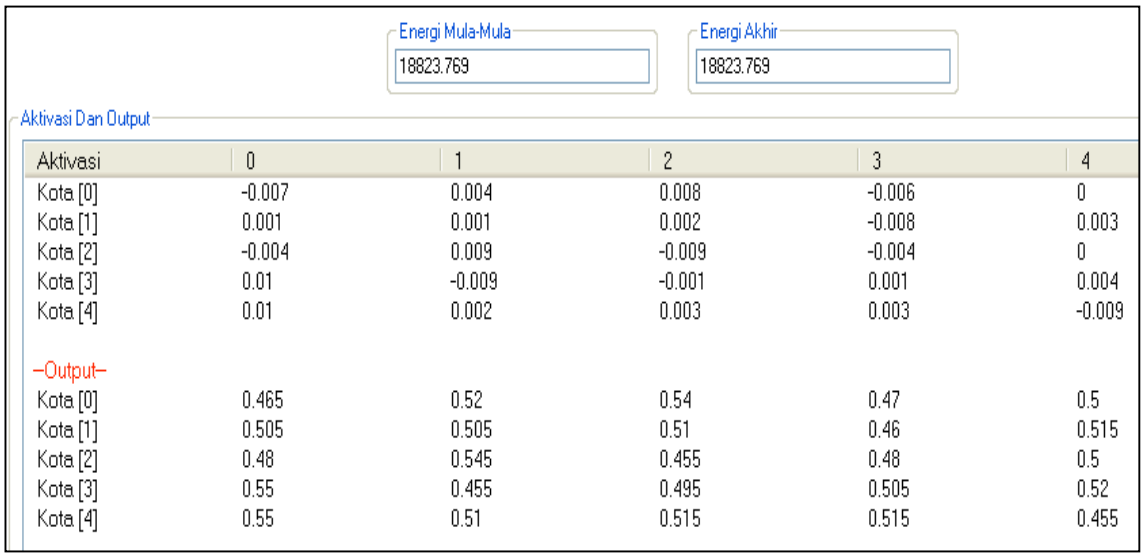
bentuk *array* 2 dimensi seperti yang terlihat pada Tabel 2.

Tabel 2. *Array* 2 Dimensi Jarak Kota

Indeks (,)	0	1	2	3	4
0	0	267	182	307	334
1	267	0	324	308	172
2	182	324	0	163	289
3	307	308	163	0	189
4	334	172	289	189	0

2. Inisialisasi unit jaringan

Pada metode Continuous Hopfield net, jumlah unit jaringan bergantung dari banyaknya jumlah kota yang dibuat. Pada contoh kasus 5 kota, jumlah unit jaringan yang terbentuk adalah 25 unit (*neuron*). Unit-unit jaringan tersebut selanjutnya diberi nilai awal untuk kemudian dihitung *output* dan nilai energi jaringan mula-mula. Gambar 3 menunjukkan nilai inisial *input,output* dan energi jaringan.



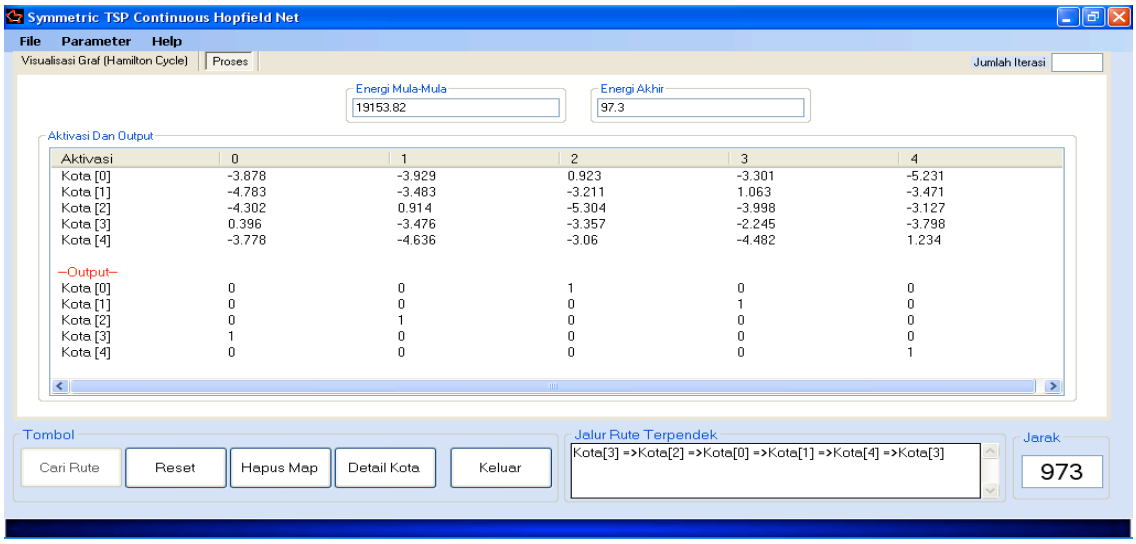
Gambar 3. Nilai Inisialisasi Unit Jaringan

3. Pencarian jarak terpendek

Untuk melakukan pencarian jarak terpendek dari rute yang valid, maka sistem akan melakukan proses *update* nilai unit jaringan. Proses *update* tersebut menggunakan informasi nilai inisialisasi *input* dan *output* masing-masing unit. Pemilihan unit jaringan yang nilainya akan diperbaharui akan dilakukan secara *random* sesuai dengan proses *update* pada *Continuous Hopfield Net*. *Output* dari unit yang terpilih akan langsung dihitung untuk

selanjutnya digunakan dalam proses *update* unit jaringan yang lainnya (*asyncronous*). Untuk kasus 5 kota, 1 kali proses *update* unit jaringan akan dilakukan selama 25 kali perulangan (n x n, dengan n sebagai jumlah kota). Setelah proses *update* dilakukan, selanjutnya sistem akan menghitung nilai energi jaringan yang baru. Jika kondisi *output* dari masing-masing unit belum membentuk rute yang valid (*konvergen*), maka proses update jaringan akan dilakukan kembali. Setelah beberapa kali proses *update*, kondisi *output* masing-masing unit jaringan akan membentuk rute yang valid sesuai dengan batasan fungsi energi pada *Continuous Hopfield Net*. Sistem kemudian akan menggunakan informasi *output* yang valid untuk menghitung jarak tempuh rute dan nilai energi akhir jaringan dalam penggambaran graf pada papan area.

Hasil yang paling optimal dalam mencari rute terpendek adalah dengan mendapatkan nilai energi akhir jaringan yang terkecil. Untuk itu, proses percobaan dilakukan beberapa kali sehingga diperoleh beberapa rute yang valid dengan kondisi nilai akhir jaringan yang berbeda-beda. Gambar 4 menunjukkan kondisi akhir jaringan dengan nilai energi akhir yang terkecil pada kasus 5 kota.



Gambar 4. Output Unit Jaringan

4. Analisis Hasil Implementasi

Setelah menganalisis proses kerja sistem dalam menemukan jalur terpendek pada contoh kasus 5 kota, kemudian dilakukan analisis hasil implementasi dari sistem terhadap beberapa kota yang lainnya. Bentuk analisis yang dilakukan dengan menggunakan parameter *default*(A = 500, B = 500, C = 200, D= 0.1) untuk jumlah percobaan yang berbeda-beda dan menghitung waktu komputasi dari percobaan tersebut. Analisis berikutnya adalah dengan mengubah salah satu parameter yaitu nilai parameter D, untuk melihat hasil rute valid yang dapat diperoleh dari perubahan tersebut

a. Analisis Parameter *default (A = 500, B = 500, C = 200, D = 0.1)*

Tabel 3. Percobaan Jaringan

Jumlah kota	Proses (Epoch)	Nilai Energi		Jarak
		Awal	Akhir	
10	50	277725.306	170.8	**1708**
	90	277600.723	172.4	1724
15	50	1374989.287	290.4	2904
	90	1362434.449	255.8	**2558**
20	50	4247520.232	421.1	4211
	90	4262065.91	374.7	**3747**
25	50	10281371.944	524.8	5248
	90	10313204.586	472.9	**4729**

b. Analisis Waktu Perhitungan

Tabel 4. Waktu Perhitungan

Jumlah kota	Proses (Epoch)	Waktu Perhitungan
10	50	6 menit 59 detik 570 milliseconds
	90	12 menit 56 detik 586 milliseconds
15	50	36 menit 12 detik 422 milliseconds
	90	1 jam 1 menit 14 detik 455 milliseconds
20	50	> 1 jam 30 menit
	90	> 3 jam
25	50	> 2 jam
	90	> 4 jam

c. Analisis Perubahan Parameter

Tabel 5. Perubahan Parameter

Parameter	Jumlah Kota	Jumlah Proses (Epoch)	Valid	Gagal
A = 500, B = 500 , C = 200 , D = 0.1 (Default)	10	50	50	0
	15	50	50	0
	25	50	50	0
A = 500, B = 500 , C = 200 , D = 0.3	10	50	41	9
	15	50	35	15
	25	50	30	20
A = 500, B = 500 , C = 200 , D = 0.5	10	50	2	48
	15	50	1	49
	25	50	0	50
A = 500, B = 500 , C = 200 , D = 0.7	10	50	0	50
	15	50	0	50
	25	50	0	50

Tabel 3 memperlihatkan bahwa energi yang terkecil yang diperoleh selama percobaan akan secara langsung menunjukkan jarak tempuh yang paling optimal terhadap suatu rute tertentu. Dengan menambah jumlah percobaan atau *epoch* maka akan dapat menghasilkan jarak yang semakin optimal.

Tabel 4 memperlihatkan waktu perhitungan terhadap sejumlah kota dengan jumlah percobaan yang berbeda-beda. Terlihat bahwa jumlah kota yang bertambah akan secara langsung menambah lamanya waktu komputasi. Hal ini disebabkan karena, pertambahan jumlah kota akan secara langsung menambah jumlah unit neuron yang harus berkoordinasi

untuk mendapatkan suatu output yang valid, yaitu n * n (n=jumlah kota). Selain itu faktor dari spesifikasi komputer yang digunakan selama pengujian juga sangat berpengaruh terhadap kinerja dari sistem tersebut.

Table 5 menunjukkan bagaimana perubahan jumlah rute valid terhadap perubahan nilai parameter yang dilakukan yaitu parameter D.. Pada perubahan tersebut nilai parameter D secara berkala dinaikkan sebesar 0.2 , lalu dihitung jumlah rute valid yang terbentuk. Dari percobaan tersebut diperoleh bahwa semakin besar nilai parameter D dari kondisi awal, maka jumlah rute valid yang dapat diperoleh juga semakin sedikit.

5. Penutup

Berdasarkan analisis dan implementasi, maka dapat diambil kesimpulan sebagai berikut :

- Penerapan metode *Continuous Hopfield Net* dalam menemukan rute perjalanan valid sangat dipengaruhi oleh parameter fungsi energi dan *update* unit jaringan, yaitu parameter yang masing-masing bernilai A=500, B=500, C=200 , D=0.1. Perubahan dari nilai parameter tersebut dapat menyebabkan tidak ditemukannya rute yang valid.
- Bertambahnya jumlah kota akan secara langsung menambah jumlah unit jaringan yang harus berkordinasi untuk membentuk suatu rute perjalanan, yaitu n x n (n adalah jumlah kota). Akibat dari jumlah unit jaringan yang bertambah adalah meningkatnya waktu perhitungan selama proses pembentukan suatu rute perjalanan.
- Penambahan jumlah epoch atau percobaan pada metode *Continuous Hopfield Net* dapat menghasilkan rute dengan jarak yang lebih optimal.
- Nilai energi *Continuous Hopfield Net* yang terkecil akan secara langsung menandakan rute perjalanan terpendek yang paling optimal selama percobaan berlangsung,

Sebagai pengembangan dari sistem yang telah dibuat, dapat dilakukan suatu perbandingan dengan menerapkan algoritma lainnya kedalam sistem, sehingga kinerja dari metode *Continuous Hopfield Net* dalam menemukan rute terpendek akan lebih teruji.

Daftar Pustaka

[1] Craig, J.C., Patrick, T. (2006). Visual basic 2005 cookbook. O'Reilly, USA.

[2] Gutin, G., Punnen, A., Barvinok, A., Gimadi, E. K., Serdyukov, A.I. (2002). The traveling salesman problem and its variations. Diambil dari http://www.math.lsa.umich.edu/~barvinok/tspch.ps.

[3] Hopfield JJ, Tank DW (1985). "Neural" computation of decisions in optimization problems. Biological Cybernetics 52, halaman 141-152

[4] Laurene V. Fausett, Fundamentals of Neural Networks. Architectures. Algorithms and Applications

[5] Nagórny, Z (2009). Application of a Modified Hopfield Network to the Traveling Salesman Problem

[6] Priyanto, R. (2008). Langsung Bisa visual Basic 2008. Andi Offset

[7] Puspitaningrum, D. (2006). Pengantar jaringan Saraf Tiruan. Andi Offset

[8] Siang, J, J.(2002). Matematika Diskrit dan Aplikasinya Pada Ilmu Komputer. Andi Offset