IMPLEMENTASI ALGORITMA GENETIKA DENGAN MAGNIFIED GRADIENT FUNCTION DAN DETERMINISTIC WEIGHT MODIFICATION DALAM MULITLAYER NEURAL NETWORK

Hendry Setiawan, Rully Soelaiman
Program Pascasarjana, Jurusan Teknik Informatika, Fakultas Teknologi Informasi,
Institut Teknologi Sepuluh Nopember,
Email : xieie@yahoo.com , rully@its-sby.edu

ABSTRAK

Algoritma backpropagation menggunakan pola pelatihan yang berdasarkan turunan fungsi gradien (gradient derivative function) untuk perubahan bobot dan fungsi aktifasi yang berupa fungsi sigmoid, sehingga error yang dihasilkan pada iterasi berikutnya semakin kecil. Dalam penggunaan algoritma backpropagation ada dua aspek yang menjadi pertimbangan yaitu flat spot dan laju konvergensi (convergence rate).

Penelitian berikut berusaha mengembangkan hibrid GA-MDPROP yang bertujuan mempercepat laju konvergensi dari algoritma backpropagation. Algoritma genetika digunakan untuk mecari topologi optimum dari jaringan saraf khususnya pada penentuan bobot awal, sedangkan MDPROP merupakan model hibrid Magnified Gradient Function Propagation (MGFPROP) dengan Deterministic Weight Modification (DWM). Penggunaan MGFPROP bertujuan mempercepat laju konvergensi dengan memperbesar fungsi gradient pada fungsi aktifasi, sedangkan DWM bertujuan mengurangi error dari sistem dengan mengubah bobot dengan cara yang telah ditentukan.

Hasil penelitian menunjukkan algoritma GA-MDPROP memiliki laju konvergensi yang lebih cepat dibandingkan dengan algoritma backpropagation. Algoritma GA-MDPROP juga memiliki laju konvergensi yang lebih cecpat dibandingkan dengan algoritma MDPROP khususnya untuk faktor penguatan S=1.

Kata kunci: *Algoritma Genetika, Magnified Gradient Function Propagation (MGFPROP), Deterministic Weight Modification (DWM), MDPROP, GA-MDPROP.*

1. PENDAHULUAN

Salah satu algoritma yang digunakan pada jaringan saraf yang berlapis-lapis (multilayer neural network) adalah algoritma backpropagation. Algoritma backpropagation menggunakan pola pelatihan yang didasarkan pada turunan gradien (gradient function) dan fungsi aktifasi yang berupa fungsi sigmoid untuk melakukan perubahan pada bobot, sehingga error yang dihasilkan semakin kecil.

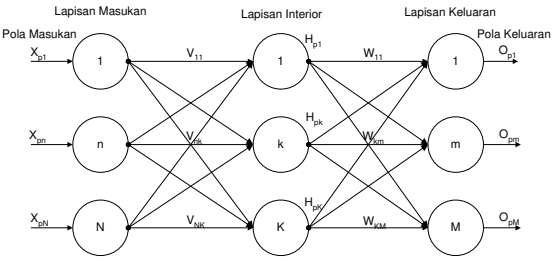
Dua aspek yang sering terjadi pada algoritma backpropagation, yaitu: terjebak dalam local minima, dimana laju perubahan error sangat kecil (lebih kecil dari nilai ambang error (threshold error) yang menyebabkan timbulnya premature saturation atau flat spot sehingga fungsi error pada iterasi (iterasi) terakhir lebih besar dari nilai ambang yang telah ditentukan dan generalisasi (global capability) tidak dapat dicapai. Premature saturation atau flat spot adalah suatu kondisi dimana dalam proses pelatihan selama beberapa iterasi secara berturut-turut error yang dihasilkan tidak mengalami perubahan dan lebih besar dari nilai ambang error yang telah ditentukan. Aspek kedua yaitu lambatnya laju konvergensi dari algoritma backpropagation.

Penelitian berikut bertujuan untuk mengatasi laju konvergensi dalam algoritma backpropagation. Dua metode yang diusulkan, yaitu:

1. Mengoptimalkan topologi jaringan saraf tiruan, khususnya dalam pembentukan bobot awal dengan algoritma genetika, dimana pada tahapan berikutnya bobot itu akan digunakan pada proses pelatihan jaringan saraf.
2. Modifikasi algoritma backpropagation, yang dilakukan dengan:
   a. Memperbesar fungsi gradient dari fungsi aktifasi (*Magnified Gradient Function Propagation* / MGFPROP) yang bertujuan untuk menangani permasalahan laju konvergensi
   b. Mereduksi error sistem dengan cara mengubah bobot (*Deterministic Weight Modification* / DWM) yang bertujuan untuk menangani generalisasi.

2. ALGORITMA BACKPROPAGATION

Algoritma backpropagation merupakan algoritma yang memiliki pola pelatihan terawasi(*supervised learning*) pada jaringan saraf yang berlapis-lapis. Pola pelatihan yang terawasi berdasarkan pada tersedianya pola masukan pada lapisan masukan dan pola target pada lapisan keluaran, sehingga setiap pola keluaran yang dihasilkan pada proses pelatihan dapat dibandingkan dengan pola target untuk mengetahui error.



Gambar 1. Jaringan Saraf Dengan Tiga Lapisan

Pada gambar 1 jaringan saraf terdiri dari 3 lapisan, yaitu sejumlah N neuron untuk lapisan masukan, sejumlah K neuron untuk lapisan interior, dan sejumlah M neuron untuk lapisan keluaran. Dua bentuk bobot, V_{11} sampai V_{NK} untuk mewakili bobot yang terletak antara lapisan masukan dan lapisan interior sedangkan W_{11} sampai W_{KM} untuk mewakili bobot yang terletak antara lapisan interior dan lapisan keluaran. Variabel *p* menyatakan jumlah pola pelatihan, sehingga untuk setiap pola pelatihan ke-*p* nilai masukan pada neuron lapisan masukan ke-*n* diwakili dengan X_{pn}, nilai keluaran pada lapisan interior ke-*k* diwakili dengan H_{pk}, nilai keluaran pada lapisan keluaran ke –*m* diwakili dengan O_{pm}, dan nilai target pada lapisan keluaran ke-*m* diwakili dengan t_{pm}. Fungsi aktifasi yang digunakan untuk menghitung output pada semua neuron dalam lapisan interior maupun lapisan keluaran adalah fungsi sigmoid, dengan formula:

$$f(net) = \frac{1}{1+e^{-net}} \qquad (1)$$

Dimana *net* merupakan sigma perkalian antara pola masukan dengan bobot, yang diformulasikan:

$$net = x_1 w_1 + x_2 w_2 + x_3 w_3 = \sum_i x_i w_i \qquad (2)$$

Turunan dari fungsi aktifasi:

$$f'(x) = f(x)\left(1 - f(x)\right) \qquad (3)$$

Langkah-langkah dalam algoritma back-propagation adalah sebagai berikut [SIN-04]:

1. Initialization. Pada tahap ini dilakukan inisialisasi untuk bobot awal $V_{nk}(0)$ dan $W_{km}(0)$ untuk semua *n*,*k*, dan *m*. Kemudian mengatur laju pelatihan (*learning rate*) μ dan faktor momentum (*momentum factor*) α dengan nilai positif yang sangat kecil. Nilai ambang error diatur sekecil mungkin.

2. Forward Pass. Pada tahap ini dilakukan pemilihan pasangan pola masukan $X_p = \{X_{p1},...,X_{pN}\}$ dan pola target $t_p = \{t_{p1},...,t_{pM}\}$ dari sekumpulan nilai yang akan dilatih. Setelah itu dilakukan proses perhitungan nilai keluaran pada lapisan interior $H_{pk}(i)$ dan nilai keluaran pada lapisan keluaran $O_{pm}(i)$ dengan:

$$H_{pk}(i) = f\left(\sum_{n=1}^{N} V_{nk}(i) X_{pn}\right) \qquad (4)$$

$$O_{pm}(i) = f\left(\sum_{n=1}^{N} W_{km}(i) H_{pk}(i)\right) \qquad (5)$$

Fungsi aktifasi untuk menghitung $H_{pk}(i)$ dan $O_{pm}(i)$ adalah fungsi sigmoid yang terdapat pada formula (1). Perhitungan square error $E(i)$ dilakukan dengan memban-dingkan antara target dan nilai keluaran pada lapisan keluaran untuk semua pola masukan, sehingga diformulasikan:

$$E(i) = \frac{1}{2}\sum_{p=1}^{P}\sum_{m=1}^{M}\left[t_{pm} - O_{pm}(i)\right]^2 \qquad (6)$$

Apabila nilai $E(i)$ ternyata lebih besar dari nilai ambang error maka dilakukan proses berikutnya yaitu Backward Pass, sebaliknya apabila nilai dari $E(i)$ tidak lebih besar dari nilai ambang error maka algoritma selesai dan konvergensi telah tercapai.

3. Backward Pass, pada tahap ini akan dilakukan perubahan bobot dengan cara mengupdate bobot untuk iterasi yang berikutnya sebesar $\Delta V_{nk}(i+1)$ dan $\Delta W_{km}(i+1)$, dimana Δ menyatakan perbedaan nilai yang sebelumnya dengan nilai yang akan datang pada iterasi berikutnya. Formula untuk update bobot ini dimulai dengan:

$$\delta_{pm}(i) = (t_{pm} - O_{pm}(i))O_{pm}(i)(1 - O_{pm}(i)) \qquad (7)$$

$$\overline{\delta_{pk}}(i) = H_{pk}(i)(1 - H_{pk}(i))\sum_{m=1}^{M}\delta_{pm}(i) W_{km}(i) \qquad (8)$$

$$\Delta W_{km}(i+1) = -\mu\frac{\partial E(i)}{\partial W_{km}(i)} + \alpha\Delta W_{km}(i)$$

$$= \mu\sum_{p=1}^{P}\delta_{pm}(i) H_{pk}(i) + \alpha\Delta W_{km}(i) \qquad (9)$$

$$\Delta V_{nk}(i+1) = -\mu\frac{\partial E(i)}{\partial V_{nk}(i)} + \alpha\Delta V_{nk}(i)$$

$$= \mu\sum_{p=1}^{P}\overline{\delta_{pk}}(i) X_{pn} + \alpha\Delta V_{nk}(i) \qquad (10)$$

Setelah dilakukan perhitungan untuk formula (9) dan (10) maka kemudian dilakukan perubahan bobot untuk iterasi berikutnya dengan:

$$W_{km}(i+1) = W_{km}(i) + \Delta W_{km}(i+1) \qquad (11)$$

$$V_{nk}(i+1) = V_{nk}(i) + \Delta V_{nk}(i+1) \qquad (12)$$

Setelah dilakukan perubahan bobot dengan (11) dan (12) kemudian *i* diubah menjadi $i = i+1$,

dan kembali ke langkah nomor 2 yaitu Forward Pass.

3. MAGNIFIED GRADIENT FUNCTION PROPAGATION (MGFPROP)

Magnified Gradient Function Propagation (MGFPROP) dilakukan pada bagian *backward pass* karena $\delta_{pm}(i)$ pada formula (7) dan $\overline{\overline{\delta}}_{pk}(i)$ pada formula (8) memuat faktor $O_{pm}(i)(1-O_{pm}(i))$ dan $H_{pk}(i)(1-H_{pk}(i))$ yang membuat laju konvergensi dari algoritma backpropagation menjadi sangat lambat. Nilai kedua sinyal $\delta_{pm}(i)$ dan $\overline{\overline{\delta}}_{pk}(i)$ akan menjadi sangat kecil apabila nilai keluaran dari lapisan interior $H_{pk}(i)$ dan nilai keluaran dari lapisan keluaran $O_{pm}(i)$ mendekati nilai 0 atau 1, sehingga error $(t_{pm}(i)-O_{pm}(i))$ tidak terlihat. Hal ini berakibat pada nilai dari $\Delta W_{km}(i+1)$ pada formula (9) dan $\Delta V_{nk}(i+1)$ pada formula (10) akan menjadi sangat kecil atau mendekati nilai $\Delta W_{km}(i)$, sehingga bobot pada iterasi berikutnya seolah-olah tidak mengalami perubahan. Karena bobot tidak ada perubahan, maka perhitungan error dari tahap *forward pass* akan selalu tetap sehingga menyebabkan *flat spot*.

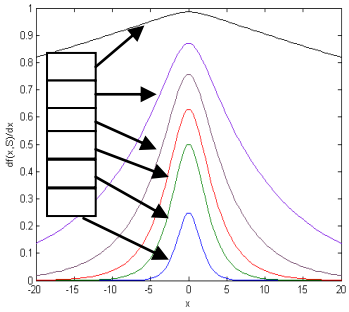
Untuk mengatasi *flat spot* dilakukan perbesaran $O_{pm}(i)(1-O_{pm}(i))$ dan $H_{pk}(i)(1-H_{pk}(i))$ dengan sebuah faktor penguatan 1/S, dimana S bernilai real positif dan lebih besar dari 1. Sehingga perumusan untuk penambahan faktor penguatan itu menjadi:

$$(O_{pm}(i)(1-O_{pm}(i)))^{1/S} \qquad (13)$$

$$(H_{pk}(i)(1-H_{pk}(i)))^{1/S} \qquad (14)$$

Dari (13) dan (14) tampak kedua formula ini adalah merupakan modifikasi dari formula (3) dengan penambahan faktor penguatan, sehingga tidak akan meng-hilangkan pengaruh dari sinyal error $(t_{pm}(i)-O_{pm}(i))$ meskipun $O_{pm}(i)$ mendekati nilai ekstrim 0 atau 1.

Pada gambar 2 tampak pengaruh dari nilai *S* pada faktor penguatan, untuk *S*= 1 sampai 100 pada turunan fungsi aktifasi.Dari perbandingan antara formula (3) dan hasil modifikasi dengan penguatan 1/S yang selanjutnya menjadi $f'(x,S)=[f(x)(1-f(x))]^{1/S}$ didapatkan:



Gambar 2. Pengaruh S pada Turunan Fungsi Aktifasi

$$\frac{f'(x,S)}{f'(x)}=[f(x)(1-f(x))]^{\frac{1}{S}-1}=(2+e^{x}+e^{-x})^{1-\frac{1}{S}} \qquad (15)$$

Dari formula (15) diatas 1 > 1- $1/S \geq 0$, ketika nilai dari $x=0$ maka

$$\frac{f'(x,S)}{f'(x)}=4^{1-\frac{1}{S}}<4 \qquad (16)$$

Dari formula (16) apabila didapati bahwa $|x|$ sangat besar maka

$$\frac{f'(x,S)}{f'(x)}=\left(2+e^{x}+e^{-x}\right)^{1-\frac{1}{S}}\approx e^{|x|(1-1/S)} \qquad (17)$$

Dari gambar 2 tampak pengaruh dari turunan fungsi aktifasi akan bertambah seiring dengan kenaikan nilai faktor penguatan. Apabila nilai dari x sangat besar, maka turunan fungsi aktifasi semakin besar sehingga pengaruh error $(t_{pm}(i)-O_{pm}(i))$ tidak hilang dan algoritma ini akan terhindar dari permasalahan *flat spot*.

4. DETERMINISTIC WEIGHT MODIFICATION (DWM)

Pendekatan pada algoritma ini adalah dengan penentuan bobot mana yang akan diubah dengan menggunakan *Deterministic Weight Modification* (DWM). Algoritma DWM bekerja pada 2 tahap yaitu:

1. DWM antara lapisan interior dan lapisan keluaran
2. DWM antara lapisan masukan dan lapisan interior

4.1. DWM antara lapisan keluaran dan lapisan interior

Tujuan yang ingin dicapai dalam mengubah bobot yang terletak diantara lapisan interior dan lapisan keluaran adalah untuk membuat nilai dari lapisan output semakin mendekati target yang diinginkan. Setelah bobot dimodifikasi diharapkan nilai output pada lapisan keluaran pada iterasi berikutnya akan semakin mendekati nilai output yang diharapkan.

Langkah pertama dari algoritma ini adalah dengan melakukan pemilihan neuron ke-*m* dari

lapisan keluaran pada iterasi ke-*i* yang diformulasikan:

$$E_{m^*} = \frac{1}{2}\sum_{p=1}^{P}[t_{pm^*} - o_{pm^*}(i)]^2 \geq \frac{E(i)}{M} \qquad (18)$$

Tujuan pencarian neuron yang ke *m** adalah untuk mengubah bobot $W_{km^*}(i+1)$ yang mempunyai koneksi dengan neuron *m**, sedangkan bobot lain tidak akan mengalami perubahan. Dengan bobot yang telah diubah nilai output pada iterasi berikutnya akan lebih dekat dengan output yang diharapkan, sehingga dapat mengatasi permasalahan *local minimum* dan mempercepat proses konvergensi.

Berikut adalah formula untuk output pada iterasi berikutnya $O_{pm^*}(i+1)$ yang diusahakan dapat mendekati nilai dari target t_{pm^*}:

$$O_{pm^*}(i+1) = \beta t_{pm^*} + (1-\beta)O_{pm^*}(i) \qquad (19)$$

untuk 0< β < 1 yang berlaku untuk semua pola input p.

Untuk mendapatkan nilai dari $\Delta W_{km^*}(i+1)$ dari output pada formula (19), maka:

$$O_{pm^*}(i+1) = f\left(\sum_{k=1}^{K} W_{km^*}(i+1)H_{pk}(i+1)\right)$$
$$= f(\sum_{k=1}^{K} W_{km^*}(i)H_{pk}(i+1) + \sum_{k=1}^{K}\Delta W_{km^*}(i+1)H_{pk}(i+1)) \qquad (20)$$

selanjutnya

$$\sum_{k=1}^{K}\Delta W_{km^*}(i+1)\,H_{pk}(i+1) = \varepsilon_{pm^*} \qquad (21)$$

$$\varepsilon_{pm^*} = f^{-1}(O_{pm^*}(i+1)) - \sum_{k=1}^{K} W_{km^*}(i)\,H_{pk}(i+1) \qquad (22)$$

Dimana:

$$f^{-1}(x) = \ln\frac{x}{1-x} \qquad (23)$$

Formula (21) dapat dibentuk matriks sebagai berikut:

$$\begin{bmatrix} H_{11}(i+1) & \dots & H_{1k}(i+1) & \dots & H_{1K}(i+1) \\ . & . & . & . & . \\ H_{p1}(i+1) & \dots & H_{pk}(i+1) & \dots & H_{PK}(i+1) \\ . & . & . & . & . \\ H_{P1}(i+1) & \dots & H_{Pk}(i+1) & \dots & H_{PK}(i+1) \end{bmatrix} \times \begin{bmatrix} \Delta W_{1m^*}(i+1) \\ . \\ \Delta W_{km^*}(i+1) \\ . \\ \Delta W_{Km^*}(i+1) \end{bmatrix} = \begin{bmatrix} \varepsilon_{1m^*} \\ . \\ \varepsilon_{pm^*} \\ . \\ \varepsilon_{Pm^*} \end{bmatrix} \qquad (24)$$

[1] menyederhanakan matrik pada formula (24) mengubah satu neuron *k** pada lapisan interior untuk memperoleh output yang diharapkan. Selanjutnya matriks itu akan menjadi:

$$\begin{bmatrix} H_{1k^*}(i+1) \\ . \\ H_{pk^*}(i+1) \\ . \\ H_{PK^*}(i+1) \end{bmatrix} \Delta W_{k^*m^*}(i+1) = \begin{bmatrix} \varepsilon_{1m^*} \\ . \\ \varepsilon_{pm^*} \\ . \\ \varepsilon_{Pm^*} \end{bmatrix} \qquad (25)$$

$$\begin{bmatrix} H_{1k^*}(i+1) \\ . \\ H_{pk^*}(i+1) \\ . \\ H_{PK^*}(i+1) \end{bmatrix} \Delta W_{k^*m^*}(i+1) = \begin{bmatrix} \varepsilon_{1m^*} - e_1 \\ . \\ \varepsilon_{pm^*} - e_p \\ . \\ \varepsilon_{Pm^*} - e_P \end{bmatrix} \qquad (26)$$

Selanjutnya $\Delta W_{k^*m^*}(i+1)$ dapat diperoleh dengan memperkecil nilai

$$e = \sum_{p=1}^{P} e_p^2 = \sum_{p=1}^{P}\left[\varepsilon_{pm^*} - H_{pk^*}(i+1)\Delta W_{k^*m^*}(i+1)\right]^2 \qquad (27)$$

Untuk mencari $\Delta W_{k^*m^*}(i+1)$ maka formula (28) dideferensialkan pada kedua ruas sehingga menghasilkan:

$$\frac{\partial e}{\partial \Delta W_{k^*m^*}(i+1)} = \sum_{p=1}^{P} 2\left[\varepsilon_{pm^*} - H_{pk^*}(i+1)\Delta W_{k^*m^*}(i+1)\right]\times \left(-H_{pk^*}(i+1)\right) \qquad (28)$$

Nilai dari $\Delta W_{k^*m^*}(i+1)$ dapat diperoleh dengan meminimalkan *e*, yang dapat dilakukan dengan membuat $\left(\partial e/\partial \Delta W_{k^*m^*}(i+1)\right) = 0$, atau dapat dituliskan:

$$\sum_{p=1}^{P}\left[\varepsilon_{pm^*}H_{pk^*}(i+1) - H_{pk^*}^2(i+1)\Delta W_{k^*m^*}(i+1)\right] = 0 \qquad (29)$$

Dari formula (30) diatas maka:

$$\Delta W_{k^*m^*}(i+1) = \frac{\sum_{p=1}^{P}\varepsilon_{pm^*}H_{pk^*}(i+1)}{\sum_{p=1}^{P}H_{pk^*}^2(i+1)} \qquad (30)$$

Pada setiap neuron *k* pada lapisan interior dilakukan perhitungan untuk menghitung $\Delta W_{k^*m^*}(i+1)$ dan *square error*.Perhitungan *square error* untuk setiap neuron *k* pada lapisan interior diformulasikan:

$$E_{m^*}^{(k)}(i+1) = \frac{1}{2}\sum_{p=1}^{P}\sum_{m=1}^{M}\left[t_{pm^*} - O_{pm^*}(i+1)\right]^2 \qquad (31)$$

Neuron *k** dari lapisan interior dipilih untuk mengupdate $\Delta W_{k^*m^*}(i+1)$ dengan syarat neuron *k** memiliki *square error* yang paling kecil diantara neuron *k* yang lain, yang dapat dituliskan $E_{m^*}^{(k^*)}(i+1) \leq E_{m^*}^{(k)}(i+1)$. Untuk neuron lain yang tidak terpilih, maka tidak ada perubahan bobot.

4.2. DWM antara lapisan interior dan lapisan input

Tujuan yang ingin dicapai dengan melakukan modifikasi terhadap bobot diantara lapisan masukan dan lapisan interior adalah mengeluarkan proses pelatihan dari *local minimum* dengan cara mengurangi error dari sistem. Dengan modifikasi bobot diantara lapisan masukan dan lapisan interior diharapkan pada iterasi yang selanjutnya error dari output akan mendekati error dari output yang telah ditentukan.

Perubahan error yang terjadi selama selang beberapa iterasi diformulasikan:

$$\Delta E = \left|E(i) - E(i-\tau)\right| / \tau \qquad (32)$$

Dimana τ menunjuk pada ukuran window. Untuk $\Delta E \le G_T$ dan $E(i) \ge E_T$, dimana G_T adalah nilai ambang gradien (gradient threshold) dan E_T adalah nilai ambang error (threshold error) maka proses pelatihan masuk dalam kondisi flat spot sehingga algoritma ini dapat digunakan. Selanjutnya pemilihan pola masukan yang sesuai dengan error output yang diinginkan diformulasikan:

$$E_{p^*}(i+1) = \frac{1}{2}\sum_{m=1}^{M}\left[t_{pm} - O_{pm}(i+1)\right]^2 \qquad (33)$$

Error output yang diinginkan dapat dipilih secara acak ataupun dengan pemilihan nilai dari square error yang terbesar ($E_{p^*}(i) \ge E_p(i)$).

Pemilihan output neuron (m*) pada lapisan keluaran dilakukan secara acak atau dengan $E_{p^*m^*}(i+1) \le E_{p^*m}(i+1)$, dimana:

$$E_{p^*m^*}(i+1) = \left|t_{p^*m} - O_{p^*m}(i+1)\right| \qquad (34)$$

Kemudian dilakukan pemilihan λ dengan batasan nilainya : 0 < λ < 1, yang akan digunakan untuk menghitung output pada layer keluaran.

$$O_{p^*m^*}(i+1) = \begin{cases} 1-\sqrt{2\lambda E(i)/PM} & \text{if } t_{p^*m^*} = 1 \\ \sqrt{2\lambda E(i)/PM} & \text{if } t_{p^*m^*} = 0 \end{cases} \qquad (35)$$

$$O_{p^*m^*}(i+1) = f\left(\sum_{k=1}^{K} W_{km^*}(i+1) H_{p^*k}(i+1)\right)$$
$$= f\left(\sum_{k=1}^{K} W_{km^*}(i+1) H_{p^*k}(i) + \sum_{k=1}^{K} W_{km^*}(i)\Delta H_{p^*k}(i+1)\right) \qquad (36)$$

Satu neuron k* pada lapisan interior yang akan dimodifikasi, pemilihannya berdasarkan pada nilai yang paling minimum yang diformulasikan:

$$\min\left(H_{p^*k^*}(i), 1-H_{p^*k^*}(i)\right) \le \min\left(H_{p^*k}(i), 1-H_{p^*k}(i)\right) \qquad (37)$$

Setelah k* terpilih maka dapat ditentukan $\Delta H_{p^*k}(i+1)$ dengan formula:

$$\Delta H_{p^*k^*}(i+1) = \frac{f^{-1}\left(O_{p^*m^*}(i+1)\right) - \left(\sum_{k=1}^{K} W_{km^*}(i+1) H_{p^*k}(i)\right)}{W_{k^*m^*}(i+1)} \qquad (38)$$

$$H_{p^*k^*}(i+1) = H_{p^*k^*}(i) + \Delta H_{p^*k^*}(i+1) \qquad (39)$$

$$H_{p^*k^*}(i+1) = f\left(\sum_{n=1}^{N} W_{nk^*}(i+1) x_{pn^*}\right)$$
$$= f\left(\sum_{n=1}^{N} W_{nk^*}(i) x_{pn^*} + \sum_{n=1}^{N} \Delta W_{nk^*}(i+1) x_{pn^*}\right) \qquad (40)$$

Formula (40) kemudian dapat diubah menjadi

$$f^{-1}\left(H_{p^*k^*}(i+1)\right) = f^{-1}\left(H_{p^*k^*}(i)\right) + \sum_{k=1}^{K} \Delta W_{nk^*}(i+1) x_{p^*n} \qquad (41)$$

$$\sum_{k=1}^{K} \Delta W_{nk^*}(i+1) x_{p^*n} = \overline{\varepsilon}_{p^*k^*} \qquad (42)$$

Formula (42) disubstitusikan ke dalam formula (41) menjadi

$$\overline{\varepsilon}_{p^*k^*} = f^{-1}\left(H_{p^*k^*}(i+1)\right) - f^{-1}\left(H_{p^*k^*}(i)\right) \qquad (43)$$

$$= \ln H_{p^*k^*}(i+1)\left(1-H_{p^*k^*}(i)\right) / H_{p^*k^*}(i)\left(1-H_{p^*k^*}(i+1)\right) \qquad (44)$$

Ruas kanan formula (44) dikalikan dengan $\sum_{n=1}^{N} x_{p^*n}^2 \Big/ \sum_{n=1}^{N} x_{p^*n}^2$ maka akan menjadi

$$\sum_{k=1}^{K} \Delta W_{nk^*}(i+1) x_{p^*n} = \overline{\varepsilon}_{p^*k^*} \frac{\sum_{n=1}^{N} x_{p^*n}^2}{\sum_{n=1}^{N} x_{p^*n}^2} \qquad (45)$$

Sehingga

$$\Delta W_{nk^*}(i+1) = \frac{\overline{\varepsilon}_{p^*k^*}\, x_{p^*n}}{\sum_{n=1}^{N} x_{p^*n}^2} \qquad (46)$$

5. ALGORITMA GENETIKA

Algoritma genetika merupakan teknik pencarian (*searching*) pada suatu ruang pencarian solusi dengan menggunakan perhitungan evolusioner dan metode non-deterministik (operator crossover dan mutasi).

Dalam algoritmaGA-MDPROP bertujuan untuk menggantikan pencarian inisialisasi bobot secara acak, sehingga kromosom akan merepresen-tasikan bobot yang terletak antara lapisan masukan-lapisan interior dan lapisan interior-lapisan keluaran dan nilai fitness.

5.1. Seleksi

Proses seleksi bertujuan untuk melakukan penyalinan kromosom agar masuk dalam *mating pool*. Peluang agar kromosom terpilih bergantung pada nilai fitnessnya, semakin besar nilai fitness maka semakin besar kemungkinan akan terpilih (*survival*). Fungsi fitness dinyatakan dengan :

$$fitness = \frac{1.001}{err + 0.001} \qquad (47)$$

Dimana err menyatakan *square error* yang terdapat pada formula (6)

Proses seleksi yang secara umum dipakai adalah dengan menggunakan *Roullete Wheel Selection* yang disusun dalam langkah-langkah sebagai berikut:

Menghitung nilai fitness eval(V_k) untuk setiap kromosom V_k

$$eval(V_k) = f(x) \qquad (48)$$

$f(x)$ yang digunakan mengacu pada formula (48)

Menghitung jumlah keseluruhan fitness yang ada dalam satu generasi

$$F = \sum_{k=1}^{k=pop_size} eval(V_k) \qquad (49)$$

Menghitung probabilitas seleksi P_k untuk setiap kromosom V_k

$$P_k = \frac{eval(V_k)}{F} \qquad (50)$$

Menghitung probabilitas kumulatif q_k untuk setiap kromosom V_k

$$q_k = \sum_{j=1}^{k} P_j \qquad (51)$$

Prosedur penyalinan kromosom dimulai dengan:

Membangkitkan suatu nilai secara acak (r) yang terletak diantara 0 dan 1 sebanyak jumlah kromosom dalam satu generasi.

Bila $q_1 \geq r$, maka kromosom pertama V_1 yang akan terpilih, kalau tidak maka chromsome ke–*k* atau V_k akan terpilih($2 \leq k \leq pop_size$) sehingga $q_{k-1} < r \leq q_k$.

5.2. Crossover

Proses *crossover* terjadi ketika kedua kromosom *parent* yang terpilih bertukar bagian sehingga terbentuk dua kromosom *offspring*. Prosedur untuk memilih kromosom untuk dijadikan *parent*:

1. Penentuan probabilitas *crossover Pc*.
2. Membangkitkan bilangan acak antara 0 dan 1 sebanyak *i* (jumlah kromosom dalam satu generasi)
3. Membandingkan setiap bilangan acak pada langkah 2 dengan probabilitas *crossover Pc*.
4. Kromosom akan terpilih menjadi parent bila random yang ke-*i* kurang atau sama dengan probabilitas *crossover Pc*.
5. Proses pemilihan ini akan diulang apabila kromosom yang terpilih hanya satu.

Prosedur proses *crossover* dimulai dengan membangkitkan bilangan acak antara 0 dan 1, kemudian dilakukan proses crossover :

$$offspring1 = (parent1 \times acak) + (parent2 \times (1 - acak)) \qquad (52)$$

$$offspring2 = (parent1 \times (1 - acak)) + (parent2 \times acak) \qquad (53)$$

5.3. Mutasi

Proses mutasi adalah proses untuk mengubah salah satu atau lebih bagian dari kromosom. Pada proses mutasi, apabila gen x_k dari parent terpilih untuk dimutasi maka kromosom baru yang terbentuk adalah $x' = [x1,....x_k',.....xn]$, dimana x_k' secara acak terpilih dari 2 pilihan:

$$x_k' = x_k + \Delta(t, x_k^u - x_k) \qquad (54)$$

$$x_k' = x_k - \Delta(t, x_k - x_k^L) \qquad (55)$$

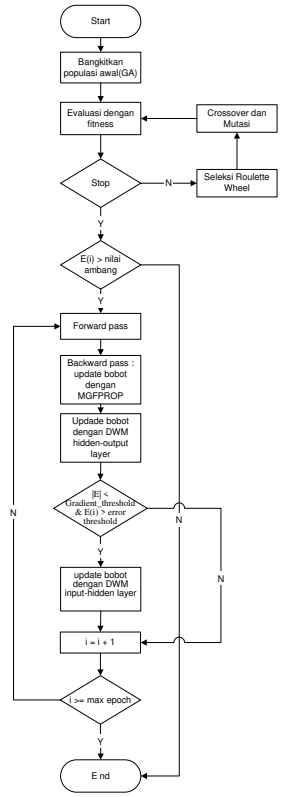
Formula (55) untuk nilai acak=1 sedangkan untuk formula (56) untuk nilai acak=0.

Fungsi Δ(t,y) diberikan oleh perumusan:

$$\Delta(t, y) = y \times r \times \left(1 - {}^{t}\!/\!_{T}\right)^{b} \qquad (56)$$

dimana :
r adalah nilai acak antara 0 dan 1
T adalah jumlah maksimal generasi
b adalah nilai parameter
t adalah generasi yang sekarang.

Berikut adalah flowchart hibrid GA-MDPROP



Gambar 3. Flowchart GA-MDPROP

6. HASIL SIMULASI

Algoritma GA-MDPROP diterapkan pada empat masalah yang dikemukakan [1] yaitu:

Tabel 1: Deskripsi Permasalahan

Permasalahan	Keterangan	Arsitektur Jaringan N – K - M	μ	α	β	Γ
XOR	Diberikan 2 input biner a dan b, dan dengan output a⊕b.	2 - 2 - 1	0,5	0,7	0,8	0,2
3-bit parity	Terdiri dari 3 input dan 1 output dengan 8 pola yang berlainan	3 – 3 - 1	0,6	0,9	0,5	0,2
5-bit counting	Bertujuan menghitung jumlah angka "1" yang tersebar pada 5 input unit.	5 – 12 – 6	0,1	0,7	0,5	0,2
Character Recognition	Bertujuan untuk mengenal input dari ukuran matrtriks 8*8 dan output berupa 26 huruf kecil (a,…,z)	64 – 20 - 26	0,01	0,7	0,5	0,2

Tabel 2 : Hasil Percobaan dengan 3 Metode Pelatihan

Metode	**Permasalahan**	**S=1**	**S=2**	**S=3**	**S=4**	**S=5**	**S=6**	**S=7**	**S=8**	**S=9**	**S=10**
BP	XOR	**1918**	**479**	405	333	268	295	242	244	234	222
MDPROP		174	89	58	55	62	59	58	55	56,6	60
GA-MDPROP		127	71	59	57	55	56	54	56	57	58
BP	3 bit parity	1025	454	345	404	319	318	276	289	319	406
MDPROP		132	65	67	63	60	64	70	66	69	65
GA-MDPROP		102	43	49	53	54	59	60	59	61	63
BP	5bit Counting	Fail	1733	542	427	731	367	378	360	321	361
MDPROP		6538	707	435	357	303	443	365	279	268	387
GA-MDPROP		2753	783	385	312	293	287	296	274	269	329
BP	Character Recognition	16359	5262	3749	1730	944	641	459	382	393	339
MDPROP		Fail	Fail	854	702	432	322	Fail	305	Fail	Fail
GA-MDPROP		1404	694	463	557	297	362	305	562	311	Fail

Hibrid GA-MDPROP diujicobakan pada 4 permasalahan yaitu : XOR, 3-bit parity, 5 bit counting, dan character recognition. Tabel 1 adalah deskripsi untuk keempat permasalahan yang ada beserta arsitektur jaringan dan nilai dari parameter-parameter yang digunakan.

Masing-masing permasalahan dilakukan dengan maksimum iterasi 30000, apabila dalam iterasi ke-30000 tidak dapat mencapai threshold=0.001 maka dinyatakan fail. Untuk ukuran window, nilai ambang error(E_T), nilai ambang gradient (G_T) berturut-turut 10, 0.005, 0.0001 yang berguna untuk mendeteksi local optima. Maksimum iterasi untuk algoritma genetika diatur = 100, probabilitas crossover P_c diatur =0.3 dan untuk probabilitas mutasi P_m diatur =0.05. Inisialisasi bobot pertama kali dibuat random dengan nilai diantara –1 dan 1. Percobaan ini dilakukan processor AMD Athlon 1,2Ghertz dan memory SDRAM 258Mb dengan menggunakan program MATLAB.

Untuk MGFPROP, MDPROP, dan GA-MDPROP diujicoba dengan menggunakan S yang berbeda-beda yaitu mulai dari 1sampai10, sehingga dapat diketahui seberapa besar pengaruh S terhadap proses pelatihan.

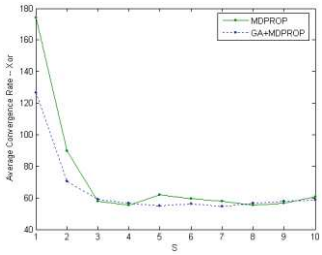
Pada tabel 2 tampak angka yang ditebali adalah peningkatan yang dilakukan dengan menggunakan hibrid GA-MDPROP. Secara umum untuk S=1 hampir semua permasalahan dapat diselesaikan dengan lebih baik dengan menggunakan hibrid GA-MDPROP. Perbedaan nilai-nilai yang paling tajam untuk setiap permasalahan terletak antara S=1 dan S=2. Untuk permasalahan XOR kestabilan proses pelatihan tercapai disekitar iterasi ke-55, ini berlaku untuk MDPROP dan hibrid GA-MDPROP. Untuk permasalahan 3bit Parity, kestabilan proses pelatihan tercapai disekitar iterasi ke-60, ini juga berlaku untuk MDPROP dan GA-MDPROP. Untuk permasalahan 5bit Counting, kestabilan proses pelatihan tercapai disekitar iterasi ke-300 berlaku untuk MDPROP dan GA-MDPROP. Untuk permasalahan character recognition, kestabilan proses pelatihan tercapai pada sekitar iterasi ke –300.

Untuk gambar 4 sampai 6, membandingkan algoritma MDPROP dengan GA-MDPROP dengan nilai S yang berbeda-beda. Untuk gambar 7, digunakan untuk membandingkan GA-MDPROP dengan MGFPROP karena MDPROP pada permasalahan ini banyak yang mengalami fail.

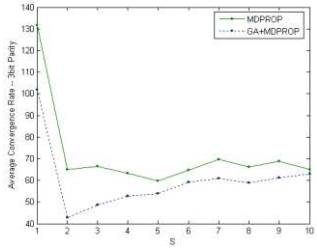
Pada gambar 4 untuk S=1 kecepatan proses pelatihan dari GA-MDPROP lebih cepat dibandingkan dengan MDPROP sehingga dapat mengurangi proses learning sebesar 27%, sedangkan untuk S=3 sampai S=10 disana terlihat bahwa kestabilan proses pelatihan dicapai bersamaan dengan

MDPROP.

Pada gambar 5 faktor penguatan S=1 hibrid GA-MDPROP lebih baik dari MDPROP dan dapat mereduksi sekitar 22,9% proses learning dari MDPROP, sedangkan untuk nilai S=2 membuat penurunan yang tajam dari average convergence rate yang membuat gabungan GA-MDPROP dapat mereduksi proses learning sebesar 33,8% dari MDPROP.



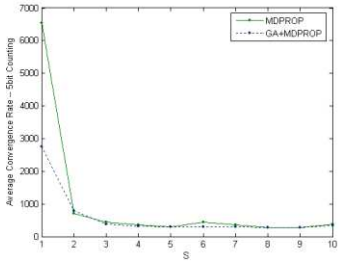
Gambar 4. Xor dengan MDPROP dan GA-MDPROP



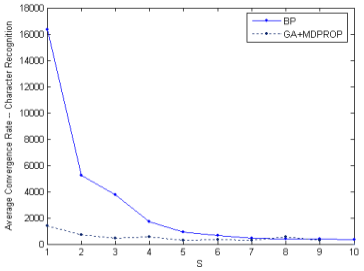
Gambar 5. 3Bit Parity dengan MDPROP dan GA-MDPROP

Pada gambar 6 untuk hibrid GA-MDPROP dapat menyelesaikan dengan proses pelatihan pada iterasi ke-2753 sehingga hibrid GA-MDPROP dapat mengurangi proses learning sebesar 53,8% dari MDPROP. Bila dilihat dari grafik ketiga metode tersebut berkecenderungan untuk menempati daerah stabil disekitar 300 iterasi. Penurunan nilai rata-rata laju konvergensi paling tajam pada S=2. Untuk Hibrid GA-MDPROP yang paling terlihat menonjol adalah pada waktu S=1 algoritma ini dapat menyelesaikan permasalahan dengan lebih baik.

Pada gambar 7 terlihat perbandingan untuk MGFPROP dan GA-MDPROP, untuk S=1 sampai S=6 terlihat perbedaan yang cukup besar. Untuk nilai S yang lain terlihat kecenderungan masuk dalam daerah yang sama.



Gambar 6. 5Bit Counting dengan MDPROP dan GA-MDPROP



Gambar 7. Character Recognition dengan MGFPROP dan GA-MDPROP

7. KESIMPULAN

Dari penelitian ini dapat disimpulkan bahwa:

1. Peningkatan faktor penguatan S sangat berpengaruh pada peningkatan laju konvergensi pada permasalahan yang ada.
2. Dari beberapa nilai faktor penguatan yang telah diujicoba yaitu mulai dari 1 sampai 10, untuk S=2 dapat memberikan peningkatan laju konvergensi yang cukup besar.
3. Pada nilai S tertentu, GA-MDPROP dan MDPROP akan berkecenderungan memiliki rata-rata laju konvergensi yang sama.
4. Hibrid GA-MDPROP mempunyai kecen-derungan bersama-sama dengan MDPROP untuk menuju daerah kestabilan. Untuk permasalahan XOR kestabilan proses pelatihan tercapai disekitar iterasi ke-55, sedangkan untuk permasalahan 3bit parity kestabilan proses pelatihan tercapai disekitar iterasi ke-60, untuk permasalahan 5bit counting, kestabilan proses pelatihan tercapai disekitar iterasi ke-300, untuk permasalahan character recognition, kestabilan proses pelatihan tercapai pada sekitar iterasi ke–300.
5. Hibrid GA-MDPROP memberikan pening-katan laju konvergensi yang cukup besar daripada MDPROP pada saat faktor penguatan S=1.

8. DAFTAR PUSTAKA

1. Sin Chun Ng, Chi Chung Cheung and Shu Hung Leung., Magnified Gradient Function With Deterministic Weight Modification in Adaptive Learning , 6 November 2004.
2. Sin Chun Ng, Chi Chung Cheung, Shu Hung Leung, A.Luk., Fast Convergence for Back-Propagation Network with Magnified Gradient Function, 2003.
3. Sin Chun Ng, Chi Chung Cheung, and Shu Hung Leung., Deterministic Weight Modification Algorithm for Efficient Learning, 2003.
4. Dan W Patterson., Artificial Neural Networks Theory and Application, 1996.
5. Mitsuo Gen, Runwei Cheung., Genetic Algorithms and Engineering Design, 1997.
6. Zbigniew Michalewicz., Genetic Algorithms + Data Structures = Evolution Programs, 1995.
7. Hendry Setiawan., Studi Penerapan Algoritma Genetika pada Kendali Logika Fuzzy, 2000.