# Algoritma Optimasi Chaos pada Ridge Polynomial Neural Network untuk Estimasi Biaya Software

Riah Ukur Ginting
Program Studi Matematika FMIPA USU
Jl. Bioteknologi I Kampus USU Pd Bulan Medan 20155
Email: riahukur81@gmail.com

***Abstract*—Ridge polynomial neural network (RPNN) originally proposed by Shin and Ghosh, constructed from an increase in the number of order pi-sigma neurons (PSN). RPNN maintains faster learning, a stonger mapping of single layer of higher order neural network (HONN) and avoids heavy loads due to the increasing number of inputs. Chaos optimization algorithm is used by utilizing the logistic equation which is sensitive to initial conditions, so that the movement of chaos can be changed in any circumstances in partcular scale within regularity, ergodic and maintain the diversity of solutions.**

**Software cost of prediction is a very important process in software development. The ignorance of cost prediction of the software will cause in poor quality of software produced as well as creates a new problem in the software financial system. To anticipate the occurrence of errors in software cost of estimation, it is necessary to develop a calculating method or prediction cost of software so that software cost which predicted be more optimal. This research will develop a way by using the Chaos Optimization Algorithm in Ridge Polynomial Neural Network to predict the cost of software.**

**Networking process training is using RPNN while of the initial value loads the searching and networking biases are using chaos optimization algorithm. The structure used consists of six (6) layer neurons and one (1) neuron layer output. This research uses data from NASA project undertaken between 1980 untill 1990. The results of this research demonstrates that the proposed algorithm can be used for prediction.**

***Keywords—Prediction software cost; pi-sigma; artificial neural network; chaos optimization algorithm; RPNN***

## I. PENDAHULUAN

Estimasi biaya software adalah proses yang sangat penting dalam pengembangan software. Estimasi biaya software sangat penting untuk mengontrol dan mengatur efisiensi pada seluruh proses yang dilakukan dalam pengembangan software. Estimasi biaya software sangat dibutuhkan dalam membuat usulan proposal, negosiasi kontrak, penjadwalan, kontrol dan monitoring. Untuk itu akurasi biaya software sangat dibutuhkan karena (Leuang & Fan, 2001):

a. Dapat membantu untuk mengklasifikasi dan menentukan prioritas pengembangan proyek sehubungan dengan perencanaan bisnis secara keseluruhan.
b. Dapat digunakan untuk menentukan *resources* yang sesuai dengan proyek yang dikerjakan dan seberapa baik *resources* ini akan digunakan.
c. Dapat digunakan untuk menilai pengaruh terhadap perubahan dan dukungan dalam perencanaan ulang.
d. Dapat memudahkan untuk mengatur dan mengontrol proyek ketika *resources* benar-benar sesuai dengan yang dibutuhkan.
e. Dapat memenuhi harapan c*ustomer* agar biaya pengembangan yang aktual dapat sejalan dengan biaya pengembangan yang di estimasi.

Umumnya, estimasi *effort* dan biaya sangat sulit dilakukan dalam proyek software karena proyek software bersifat dinamis dan kesulitan menemukan proyek yang sangat mirip dengan proyek tersebut sebelumnya. Selain itu, sulit untuk menentukan metode estimasi yang cocok untuk proyek tersebut.

Berdasarkan latar belakang tersebut di atas, penelitian ini akan mengkaji kemampuan Estimasi biaya software dengan menggunakan *ridge polynomial neural network* (RPNN).

## II. TINJAUAN PUSTAKA

Penelitian [1] melakukan penelitian dengan cara penggabungan yaitu pencarian lokal yang disebut pencarian chaos dengan multi-objective memetic algorithm (MOMA). Hasil yang diperoleh dibandingkan dengan multi-objective genetic local search (MOGLS) yaitu MOMA yang sangat efisien, menunjukkan proses pengukuran konvergensi MOCMA lebih baik daripada MOGLS, dan juga pengukuran penyebaran MOCMA lebih merata daripada MOGLS. Pada penelitian [2] membahas tentang approksimasi polynomial multivariate order tinggi dan memperkirakan banyaknya akar dari polynomial univariate order tinggi. Hasilnya dapat meningkat proses pembelajaran pada RPNN.

Pada penelitian [4] menerapkan polynomial neural network (PNN) yaitu jaringan pi-sigma dan jaringan ridge polynomial untuk mengontrol perasangan dari sistem pembangkit listrik. Hasilnya menunjukkan algoritma pembelajaran RPNN bekerja dengan baik. Pada penelitian [5] melakukan penginisialisasian bobot Multilayer Neural Networks dengan menggunakan Chaos Optimization Algorithm dan algoritma pembelajaran dengan backpropagation. Pada penelitian [7] menunjukkan bahwa modifikasi metode analogy menghasilkan estimasi biaya proyek pengembangan perangkat lunak yang akurat dibandingkan pada hasil dan metode anology sheppered standard. Dan Pada Penelitian [8] melakukan penggabungan

antara variabel metrik algoritma optimasi chaos dan jaringan syaraf tiruan dengan algoritma backpropagation digunakan untuk pengenalan pola datar. Model diterapkan untuk mengatasi beberapa kelemahan jaringan syaraf bacpropagation biasa, yaitu: konvergensi lambat, akurasi rendah dan kesulitan dalam menemukan optimum global.

## III. METODE PENELITIAN

### 3.1 Chaos

Menurut Gulick (1992) memiliki definisi berikut dari chaos

***Definisi 3.1*** Sebuah fungsi $f$ adalah chaotic jika memenuhi setidaknya satu dari kondisi berikut:

i. $f$ memiliki *Lyapunov eksponen* positif pada setiap titik dalam domainnya yang bukan akhir periodik. atau
ii. $f$ memiliki *bergantung sensitif pada kondisi awal* pada domainnya.

Dimana $fo$ adalah fungsi objektif, dan $x$ adalah keputusan vektor berisikan $n$ variabel.

### 3.2 Algoritma optimisasi chaos

Masalah fungsi optimasi menurut Khoa dan Nakagawa (2007) dapat dirumuskan sebagai berikut :

$$fo = \min f(X), \qquad X = [x_1, x_2, \dots, x_n] \quad (1)$$
$$x_i \in [a_i, b_i], \qquad i = 1,2, \dots, n$$

Dimana $fo$ adalah fungsi objektif, dan $x$ adalah keputusan vektor berisikan $n$ variabel.

**Pencarian pertama**

**Langkah 1** : Inisialisasi

$r$ = 0, $r$ adalah variabel tanda untuk iterasi variabel chaos. Tetapkan jumlah iterasi maksimum $r$ *max*. Mengisi variabel chaos awal dengan random, $A_i^0 \in (0,1)$, dimana $i$ = 1, 2, ..., $n$ adalah jumlah bobot diantara layer input dan layer hidden ditambah bobot bias.

**Langkah 2 :** Memetakan variabel chaos $A_i^r$ ke dalam rentang variabel optimisasi $X_i^r \in [a^1, b^1]$ dengan persamaan:

$$X_i^r = a^1 + A_i^r(b^1 - a^1) \quad (2)$$

Dimana :

$X_i^r$ adalah variabel optimisasi
$A_i^r$ adalah variabel chaos
$r$ adalah variabel tanda untuk iterasi variabel chaos pada pencarian pertama
$i$ = 1, 2, ..., $n$ adalah banyaknya jumlah variabel
$a^1$ adalah interval paling bawah pada pencarian pertama
$b^1$ adalah interval paling atas pada pencarian pertama

**Langkah 3 :** Menghitung fungsi objektif $F(X^r)$ dengan persamaan MSE

Jika $F(X^r) < F^*$ maka $A^* = A^r$, $X^* = X^r$ dan $F^* = F(X^r)$.

Jika $F(X^r) \geq F^*$ maka tinggalkan $X^r$

**Langkah 4 :** Menghitung variabel chaos dengan persamaan

$$A_i^{r+1} = \mu A_i^r (1 - A_i^r) \quad (3)$$

dan $r = r + 1$

Dimana :

$A_i^{r+1}$ adalah variabel chaos iterasi ke-$r+1$
$A_i^r$ adalah variabel chaos iterasi ke-$r$
$r$ adalah variabel tanda untuk iterasi variabel chaos pada pencarian pertama
$i$ = 1, 2, ..., $n$ adalah banyaknya jumlah variabel

**Langkah 5 :** Jika $r$ belum mencapai $r\,max$, maka ulangi langkah 2 – 4. Jika sudah, maka lanjut ke langkah 6.

**Pencarian kedua**

$h$ adalah variabel tanda untuk iterasi variabel chaos. Tetapkan jumlah iterasi maksimum $h$ *max*. Pada penelitian ini menetapkan interval untuk $a^2 = -10$ dan $b^2 = 10$.

**Langkah 6 :** Memetakan variabel chaos $A_i^h$ ke dalam rentang variabel optimisasi $X_i^h \in [a^2, b^2]$ dengan persamaan

$$X_i^h = X_i^* + \omega\,(2 * A_i^h - 1), i = 1, 2, \dots, n \quad (4)$$

Dan $\omega = \rho(b^2 - a^2)$, $\rho \in (0,\ 0.5)$

Dimana :

$X_i^h$ adalah variabel optimisasi
$A_i^h$ adalah variabel chaos
$h$ adalah variabel tanda untuk iterasi variabel chaos pada pencarian kedua
$i$ = 1, 2, ..., $n$ adalah banyaknya jumlah variabel
$a^2$ adalah interval paling bawah pada pencarian kedua
$b^2$ adalah interval paling atas pada pencarian kedua
$\rho$ adalah faktor ruang penyempit pada pencarian kedua, pada penelitian ini menetapkan $\rho$ = 0,4

**Langkah 7 :** Men[illegible]ungsi objektif $F(X^h)$ dengan persamaan MSE (3.3)

Jika $F(X^h) < F^*$ maka $A^* = A^h$, $X^* = X^h$ dan $F^* = F(X^h)$.

Jika $F(X^h) \geq F^*$ maka tinggalkan $X^h$

**Langkah 8 :** Menghitung variabel chaos dengan persamaan

$$A_i^{h+1} = \mu A_i^h (1 - A_i^h) \quad (5)$$

dan $h = h + 1$

Dimana :

$A_i^{h+1}$ adalah variabel chaos iterasi ke-$h+1$
$A_i^h$ adalah variabel chaos iterasi ke-$h$
$h$ adalah variabel tanda untuk iterasi variabel chaos pada pencarian kedua
$i$ = 1, 2, ..., $n$ adalah banyaknya jumlah variabel

**Langkah 9 :** Jika $h$ belum mencapai $h\,max$, maka ulangi langkah 6 – 8 Jika sudah, maka COA diakhiri dan $X^*$ adalah solusinya.

### 3.3 Jaringan Ridge Polynomial Neural Network (RPNN)

Jaringan ridge polynomial adalah generalisasi dari jaringan pi-sigma yang menggunakan bentuk khusus dari ridge polynomial [4].

Untuk $x = [x_1 \dots x_n]^T$ dan $w = [w_1 \dots w_n]^T \in R^n$,

diberikan $(x, w) = \sum_{i=1}^{n} w_i x_i \quad (6)$

(x,w) didefinisikan sebagai *inner product* antara dua definisi vector.

***Definisi 3.2*** *Diberikan himpunan compact* $K \subset R^d$, *semua fungsi disefenisikan pa* (3.5) *an bentuk,* $f((.,w)) : K \to R$, *dimana* $w \subset R^d$ *dan* $f(.) : R \to R$ *adalah kontinu, disebut*

*fungsi ridge*. Ridge polynomial adalah fungsi ridge yang dapat digambarkan sebagai : $\sum_{i=0}^{N}\sum_{j=0}^{M} a_{ij}\,(\boldsymbol{x}, \boldsymbol{w}_{ij})^i$ (7)
untuk beberapa $a_{ij} \in \mathbb{R}$ dan $\langle \boldsymbol{x}, \boldsymbol{w}_{ij} \rangle \in \mathbb{R}^d$.
Sebuah teorema ditampilkan menyatakan setiap polynomial multivariate dapat digambarkan dalam istilah ridge polynomial, dan dapat direalisasikan dengan jaringan ridge polynomial yang disesuaikan (Teorema 3.1)

***Teorema 3.1*** *Setiap polynomial multivariate dapat digambarkan sebagai ridge polynomial.*

$$p(x) = \sum_{j=0}^{k}\sum_{m=1}^{n_j} c_{jm}\, x^{ijm} \;\Leftrightarrow\; p(x) = \sum_{j=i}^{N}\prod_{i=1}^{j}\left(\langle x, w_{ji}\rangle + w_{ji}\right) \quad (8)$$

Jaringan ridge polynomial memiliki kemampuan pemetaan yang bagus didalam artinya bahwa setiap fungsi kontinu pada himpunan *compact* di $\mathbb{R}^d$ dapat diaproksimasi secara merata dengan jaringan (Teorema 3.2).

***Teorema 3.2*** *Setiap fungsi kontinu pada himpunan compact di* $\mathbb{R}^d$ *dapat diaproksimasi secara merata dengan jaringan ridge polynomial.*

Jadi, sebuah fungsi yang tidak diketahui $f$ didefinisikan pada himpunan compact $k \subset \mathbb{R}^d$ dapat diaproksimasi dengan jaringan ridge polynomial seperti berikut :

$$f(x) \simeq (\langle x, w_{11}\rangle + w_{11}) + (\langle x, w_{21}\rangle + w_{21}) \cdot (\langle x, w_{22}\rangle + w_{22}) + \cdots + (\langle x, w_{N1}\rangle + w_{N1}) \cdots (\langle x, w_{NN}\rangle + w_{NN}) \quad (9)$$

dimana setiap perkalian diperoleh sebagai output dari jaringan pi-sigma dengan unit output.

3.4 Persamaan output jaringan ridge polynomial

Persamaan output jaringan ridge polynomial yang mengacu dari (Gupta dkk, 2003) dapat dijelaskan sebagai berikut :

$$y = \emptyset\left(\sum_{j=1}^{N}\prod_{i=1}^{j}\left(\langle x, w_{ij}\rangle + \theta_{ji}\right)\right) \text{ atau } y = f\left(\sum_{j=1}^{N}\prod_{i=1}^{j}\left(\sum_{k=1}^{n} w_{ijk}x_k + \theta_{ji}\right)\right) \quad (10)$$

Dimana :
$j$ adalah jumlah PSN dari 1 sampai $N$,
$i$ adalah banyaknya order di PSN dari $i$ sampai $j$,
$k$ adalah jumlah input dari 1 sampai $n$,
$w_{ijk}$ adalah bobot yang diperbaharui dari input $x_k$ ke PSN order
ke-$i$ dariPSN ke-$j$.
$f(x)$ adalah fungsi aktivasi nonlinear.

Dimisalkan pada contoh jaringan ridge polynomial dengan order $N$ = 2, jumlah input $n$ = 5

$$y = \sum_{j=1}^{2}\prod_{i=1}^{j}\left(\sum_{k=1}^{5} w_{ijk}x_k + \theta_{ji}\right) \quad (11)$$

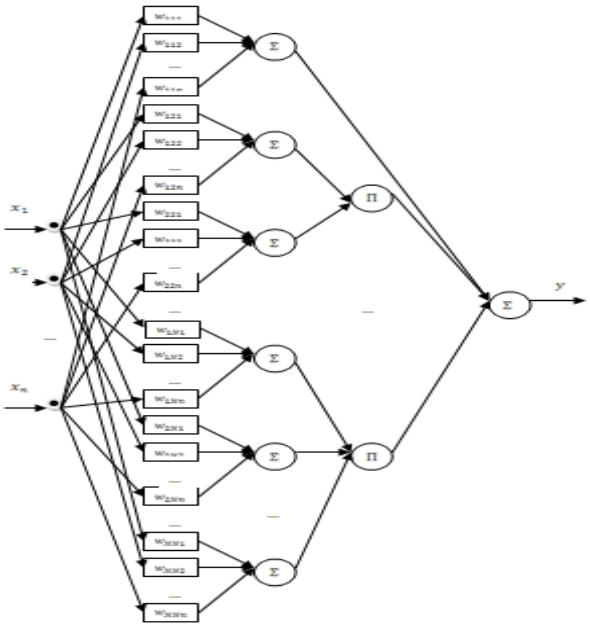
Dan total jumlah bobot adalah $N(N+1)(n+1)/2$ = 2(2+1)(5+1)/2 = 18.

Perbandingan jumlah bobot dari tiga jenis struktur jaringan polynomial diberikan dalam Tabel 1 mengacu dari (Gupta dkk, 2003). Hasil menunjukkan bahwa ketika jaringan mempunyai tingkat order tinggi yang sama, bobot RPNN secara signifikan kurang dari HONN. Secara khusus, ini adalah sangat menarik dalam perbaikan yang ditawarkan oleh RPNN.

TABLE I. JUMLAH BOBOT DI DALAM JARINGAN POLYNOMIAL

| Order of Network | Number of Weights | | | | | |
|---|---|---|---|---|---|---|
| | Pi–sigma | | RPNN | | HONN | |
| $N$ | $n=5$ | $n=10$ | $n=5$ | $n=10$ | $n=5$ | $n=10$ |
| 2 | 12 | 22 | 18 | 33 | 21 | 66 |
| 3 | 18 | 33 | 36 | 66 | 56 | 286 |
| 4 | 24 | 44 | 60 | 110 | 126 | 90 |

Arsitektur RPNN yang menggunakan PSN sebagai blok dasar dapat dilihat pada Gambar 1.



Gambar 1. Jaringan RPNN

3.5 Algoritma pelatihan jaringan ridge polynomial

Algoritma pelatihan jaringan ridge polynomial pada dasarnya terdiri dari 5 tahapan (Ghazali dkk. 2008) yaitu:

1. Mulai dengan RPNN order 1, yang mana memilki satu unit PSN order pertama.
2. Melakukan pelatihan dan update bobot secara asinkronus setelah setiap pola pelatihan.
3. Ketika error dari PSN yang diamati berubah, dibawah error standart $r$, yaitu $\left|\frac{e_c - e_p}{e_p}\right| < r$ , maka PSN order lebih tinggi ditambahkan. Catatan bahwa $e_c$ adalah MSE untuk iterasi saat ini dan $e_p$ adalah MSE untuk iterasi sebelumnya.
4. Error target $r$ dan learning rate $n$ dibagi dengan faktor penurun.
5. Jaringan diperbaharui melakukan siklus pembelajaran (ulangi langkah 2 sampai 4) sampai jumlah unit PSN yang diinginkan tercapai atau jumlah iterasi maksimum tercapai.

Pelatihan jaringan ridge polynomial :

**Langkah 1 :** Menambah jaringan pi-sigma. Jika jumlah neuron jaringan pi-sigma belum terpenuhi sampai yang diinginkan, lakukan langkah 2 – 10.
**Langkah 2** : Inisialisasi bobot dan bias. Bobot dan bias diperoleh dari perhitungan algoritma optimasi chaos yang sebelumnya sudah dihitung.
**Langkah 3** : Jika kondisi STOP yaitu belum mencapai eror yang diinginkan atau maksimum iterasi terpenuhi, lakukan langkah 4 – 9.
**Langkah 4** : Untuk setiap pola data, lakukan langkah 5 – 8.
**Langkah 5** : Menerima sinyal input dari pola data
**Langkah 6** : Menghitung output jaringan pi-sigma dengan persamaan (11)

$$h_j = \sum_k w_{kj} x_k + \Theta_j \quad \text{dan} \quad y_{PSN} = f\left(\prod_j h_j\right) \qquad (12)$$

**Langkah 7 :** Menghitung delta bobot jaringan pi-sigma dengan persamaan berikut :

$$\delta_l = \eta(d-y)y' \prod_{\substack{z=1 \\ z \neq l}}^{j} h_z \quad \Delta w_{kl} = \delta_l x_k \text{ dan } \Delta w_{0l} = \delta_l \qquad (13)$$

**Langkah 8** : Memperbarui bobot dan bias $w_{kl}$(baru) = $w_{kl}$ (lama) + $\Delta w_{kl}$ (14)
**Langkah 9** : Memeriksa kondisi STOP Untuk memeriksa kondisi, digunakan kriteria MSE (Mean Square Error) berikut :

$$e^2 = \frac{1}{2 * p} \sum_p (d^p - y^p)^2 \qquad (15)$$

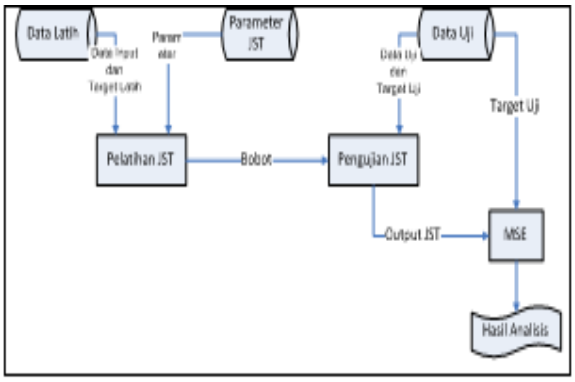
**Langkah 10 :** Menghitung output jaringan ridge polynomial. Menghitung persamaan output jaringan ridge polynomial dengan persamaan (10)

$$y_{RPNN} = \sum_{j=1}^{N} \prod_{i=1}^{j} \left( \sum_{k=1}^{n} w_{ijk} x_k + \theta_{ji} \right) \qquad (16)$$

## IV. HASIL DAN PEMBAHASAN

### 4.1 Bagan Proses Sistem

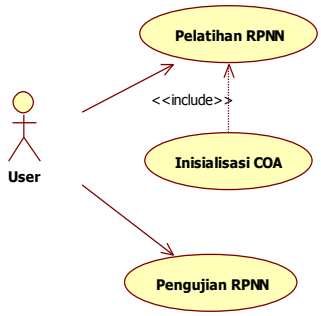
PSN, parameter pembelajaran dan data uji yang terdiri dari data input Estimasi biaya perangkat lunak pada Gambar 2, pertama kali menerima masukan jumlah neuron layer PSN, parameter pembelajaran dan data latih yang terdiri dari data input latih dan data target latih pada saat proses pelatihan jaringan syaraf tiruan. Setelah proses pelatihan JST, sistem yang dibangun menghasilkan bobot dan bias. Kemudian sistem menerima masukan jumlah neuron layer uji dan data target uji pada saat proses pengujian jaringan syaraf tiruan. Setelah proses pengujian JST, sistem yang dibangun menghasilkan output jaringan syaraf tiruan. Hasil analisis adalah mengamati kesalahan sistem yang dibuat dengan cara menghitung kesalahan jaringan yaitu MSE.



Gambar 2 Bagan Proses Sistem Estimasi Biaya Perangkat Lunak

### 4.2 Use Case

Rancangan sistem yang menggunakan use case diagram yang diperlihatkan pada Gambar 3.



Gambar 3 Use Case Diagram

Terdapat 3 use case yang digunakan dalam penelitian ini, yaitu

1. Melakukan pelatihan jaringan syaraf tiruan dengan algoritma pembelajaran ridge polynomial neural network.
2. Sebelum melakukan pelatihan, jaringan menentukan bobot dan bias awal dengan algoritma optimasi chaos.
3. Melakukan pengujian jaringan syaraf tiruan dengan bobot dan bias awal dari pelatihan sebelumnya.

### 4.3 Pelatihan Jaringan Saraf Tiruan

Hasil percobaan pertama adalah hasil pelatihan jaringan dengan menggunakan parameter di jaringan ridge polynomial dan di algoritma optimasi chaos hasil dari data proyek NASA. Parameter di jaringan ridge polynomial yaitu learning rate dimana ketika setiap kali PSN ditambahkan learning rate menurun dengan pembagi 1.7, error pada PSN awal = 0.00007 ketika setiap kali PSN ditambahkan menurun dengan pembagi 10. Parameter di algoritma optimasi chaos yaitu maksimum iterasi pencarian pertama $N_1$ = 5000, maksimum iterasi pencarian kedua $N_2$ = 10000 dan rentang penyempit $p$ adalah 0.1. Interval percarian pertama = [-50, +50] dan interval pencarian kedua = [-2, +2]. Untuk error pada PSN awal = 0.00007, hasil data pelatihan berbeda. Pelatihan akan berhenti jika mendapatkan MSE < error RPNN, dimana error RPNN adalah target kesalahan yang diinginkan yaitu selisih antara data target dengan output jaringan. Jika MSE tidak terpenuhi

maka pelatihan akan berhenti pada maksimum iterasi yang dimasukkan. Setiap arsitektur jaringan mempunyai bobot akhir yang akan digunakan untuk bobot awal proses pengujian.
Hasil pelatihan ditampilkan dalam Tabel 2 sampai dengan Tabel 8.

TABLE II. HASIL PELATIHAN 1

| Learning rate | Iterasi ke- | MSE |
|---|---|---|
| 0.1 | 839 | 0.022270 |
| 0.2 | 486 | 0.022114 |
| **0.3** | **393** | **0.022099** |
| 0.4 | 327 | 0.022115 |
| 0.5 | 266 | 0.022192 |
| 0.6 | 233 | 0.022125 |
| 0.7 | 215 | 0.022302 |
| 0.8 | 201 | 0.022243 |
| 0.9 | 186 | 0.022321 |

TABLE III. HASIL PELATIHAN 2

| Learning rate | Iterasi ke- | MSE |
|---|---|---|
| 0.1 | 843 | 0.022249 |
| 0.2 | 565 | 0.022178 |
| **0.3** | **451** | **0.021954** |
| Learning rate | Iterasi ke- | MSE |
| 0.4 | 364 | 0.022127 |
| 0.5 | 358 | 0.022139 |
| 0.6 | 278 | 0.022155 |
| 0.7 | 282 | 0.022256 |
| 0.8 | 262 | 0.022302 |
| 0.9 | 192 | 0.022325 |

TABLE IV. HASIL PELATIHAN 3

| Learning rate | Iterasi ke- | MSE |
|---|---|---|
| 0.1 | 1101 | 0.022342 |
| 0.2 | 1073 | 0.278849 |
| 0.3 | 910 | 0.278850 |
| **0.4** | **519** | **0.022058** |
| 0.5 | 597 | 0.278852 |
| 0.6 | 527 | 0.278854 |
| 0.7 | 442 | 0.278845 |
| 0.8 | 224 | 0.022288 |
| 0.9 | 425 | 0.278850 |

TABLE V. HASIL PELATIHAN 4

| Learning rate | Iterasi ke- | MSE |
|---|---|---|
| 0.1 | 1787 | 0.022416 |
| 0.2 | 1037 | 0.024059 |
| 0.3 | 576 | 0.022227 |
| **0.4** | **1020** | **0.022045** |
| 0.5 | 665 | 0.278841 |
| 0.6 | 1461 | 0.278854 |
| 0.7 | 599 | 0.278853 |
| 0.8 | 499 | 0.022559 |
| 0.9 | 812 | 0.278853 |

TABLE VI. HASIL PELATIHAN 5

| Learnig rate | Iterasi ke- | MSE |
|---|---|---|
| 0.1 | 2307 | 0.022453 |
| **0.2** | **2198** | **0.022096** |
| 0.3 | 1237 | 0.278850 |
| 0.4 | 1667 | 0.278851 |
| 0.5 | 1354 | 0.278853 |
| 0.6 | 1975 | 0.278829 |
| 0.7 | 712 | 0.278848 |
| 0.8 | 1154 | 0.278852 |
| 0.9 | 1229 | 0.278644 |

TABLE VII. HASIL PELATIHAN 6

| Learning rate | Iterasi ke- | MSE |
|---|---|---|
| **0.1** | **3155** | **0.022093** |
| 0.2 | 1842 | 0.022212 |
| 0.3 | 1487 | 0.022131 |
| 0.4 | 4885 | 0.278852 |
| 0.5 | 1354 | 0.104362 |
| 0.6 | 965 | 0.022164 |
| 0.7 | 1091 | 0.278853 |
| 0.8 | 2431 | 0.278851 |
| 0.9 | 1283 | 0.278855 |

TABLE VIII. HASIL PELATIHAN 7

| Learning rate | Iterasi ke- | MSE |
|---|---|---|
| 0.1 | 2390 | 0.278849 |
| 0.2 | 3346 | 0.022510 |
| **0.3** | **4999** | **0.022119** |
| 0.4 | 3172 | 0.278854 |
| 0.5 | 4999 | 0.022166 |
| 0.6 | 3134 | 0.278846 |
| 0.7 | 4495 | 0.022357 |
| 0.8 | 1612 | 0.278800 |
| 0.9 | 4999 | 0.278848 |

Dari hasil percobaan tersebut dapat diambil kesimpulan terlihat pada tabel 9.

TABLE IX. KESIMPULAN HASIL PELATIHAN DENGAN PELATIHAN 2 DAN 8

| Percobaan | Pelatihan | Learning rate | Iterasi ke- | MSE |
|---|---|---|---|---|
| 1 | 1 | **0.3** | **393** | **0.022099** |
| 2 | 2 | **0.3** | **451** | **0.021954** |
| 3 | 3 | **0.4** | **519** | **0.022058** |
| 4 | 4 | **0.4** | **1020** | **0.022045** |
| 5 | 5 | **0.2** | **2198** | **0.022096** |
| 6 | 6 | **0.1** | **3155** | **0.022093** |
| 7 | 7 | **0.3** | **4999** | **0.022119** |

Proses pelatihan yang dilakukan dengan 7 kali percobaan, didapat hasil MSE terkecil pada percobaan ke-2, yaitu: Learning rate = 0.3, pelatihan 2, maksimum iterasi = 451, MSE = 0.0219. Untuk menguji pengaruh parameter input algoritma optimasi chaos pada jaringan ridge polynomial, maka dilakukan pengujian dengan melakukan variasi pada maksimum iterasi dan interval pencarian pada pencarian pertama dan pencarian kedua.

4.3.1 Pengaruh parameter interval pencarian

Interval pencarian mentukan proses pencarian solusi yang optimal. Jika nilai awal sudah ditentukan pada hasil perhitungan pertama, tidak akan pernah sama dengan nilai

awal pada hasil perhitungan kedua yang hampir mendekati nilai awal pertama. Pada pembahasan ini akan diamati pengaruh interval pencarian pada sistem dengan memperluas atau menyempit interval pencarian. Percobaan pertama adalah dengan mengambil parameter di jaringan ridge polynomial yaitu learning rate dimana ketika setiap kali PSN ditambahkan learning rate menurun dengan pembagi 1.7, error pada PSN awal = 0.00007 ketika setiap kali PSN ditambahkan menurun dengan pembagi 10. Dan parameter di algoritma optimasi chaos yaitu maksimum iterasi pencarian pertama $N_1$ = 5000 dan maksimum iterasi pencarian kedua $N_2$ = 1000. Dan Learning rate = 0.3, arsitektur = 6-2-1, maksimum iterasi = 5000. Dengan menambah dan mengurangi interval pencarian dan tetap menggunakan nilai parameter lain yang sama. Setelah dilakukan percobaan yang berbeda diperoleh hasil seperti Tabel 10 dengan mengubah pada interval pencarian pertama.

TABLE X. PENGARUH PARAMETER INTERVAL PENCARIAN PADA PENCARIAN PERTAMA

| Pencarian pertama | Pencarian kedua | Iterasi ke- | MSE |
|---|---|---|---|
| [-100, 100] | [-2, 2] | 445 | 0.022138 |
| [-90, 90] | [-2, 2] | 580 | 0.022142 |
| **[-80, 80]** | **[-2, 2]** | **427** | **0.022011** |
| [-70, 70] | [-2, 2] | 452 | 0.022269 |
| [-60, 60] | [-2, 2] | 478 | 0.022129 |
| [-50, 50] | [-2, 2] | 451 | 0.022037 |
| [-40, 40] | [-2, 2] | 408 | 0.022189 |
| [-30, 30] | [-2, 2] | 420 | 0.022101 |
| [-20, 20] | [-2, 2] | 415 | 0.022029 |
| [-10, 10] | [-2, 2] | 383 | 0.022145 |

Dari Tabel 10 diketahui bahwa nilai MSE terkecil yaitu **0.022011** dicapai dengan jumlah iterasi 427 dengan interval pencarian pertama [-80, 80] dan interval pencarian kedua [-2, 2].

4.3.2 Pengaruh parameter maksimum iterasi

Jumlah iterasi akan menentukan proses pencarian solusi lebih baik. Lebih banyak jumlah iterasi, proses pencarian lebih mungkin untuk mendapatkan solusi yang diinginkan. Pada pembahasan ini akan diamati pengaruh jumlah ietrasi terhadap nilai MSE yang terkecil.

Pada percobaan ini adalah dengan mengambil jumlah iterasi kedua = 10.000, interval pencarian pertama [-80, 80] dan interval pencarian kedua [-8, 8] pada algoritma optimasi chaos. Percobaan selanjutnya adalah dengan menambah jumlah iterasi dan tetap menggunakan nilai parameter lain yang sama seperti pada percobaan sebelumnya. Setelah dilakukan percobaan yang berbeda diperoleh hasil seperti Tabel 11 dengan mengubah maksimum iterasi pada pencarian pertama.

TABLE XI. PENGARUH PARAMETER MAKSIMUM ITERASI PADA PENCARIAN PERTAMA

| Pencarian pertama | Pencarian kedua | Iterasi ke- | MSE |
|---|---|---|---|
| 1000 | 10.000 | 428 | 0.022324 |
| | | | |
| 2000 | 10.000 | 440 | 0.022640 |
| 3000 | 10.000 | 437 | 0.022104 |
| 4000 | 10.000 | 423 | 0.022069 |
| 5000 | 10.000 | 406 | 0.022188 |
| 6000 | 10.000 | 382 | 0.022650 |
| 7000 | 10.000 | 435 | 0.022253 |
| 8000 | 10.000 | 493 | 0.022141 |
| 9000 | 10.000 | 450 | 0.021943 |
| **10.000** | **10.000** | **436** | **0.021844** |

Dari Tabel 11 diketahui bahwa nilai MSE terkecil yaitu **0.021844** dicapai dengan jumlah iterasi 436 dengan maksimum iterasi pada pencarian pertama 10.000 dan maksimum iterasi pada pencarian kedua 10.000 Kesimpulan dari nilai yang sama di maksimum iterasi pada pencarian kedua pada setiap percobaan diperoleh nilai MSE yang berubah-ubah.

## V KESIMPULAN

Berdasarkan penelitian yang telah dilakukan, maka diambil beberapa kesimpulan sebagai berikut :

1. Jaringan ridge polynomial memiliki kemampuan yang baik dalam memprediksi estimasi biaya perangkat lunak, menggunakan arsitektur unit neuron jaringan pi-sigma dua.
2. Penggunaan interval pencarian pertama dan pencarian kedua pada algoritma optimasi chaos tidak mempunyai pengaruh yang signifikan pada proses pelatihan.
3. Penggunaan maksimum iterasi pencarian pertama dan interval pencarian kedua pada algoritma optimasi chaos tidak mempunyai pengaruh yang signifikan pada proses pelatihan.

## DAFTAR PUSTAKA


[1] Ammaruekarat, P., Meesad, P., A Chaos Search for Multi-Objective Memetic Algorithm, *International* Conference on Information and Electronics Engineering *IPCSIT* vol.6 (2011) © IACSIT *Press*, Singapore.

[2] Epitropakis, M. G., Vrahatis, M. N., Root finding and approximation approaches through neural networks, *ACM SIGSAM Bulletin*, Vol 39, No. 4, Desember 2005.

[3] Fausett L., 1994, Fundamentals of Neural Networks (Architectures, Algorithms, and Applications), Prentice Hall, Englewood Cliffs, New Jersey.

[4] Karnavas, Y.L., Papadopoulus, D.P., 2004, Exicitation Control of a Synchronous Machine Using Polynomial Neural Network, *Journal of Electrical Engineering, Vol.55, No.7-8, 169-179, ISSN 1335-3632*.

[5] Khoa T. Q. D. dan Nakagawa M., Neural Network Learning based on Chaos, *International Journal of* Computer and Information Engineering 1:2 2007.

[6] Leung, H., & Fan, Z. (2001). *Software Cost Estimation*, Departemen of Computing, The Hongkong Polytechnic University, 1- 14.

[7] Sarno, R., Buliali, J.L., Maimunah, S., Pengembangan Metode Analogy untuk estimasi Rancang Bangun Perangkat Lunak, Makara Teknologi,Vol.6,No.2, Agustus 20002,hal 46-55.

[8] Zhang, R., Zheng, X., A New Flatness Pattern Recognition Model Based on Variable Metric Chaos Optimization Neural Network, *R. Zhu et al. (Eds.): ICICA 2010, LNCS 6377, pp. 357-364, Springer-Verlag Berlin Heidelberg*.

[9] Shin, Y., Ghosh, J., Approximation of Multivariate Functions Using Ridge Polynomial Networks, *Department* of Electrical and Computer Engineering, The University of Texas at Austin, 1991.

[10] Gulick, D., 1992, *Encounters With Chaos,* McGRAW HIL INTERNATIONAL EDITIONS, Singapore.