PENERAPAN ALGORITMA *SELF-ORGANIZING MAP* DENGAN INFORMASI STATISTIK PADA PERSOALAN *EUCLIDEAN TRAVELLING SALESMAN*

Rully Soelaiman dan M.Gandi Wijaya

Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS)
Kampus ITS, Jl. Raya ITS, Sukolilo, Surabaya 60111, Indonesia
Tel. +62 31 5939214, Fax. +62 31 5939363,
Email: rully@its-sby.edu

ABSTRAK

Self-Organizing Map (SOM), yang dikenal juga dengan Kohonen feature map, merupakan algoritma Jaringan Saraf Tiruan yang merepresentasikan arsitektur Topological Preserving Map. Algoritma tersebut menggunakan pembelajaran dengan metode unsupervised learning. Dari sisi topologi, jika terdapat sekumpulan data yang dimasukkan ke dalam SOM, maka akan terbentuk kumpulan neuron yang merupakan representasi dari data tersebut. Dan ketika memasuki proses learning, neuron-neuron tersebut dengan sendirinya akan menempati tempatnya masing-masing secara statistik dan topologis.

Dengan kata lain, pada setiap iterasi, dicari strategi dengan memanfaatkan dua buah informasi, yaitu informasi lokal dalam point yang direpresentasikan dan juga informasi global dari keseluruhan data dalam himpunan point tersebut. Penerapan metode yang menggunakan informasi statistik tersebut menjadi dasar pembentukan algoritma pembelajaran yang disebut Kohonen Network Incorporating Explicit Statistics (KNIES).

Uji coba terhadap kinerja algoitma KNIES dilakukan pada permasalahan *Travelling Salesman* Problem (TSP). Kumpulan kasus-kasus TSP disediakan dalam pustaka yang disebut TSPLIB. Dalam TSPLIB juga disertakan hasil optimum tour yang dilakukan oleh algoritma eksak. Dari kumpulan kota tersebut, diujikan pada perangkat lunak KNIES dengan radius 0.5, diperoleh hasil bahwa tour terbaik didapatkan oleh Pr107 (0.42%), hasil terburuk adalah (9.70%) didapatkan oleh Pcb442. Sedangkan hasil rata-rata adalah 3.85 %.

Kata kunci : KNIES, Neural Network, Self-Organizing Map, *Travelling Salesman* Problem.

1. PENDAHULUAN

Artificial neural network adalah sebuah sistem untuk pemroses informasi yang mempunyai ciri umum seperti sistem saraf biologis. Neural network dikembangkan sebagai generalisasi permodelan pengamatan manusia atau sering disebut neural biologi. Neural network secara topologi dapat dibagi menjadi beberapa macam : *adaptive learning*, *self-organizing*, *error tolerance*, *real time operation*, dan *paralel information processing*.

Pada model self-organizing, ketika data diberikan ke dalam neural network, data akan mengatur struktur dirinya sendiri untuk merefleksikan properti-properti dari data yang diberikan. Pada kebanyakan model neural network, batasan self-organizing mengacu pada determinasi kekuatan antar neuron .

Pada topologi *organized feature map*, masing-masing neuron dalam network berisi sebuah *feature vector*. Ketika sebuah pola dari *data training* diberikan pada network, neuron yang corak vektornya terdekat dengan input vektor akan diaktifkan (activated). Hal tersebut akan menyebabkan perubahan pada input vektor yang menyebabkan pengaktifan. Dalam proses perubahan tersebut, dapat dilakukan untuk mendekati arah input vektor atau menjauh dari input vektor (tergantung algoritma learning yang digunakan). Model network yang memiliki tingkah laku seperti ini diantaranya adalah *Self-Organizing Map* dari Kohonen.

Self-Organizing Map (SOM) diperkenalkan oleh Teuvo Kohonen pada tahun 1982. SOM (yang juga dikenal sebagai Kohonen *feature* map) adalah salah satu algoritma neural network terbaik. Metode ini cukup unik karena membangun sebuah *topology preserving map* dari ruang berdimensi tinggi ke dalam neuron-neuron sebagai representasi dari datapoint-datapoint yang ada.

SOM sudah dipergunakan dalam bermacam-macam aplikasi. Dalam *statistical pattern recognition*, digunakan untuk pengenalan bahasa Finlandia dan Jepang (Kohonen, Makisara & Samaraki, 1984; Kohonen, Torkkola, Shozokai, Kangas & Venta, 1987). Penerjemahan kalimat (Samarabandu & Jakubowicz, 1990), klasifikasi *sea-ice* (Orlando, Mann & Haykin, 1990), dan dalam klasifikasi suara serangga (Neumann, Wheeler, Burnside, Bernstein & Hall, 1990). Pada perangkat keras, dalam *low level*, digunakan pada kendali produksi *semi-conductor substrates* (Marks & Goser, 1988; Triba, Marks Rutcker

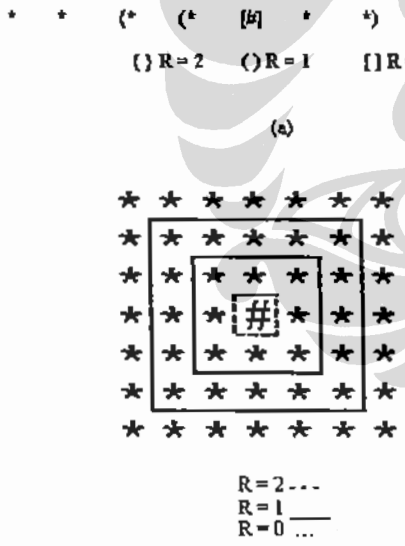
& Goser, 1988) dan untuk *high level*, pada sintesis sistem digital (Hermani & Postula, 1990).

2. ARSITEKTUR DAN MODEL PEMBELAJARAN *SELF-ORGANIZING MAP*

Dari sisi topologi, SOM menganut topologi self-organizing map, sehingga jika sekumpulan data dimasukkan ke dalam SOM, akan dibentuk kumpulan neuron yang merupakan representasi dari data, dan ketika memasuki proses learning, neuron-neuron tersebut dengan sendirinya akan menempati tempatnya sendiri-sendiri secara statistik dan topologis. SOM menganut arsitektur *topological feauture map* atau sering disebut juga *topological preserving map*. Sedangkan untuk model pembelajaran, metode ini menganut *unsupervised learning*.

2.1. Arsitektur Self-Organizing Map

Ketika dalam suatu training terdapat sebuah neuron yang menang (*winning neuron*), maka yang akan di-update tidak hanya neuron yang menang saja, tetapi juga neuron yang terletak di sekitarnya, konsep ini disebut *neighborhood*. Konsep neighborhood dapat berupa *linear array* (lateral) dan *rectangular grid*.



Gambar 1.
a. Konsep ketetanggaan untuk *linear array*
b. Konsep ketetanggaan untuk *rectangular grid*

Gambar 1.a. menunjukkan konsep ketetangaan lateral, dalam permodelan tersebut neuron disusun dalam suatu array atau list satu dimensi. Misalkan neuron dimasukkan dalam array atau list neuron yang diberi nama Y dengan jumlah neuron 10 buah, dimana $Y_j = \{1 \le j \le 10\}$. Ketika sebuah neuron memenangkan suatu training, misalkan neuron ke 5 (Y_5), maka Y_5 disebut closest neuron. Jika parameter neighborhood di set 1, maka satu indeks neuron di sebelah kiri Y_5, yaitu Y_4, dan sebuah neuron di sebelah kanan Y_5, yaitu Y_7, masuk dalam himpunan neighborhood. Begitu pula jika parameter neighborhood di set 2, berarti 2 indeks neuron di sebelah kiri Y_5, yaitu Y_4 dan Y_3, serta dua buah indeks neuron di sebelah kanan Y_5, yaitu Y_6 dan Y_7, masuk dalam himpunan neighborhood. Konsep neighborhood semacam ini dapat digunakan untuk menyelesaikan permasalahan Euclidean TSP.

Pada gambar 1.b. konsep ketetanggaan berbentuk rectangular grid. Model seperti ini biasanya digunakan untuk indeks neuron 2 dimensi. Cara penentuan ketetanggaan hampir sama dengan lateral, hanya himpunan neighborhood bukan hanya di sebelah kiri dan kanan saja, tetapi juga atas-bawah, samping kiri-kanan atas dan samping kiri-kanan bawah. Konsep neighborhood seperti ini biasanya digunakan pada aplikasi *image processing*.

2.2. Algoritma Pelatihan SOM

Diasumsikan terdapat sebuah himpunan sejumlah N point, dimana $P = \{X_i : 1 \le i \le N\}$ dalam sebuah ruang d-dimensi. Tujuan utama SOM adalah merepresentasikan point-point pada P dengan M representatif point, atau dengan neuron-neuron, $Y = \{Y_j : 1 \le j \le M\}$. Dalam model pattern recognition, neuron-neuron secara umum diasumsikan sebagai $M \le N$, atau dengan kata lain jumlah neuron lebih kecil sama dengan jumlah datapoint.

Himpunan neuron $Y = \{Y_j : 1 \le j \le M\}$ diinisialisasi secara random (atau dengan menggunakan pola tertentu) dalam convex hull datapoint yang direpresentasikan, P. Langkah berikutnya mengaktifkan neuron yang nantinya akan diikutkan dalam proses learning.

Neuron-neuron hasil inisialisasi berkompetisi terhadap datapoint X_i yang direpresentasikan ke network. Neuron paling dekat dengan datapoint X_i adalah pemenang dari kompetisi dan diberi nama closest neuron (winning neuron). Kemudian ditentukan neuron-neuron disekitarnya yang masuk dalam himpunan neighborhood. Penentuan himpunan neighborhood ini dapat dilakukan secara lateral atau rectangular (seperti pada gambar 2.1). Untuk selanjutnya neighborhood akan disebut dengan "Activation Bubble", yang dinotasikan dengan B_{j^*}.

Selanjutnya, closest neuron dan neuron-neuron dalam himpunan activation bubble akan bergerak ke arah datapoint yang direpresentasikan, X_i. Perpindahan neuron-neuron tersebut dicapai dengan mengikuti algoritma peng-*update*-an sebagai berikut :

$$Y_j(t+1) = \alpha(t)Y_j(t) + \alpha(t)X_i \text{ if } j \in B_{j^*}(t)$$
$$= Y_j(t) \quad \text{if } j \notin B_{j^*}(t) \qquad (1)$$

Dimana t adalah indeks waktu (iterasi). Algoritma dasar ini mempunyai dua parameter fundamental. $\alpha(t)$ dan ukuran dari himpunan $B_j(t)$. $\alpha(t)$ dinamakan konstanta adaptasi (*learning rate*) dan mempunyai nilai $0<\alpha(t)<1$. Kohonen dan lain-lain (Gray, 1984; Kohonen, 1990, 1995; Linde et al., 1980; Wong, 1996) merekomendasikan penurunan $\alpha(t)$ secara linear, dengan nilai 0.2 untuk pemilihan terbaik.

Ukuran dari bubble diinisialisasi cukup besar agar penataan global dapat terbentuk. Konsekuensinya semua neuron mempunyai kecenderungan mengikat dirinya ke dalam ikatan (simpul) untuk sebuah nilai $\alpha(t)$ secara erat sebagai suatu kesatuan. Saat resolusi *coarse spatial* dicapai, hanya neuron terdekat dengan datapoint yang tepengaruh oleh datapoint tersebut. Resolusi *coarse spatial* adalah kumpulan dari neuron-neuron yang tidak pernah memenangkan kompetisi terhadap datapoint yang direpresentasikan. Dalam metode SOM. *coarse spatial* diabaikan karena tidak berpengaruh pada hasil akhir dari keseluruhan proses.

Ada dua cara untuk mempengaruhi ukuran bubble. Pertama adalah dengan secara eksplisit menurunkan ukuran bubble. dan ini dilakukan dengan penurunan linear:

$$W(t+1) = K_w W(t) \qquad (2)$$

Dimana K_w adalah konstanta *user defined*, dan $[-W, W]$ adalah activation bubble. Cara alternatif lain adalah memperkenalkan fungsi bobot yang nilai penurunannya tergantung jarak dari closest neuron.

2.3. Perbaikan *Self-Organizing Map* Dengan Informasi Statistik Global

Meskipun pada SOM akhirnya neuron-neuron akan mematuhi distribusi dari point-point original, sayangnya banyak informasi yang sebenarnya tidak diperlukan. Berdasarkan hal tersebut, dapat disusun perbaikan metode SOM dengan menggunakan informasi yang telah dikumpulkan dan residen dalam datapoint yang direpresentasikan, yaitu menggunakan informasi-informasi yang berhubungan dengan potongan-potongan informasi statistik. Dengan kata lain, pada setiap iterasi, dicari strategi dengan memanfaatkan dua buah informasi, yaitu informasi lokal dalam point yang direpresentasikan dan juga informasi global dari keseluruhan data dalam himpunan, *P*. selanjutnya akan dibahas bagaimana hal tersebut dicapai.

Dimisalkan, mean dari himpunan point-point yang direpresentasikan oleh neuron-neuron dalam P adalah μ. Pertama yang dilakukan adalah me-*maintain* μ untuk keseluruhan training. Yang menjadi masalah adalah bagaimana ketika neuron-neuron secara kolektif terupdate dapat dikendalikan oleh meannya yang juga bernilai μ. Untuk itu, dilakukan dekomposisi untuk tiap-tiap training ke dalam dua fase, yang dinamai fase **attracting** dan **dispersing**.

Ketika datapoint X_i direpresentasikan ke network. closet neuron dan semua neuron dalam activation bubble berpindah menggunakan sebuah persamaan yang serupa dengan (1). Aturan ini disebut modul attracting. dan dalam fase ini neuron-neuron dalam bubble menggunakan informasi lokal yang ada dalam point-point. baik ketika muncul secara individu. maupun ketika menggerakkan semuanya secara bersama-sama.

Dengan mengamati pemisahan neuron-neuron ketika ikut dalam fase attracting dan yang tidak, dalam KNIES dapat diasumsikan bahwa activation bubble di maintain tetap untuk sebuah nilai tertentu dari M. Penurunan jarak yang ditempuh oleh masing-masing neuron dalam himpunan activation bubble ke datapoint pada fase attracting disarankan dicapai dengan menggunakan sebuah fungsi kernel. Untuk mudahnya, diasumsikan kernel tersebut adalah kernel Gaussian (Duda & Hart, 1973; Fukunaga, 1990), yaitu :

$$G(a,b) = K_g \exp\left(\frac{-\| Y_a - Y_b \|^2}{2\sigma^2}\right) \qquad (3)$$

dimana K_g adalah konstanta normalisasi, σ adalah standar deviasi dari kernel yang nilainya selalu diturunkan (misalkan secara linear) untuk mendapatkan efek dari penurunan aktifitas bubble, dan Y_a dan Y_b adalah koordinat-koordinat dari point yang dicari. Sehingga fase attracting mengikuti aturan :

$$Y_j(t+1) = \begin{cases} Y_j(t) + G(j, j^*)(X_i(t) - Y_j(t)) & \text{jika } j \in B_{j^*}(t) \\ Y_j(t) \end{cases} \qquad (4)$$

dimana X_i adalah datapoint ke-i yang direpresentasikan ke network dan j^* adalah closest neuron (menggunakan bentuk Euclidean) ke X_i. Keuntungan penggunaan sebuah fungsi kernel adalah bahwa sebuah parameter tunggal σ dapat menangani activation bubble dan intensitasnya dalam attracting.

Sebagai hasil dari perpindahan neuron dalam activation bubble, terlihat bahwa mean dari semua neuron berpindah jauh dari inisial nilai yang ditunjuk, μ. Dengan demikian sekarang, fase dispersing menggerakkan neuron-neuron di luar activation bubble untuk berpindah jauh dari lokasi sekarang ke letak yang baru dengan menggunakan perhitungan tertentu. Total perpindahan dipengaruhi sebagai hasil dari modul attracting adalah vektor $\Delta Y(t)$, dimana :

$$\Delta Y(t) = \sum_{j \in B_{j^*}(t)} \left[Y_j(t+1) - Y_j(t) \right] \qquad (5)$$

perubahan $\Delta Y(t)$ ini disebarkan ke neuron-neuron yang tidak ikut dalam attracting sebagai :

$$Y_j(t+1) = Y_j(t) + \frac{1}{M(t) - |B_{j^*}(t)|} \Delta Y \qquad (6)$$

3. PENYELESAIAN TSP DENGAN KNIES

Dalam algoritma-algoritma heuristik klasik untuk riset operasional, beberapa solusi neural network pernah ditujukan untuk menyelesaikan TSP. Penelitian dari persoalan ini dapat ditemukan di Potvin (1993). Sukses pertama untuk penelitian ini adalah hasil kerja dari *Hopfield and Tank* (1985). Model dasar Hopfield-Tank adalah dengan mencari penurunan gradient untuk mendapatkan *local minimum* dari fungsi energi *E*. Hal Ini disebabkan local minimum berhubungan dengan solusi yang bagus untuk penyelesaian TSP. Namun begitu tidak ada jaminan, karena tidak semua energi minimumnya merepresentasikan solusi untuk TSP.

Disamping model Hopfield-Tank, dua skema neural network lain untuk TSP juga dilaporkan. Pertama adalah pendekatan Elastic Net dimana sebuah local minimum dari sebuah kecocokan fungsi energi ditemukan dengan cara penurunan gradient (*Dublin & Willshaw*, 1987; *Simic*, 1990) kedua model dapat diturunkan dari prinsip mekanika statistik. Model kedua adalah pendekatan SOM, yang merupakan topik utama dari penelitian ini.

3.1. Penyelesaian KNIES-TSP

Pertama-tama diasumsikan jumlah neuron pada suatu saat adalah $M(t)$, dan jumlah ini dapat bertambah atau berkurang dengan cara penyisipan (insertion) atau penghapusan (deletion) neuron. Pada suatu saat 't', dapat dikatakan bahwa sebuah neuron j yang berlokasi di $Y_j(t)$ terhubung dengan sebuah kota (datapoint $T_i(t)$ jika :

$$T_j(t) = \min_i \|X_i - Y_j(t)\| \quad j = 1,..,M(t) \qquad (7)$$

sehingga pada saat 't', $T_j(t)$ adalah kota terdekat ke neuron j, dan ini direpresentasikan oleh $Y_j(t)$. Untuk setiap saat, sebuah datapoint X_i direpresentasikan ke network. Neuron terdekat ke X_i, yaitu j*, ditentukan sebagai pemenang (closest neuron). Neuron ini dan neuron-neuron dalam himpunan activation bubble digerakkan ke arah X_i dengan kekuatan yang ditentukan oleh sebuah fungsi kernel. Kernel yang digunakan adalah kernel Gaussian dengan standar deviasi σ seperti yang disebutkan pada formula no. (9), sebab penurunan nilai σ juga berpengaruh ke ukuran activation bubble dan learning rate, α(t). Berikutnya, kumpulan neuron di luar activation bubble dipindahkan dengan menggunakan mean neuron-neuron yang ada di ring dan mean dari datapoint-datapoint yang direpresentasikannya. Sebagai catatan, di sini mean dari neuron-neuron tidak dibuat konvergen dengan mean datapoint, namun hanya dibuat bergerak ke arah mean dari datapoint.

Selanjutnya, jumlah neuron M(t) akan berubah setiap saat 't'. Penyisipan dan penghapusan diijinkan dengan mengggunakan cara yang sama dengan prosedur Angeniol, Vaubois & Le Texier (1988). Jika sebuah neuron memenangkan kompetisi terhadap dua kota yang berbeda dalam iterasi yang sama, maka sebuah neuron baru dibuat dengan koordinat yang sama dengan closest neuron. Neuron baru ini disisipkan ke dalam ring sebagai neighbor dari *closest neuron*. Keduanya di-inhibit untuk satu iterasi berikutnya. Dengan cara yang sama sebuah neuron dihapus, jika neuron tersebut tidak memeangkan kompetisi untuk sejumlah iterasi.

Perbedaan penyisipan/penghapusan KNIES dengan Angeniol, Vaubois & Le Texier (AVL) sangat kecil. Ketika sebuah neuron k, disisipkan sebab sebuah neuron l memenangkan dua kali kompetisi. Maka, $Y_k(t)$ sama persis dengan $Y_l(t)$ jika ditinjau dari letaknya, tetapi $T_k(t)$ ≠ $T_l(t)$ karena merepresentasikan kota yang berbeda.

Fase *dispersing* pada KNIES adalah untuk memperbesar efek ketika keduanya berada di dalam ring dan juga keduanya memberikan kontribusi yang sama ke *mean* dari kumpulan neuron, meskipun kota-kota yang direpresentasikan oleh keduanya adalah berbeda. Dengan demikian efek dari modul *dispersing* menyebabkan neuron-neuron menjadi benar-benar berbeda letaknya dibandingkan jika hanya semata-mata menggunakan prosedur AVL. Dengan cara yang sama, sebuah neuron jika dihapus tidak akan berada lagi di ring, dan koordinatnya juga hilang, sehingga neuron yang dihapus tidak memberikan kontribusi ke *mean* dari neuron-neuron.

Algoritma KNIES-TSP formal bekerja dalam prinsip yang sama dengan dengan algoritma KNIES. Perbedaanya adalah pada penyisipan dan penghapusan neuron.notasi untuk algoritma ini adalah sebagai berikut :

- **Won(j)**, record-record jumlah berapa kali neuron j memenangkan kompetisi dalam suatu iterasi (epochs).
- **Insert(j)**, menyisipkan sebuah neuron di lokasi yang sama dengan neuron j, yang diindeks sebagai j+1.
- **Inhibit(j)**, adalah array boolean yang mengindikasikan apakah neuron j terhalang (*inhibited*) atau tidak.
- **Delete(j)**, menghapus neuron j dari kumpulan neuron di ring.

3.2 Penyederhanaan KNIES-TSP dengan KNIES-TSP-Global

KNIES–TSP menyebakan komputasi yang mahal. Hal ini disebabkan pada setiap iterasi selalu me-*maintain* mean dari neuron-neuron menjadi mean dari datapoint-datapoint yang direpresentasikannya, dan dihadapkan pada masalah identifikasi kota yang terhubung dengan masing-masing neuron. Akibatnya himpunan $\{X_{T_j(t)} : j = 1,..,M(t)\}$ harus diidentifikasi setelah proses dari masing-masing kota.

Modifikasi KNIES–TSP (dinamakan KNIES–TSP–Global) melakukan pendekatan dengan cara selalu menggerakkan (bukan memindahkan) neuron-neuron ke arah global *mean* dari semua kota.

Algoritma KNIES-TSP-Global
Input: koordinat-koordinat dari N datapoint $X_i; 1 \le i \le N$.
Output: Penyelesaian TSP
Parameter: ω, besar dari ukuran activation bubble, B_{j^*} σ, standar deviasi kernel Gaussian yang diturunkan, M_0, jumlah neuron yang diinisialisasi pertama kali.
Begin
Inisialisasi jumlah neuron (M) oleh M_0 dan lokasi dari M neuron di ring
t=0
Repeat
For j=1,..,M **do**
Won(j)=0
Inhibit(j)=false
EndFor
For setiap datapoint $X_i \in P$ **do**
$j^* \leftarrow \text{argmin}\,|X_i - Y_j|$
if(inhibit(j^*)) and (neuron j^* menutupi neighbor neuron) **go to** jump
Won(j^*) ← **Won**(j^*)+1
If(Won(j^*)=2 **then**
MustInsert ← true
Won(j^*)=0
EndIF
For all $j \in B_{j^*}$ **do**
$\Delta Y_j \leftarrow \Lambda(j,j^*)(X_i - Y_j)$, dengan Λ seperti yang terdefinisi pada (3)
$Y_j \leftarrow Y_j + \Delta Y_j$
EndFor
$X_{total} = \sum_j X_{Tj}$
$Y_{total} = \sum_j Y_j$
$\Delta Y_{total} = X_{total} - Y_{total}$
For all $j \notin B_{j^*}$ **do**
$Y_j \leftarrow Y_j + \frac{1}{M(M-1-B_{j^*})} \Delta Y$
EndFor
If(MustInsert) then
Insert(j^*)
M ← M+1
Inhibit(j^*) ← true
Inhibit(j^*+1) ← true
EndIf
t ← t+1
jump:**EndFor**
For j=1,..,M **do**
If (j tidak memenangkan neuron untuk lebih dari 3 kali iterasi) **then**
delete(j)
M ← M-1
EndIf
EndFor
Decrement σ
Until kondisi optimal dicapai
TSP_tour ≤ tour mengelilingi kota dengan urutan dari indeks neuron yang ada dalam ring.
End Algoritma KNIES-TSP-Global

4. UJI COBA PERANGKAT LUNAK

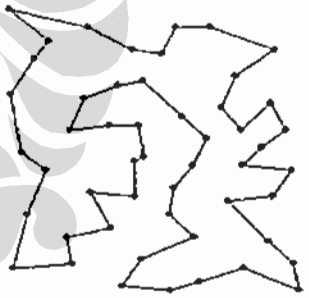
Uji coba perangkat lunak dilakukan dengan menggunakan set persoalan TSP dari TSPLIB[18]. TSPLIB adalah kumpulan kasus-kasus TSP, baik untuk Euclidean TSP maupun Asymetric TSP.

Dari kumpulan kota tersebut, yang akan diujikan untuk perangkat lunak KNIES yang dibuat adalah :

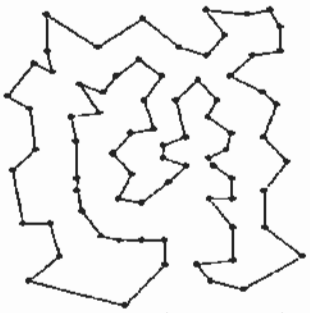
Tabel 1. Kasus TSP standar yang diujikan

KASUS	JUMLAH KOTA	TOUR OPTIMUM TSPLIB
Bier127	127	118,282
Eil51	51	426
Eil76	76	538
Eil101	101	629
KroA200	200	28,568
Lin105	105	14,379
Pr107	107	44,303
Pr124	124	59,030
Pr136	136	96,772
Pr152	152	73,682
Rd100	100	7,910
St70	70	675

Contoh optimum tour yang ditakukan oleh TSPLIB diambilkan dari kasus-kasus yang akan diujikan.



Gambar 2. Tour optimum kasus Eil51



Gambar 3. Tour optimum kasus Eil76

Ujicoba perangkat lunak ini dilakukan dengan cara memasukkan kasus standar yang dimaksudkan diikuti dengan kombinasi parameter yang sesuai. Parameter yang dimasukkan adalah ada 4 : Jumlah Neuron (M), standar deviasi kernel Gaussian (σ), konstanta penurunan kernel Gaussian, (K_σ) dan activation bubble (ω).

Dari ketiga ukuran jari-jari inisialisasi, dilihat bahwa hampir semua hasil tour paling optimum didapatkan untuk jari-jari 0.5, yaitu pd Bier127, Eil51, Eil76, Eil101, KroA200, Lin105, Pr107, Pr124, Pr136, Pr152, Rd100 dan Pcb442. Satu-satunya hasil paling optimum yang dicapai oleh jari-jari selain 0.5 adalah pada kasus St70, yaitu oleh jari-jari 0.75. Dari tabel perbandingan hasil KNIES dengan optimum TSPLIB dapat diketahui bahwa tour terbaik didapatkan oleh Pr107 (0.42%), hasil terburuk adalah (9.70%) didapatkan oleh Pcb442. Sedangkan hasil rata-rata adalah 3.85 %.

5. KESIMPULAN

Berdasarkan hasil uji coba dan evaluasi dari perangkat lunak yang telah berhasil diimplementasikan ditarik beberapa kesimpulan seperti berikut:

- Perangkat lunak hasil perancangan cukup optimal untuk menyelesaikan permasalahan optimasi, khususnya Travelling Salesman.
- Uji coba terhadap kinerja algoitma KNIES dilakukan pada Kumpulan kasus-kasus TSP, baik untuk Euclidean TSP maupun Asymetric TSP disediakan dalam pustaka yang disebut TSPLIB. Dari kumpulan kota tersebut, yang diujikan pada perangkat lunak KNIES dengan r=0.5, diperoleh hasil bahwa tour terbaik didapatkan oleh Pr107 (0.42%), hasil terburuk adalah (9.70%) didapatkan oleh Pcb442. Sedangkan hasil rata-rata adalah 3.85 %.

Tabel 2. Tabel ujicoba inisialisasi neuron berbentuk lingkaran

Kasus Standar TSPLIB	Parameter	*Jari-jari*		
		0.25	0.5	0.75
Bier127	125,55,0.8,0.1	123,914.91	122,081.01	123,777.09
Eil51	30,30,0.8,0.2	442.86	438.59	443.92
Eil76	25,25,0.8,0.1	565.45	565.45	565.45
Eil101	10,35,0.9,0.2	685.42	662.45	662.45
KroA200	160,53,0.8,0.2	31,002.29	30,493.45	31,581.35
Lin105	50,35,0.9,0.2	14,726.10	14,685.17	14,891.68
Pr107	60,25,0.9,0.1	45,329.30	44,489.21	44,986.66
Pr124	125,15,0.8,0.15	61,829.82	60,275.17	61,929.16
Pr136	30,35,0.9,0.05	104,330.61	101,230.32	101,230.32
Pr152	60,15,0.9,0.25	76,266.93	75,273.44	76,095.78
Rd100	60,10,0.9,0.15	8,309.70	8,172.02	8,289.48
St70	40,10,0.9,0.15	703.31	703.31	690.29
Pcb442	50,40,0.9,0.2	57,730.22	55,704.48	57,730.22

Tabel 3. Perbandingan Tour hasil ujicoba perangkat lunak untuk inisialisasi neuron berbentuk lingkaran dengan jari-jari 0.25, 0.5 dan 0.75 dengan hasil optimum yang dicapai oleh tour optimum TSPLIB

	Tour Optimum TSPLIB	Jari - jari r=0.25	%	r=0.5	%	r=0.75	%
Bier127	118,282.00	123,914.91	4.76%	122,081.01	3.21%	123,777.09	4.65%
Eil51	426.00	442.86	3.96%	438.59	2.96%	443.92	4.21%
Eil76	538.00	565.45	5.10%	565.45	5.10%	565.45	5.10%
Eil101	629.00	685.42	8.97%	662.45	5.32%	662.45	5.32%
KroA200	28,568.00	31,002.29	8.52%	30,493.45	6.74%	31,581.35	10.55%
Lin105	14,379.00	14,726.10	2.41%	14,685.17	2.13%	14,891.68	3.57%
Pr107	44,303.00	45,329.30	2.32%	44,489.21	0.42%	44,986.66	1.54%
Pr124	59,030.00	61,829.82	4.74%	60,275.17	2.11%	61,929.16	4.91%
Pr136	96,772.00	104,330.61	7.81%	101,230.32	4.61%	101,230.32	4.61%
Pr152	73,682.00	76,266.93	3.51%	75,273.44	2.16%	76,095.78	3.28%
Rd100	7,910.00	8,309.70	5.05%	8,172.02	3.31%	8,289.48	4.80%
St70	675.00	703.31	4.19%	703.31	4.19%	690.29	2.27%
Pcb442	50,778.00	57,730.22	13.69%	55,704.48	9.70%	57,730.22	13.69%

REFERENSI

[1]. Angeniol, B.,Vaubois, C., & Le Texier, J.Y, Self-Organizing feature maps and travelling salesman problem, Neural Networks, 1, 289-293, 1988.

[2]. Aras N., Oommen B.J., Altinel I. K., *The Kohonen network incorporating explicit statistics*, School of Science, Carleton Unoversity, Ottawa, Canada, 1997.

[3]. Duda,R.O., & Hart, P.E., Pattern Classification and scene analysis, Wiley:Chicester, 1973.

[4]. Durbin, R., & Wilshaw, D., An analogue approach to the travelling salesman problem using an elastic net method,326, 689-691, 1987.

[5]. Fukunaga,K., Introduction to statistical pattern recognition, 2, San Diego:Academic Press, 1990.

[6]. Gray,D.H., Vector Quantization, IEEE ASSP Magazine,1,4-29, 1984.

[7]. Hermani, A. & Postula, A., Scheduling by self organization, Proc. Of IJCNN, Washington DC,1990

[8]. Hopfield,J.J.,& Tank,D.W., Neural computation of decisions in optimization problems, Biological Cybernetics, 52, 141-152, 1985.

[9]. Kohonen,T., The Self-Organizing Map, Proc. Of the IEEE,78,74-90,1990.

[10]. Kohonen,T., Self Organizing Maps, Berlin:Springer, 1995.

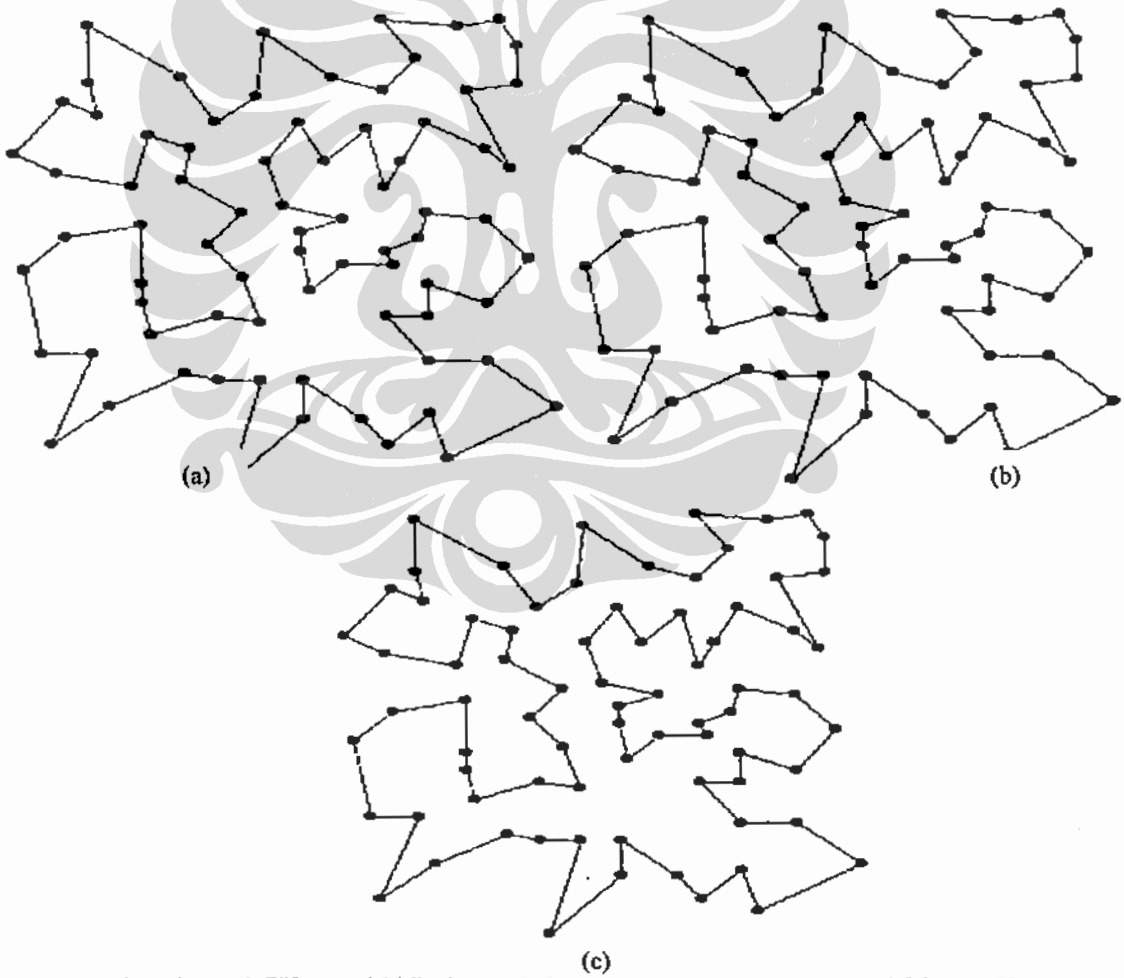
[11].Laurence Fausett, *Fundamentals of Neural Networks: Architectures, Algorithms*, and Applications, Florida Institute of Technology, Prentice-Hall Inc., 1994

[12]. Linde,Y., Buzo,A., & Gray, R.M, An Algorithm for vector quantization, IEEE Trans. On Comm.,28,61-71, 1980.

[13]. Lorens, Travelling Salesman Problem Using Kohonen Neural Networks, http:// www.patol.com /java/java.html, 1999.

[14]. Potvin,J.,Y., The travelling salesman problem : a neural network perspective, ORSA Journal on Computing, 5, 328-348, 1993.

[15]. Reinelt,G.,TSPLIB-a travelling salesman problem library, ORSA Journal on Computing,3,376-384,1991.



Gambar 4. Tour pada Eil76 untuk inisialisasi neuron berbentuk lingkaran dengan (a) r = 0.25, (b) 0.5 dan (c) 0.75