I.J. Intelligent Systems and Applications, **2018, 9, 13-22**
Published Online September 2018 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2018.09.02

A Swarm Intelligence based Chaotic Morphological Approach for Software Development Cost Estimation

Saurabh Bilgaiyan, Kunwar Aditya, Samaresh Mishra and Madhabananda Das
School of Computer Engineering, Kalinga Institute of Industrial Technology (KIIT)
(Deemed to be University), Bhubaneswar-751024, Odisha, India
E-mail: {saurabh.bilgaiyanfcs, 1421019, smishrafcs, mndas_prof}@kiit.ac.in

Received: 02 September 2017; Accepted: 20 December 2017; Published: 08 September 2018

***Abstract*—**In the last century, with the inception of various software development industries at around mid-1960's, the complexities and size of the software have always been a major concern for the industries. The ad-hoc process of development has evolved into a standardized one due to the increase in the size and complexity of software projects. The standardized process of software development was further evolved to predict the overall cost required for the development before the software is actually built. To achieve the same, many cost estimation methodologies have already been successfully implemented, each with certain pros and cons. The present scenario demands even further refined and accurate predictions, which the above-said methods cease to provide. In this paper, we present a chaotically modified particle swarm optimization (CMPSO) based morphological learning approach to accurately estimate the cost incurred in the development process. The proposed approach focuses on a mathematical morphological (MM) framework based hybrid artificial neuron (also called dilation-erosion perceptron or DEP) with algebraic foundations in complete lattice theory (CLT). The proposed CMPSO-DEP model was tested on 5 well-known datasets of software projects with three popular performance metrics and the results were compared with the best existing models available in the literature.

***Index Terms*—**Software development, cost estimation, genetic algorithm (GA), particle swarm optimization (PSO), dilation-erosion perceptron.

I. Introduction

Building high-quality software within time and given budget is the major aim of software development. With an increase in the demand for software industries and societies,much relevant software has been developed over the ages to meet this ever-increasing demand. Due to this a proper project planning and management technique is required. In order to tackle the various issues such as poor performance, delay in software completion time and higher software production costs, etc. various techniques are being used [1, 2].

The 1994 Standish Group Chaos Report reported that 16.2% of the projects were successfully completed and delivered on time, within the given budget, whereas 52.7% exceeded the time and the given budget [3]. But over the years with the introduction of better estimation techniques, the percentage has gone up and a recent report of Standish Group Chaos Report 2013 reported that 39% of the projects were successfully completed and delivered on time, within the given budget, whereas 43% exceeded the time and the given budget. Hence, new methods with improved software development cost estimation (SDCE) techniques are of prime importance for project success [4].

Software cost estimation is basically an estimation of effort needed in the development process of a software system. The accuracy of estimation values is a challenging aspect here [5, 6]. Many uncertain variables such as programming language, development process, organization, etc. are there in software cost estimation which makes the size of software a fuzzy number. Even at the end of the project some important variables are identified which makes the process even more complex [7, 8]. Proper allocation of resources according to a general plan and prioritization and categorization of software projects can be done with accurate cost estimation [5, 8].

Literature shows that many diversified applications have been proposed using morphological neural networks (MNNs) [9]. It is basically a type of artificial neural network based on the principle of MM where Classical Lattice Theory contains the underlying framework of MM. MNNs typically process an elementary operation of MM at each node of the network [10]. The MNN defers from the classical neural network in the way that the operations involved in classical neural network merge and process the information by multiplying and summing the output values with their corresponding weights, whereas the MNNs process involves adding values with their respective weights followed by selecting the highest value from it [11].

The layout of this paper is as follows: Section 2

presents the existing work in the related field. Section 3 presents an insight into the fundamentals of MNNs and DEP. In section 4, we propose our swarm intelligence based learning method followed by performance metrics in section 5. Section 6 presents the simulations and experimental analysis. Section 7 presents the conclusion and future work.

II. Related Works

The DEP and soft computing techniques have been successfully used for prediction problems among the many proposed MNNs, which utilizes MM operators on CLT. Araújo et al., [9] proposed a hybrid artificial neuron called as DEP. The proposed DEP model uses a modified genetic algorithm (MGA) for setting up its parameters and it was successfully tested for SDCE problem. Araújo et al., [1] designed a morphological rank linear perceptron (MRL) using hybrid methods for successfully solving the SDCE problem. In this MRL, perceptrons parameters were optimized using MGA. Oliveira et al., [12] presented a machine learning (ML) based feature selection and parameter optimization method based on GA for SDCE. The accuracy of the estimated effort is improved by using input feature selection and parameter optimization method of ML. Braga et al., [13] presented the use of ML methods along with robust confidence interval for improving the accuracy of estimated software effort. The training set's probability distribution of error has no dependence on this method. Araújo et al., [14] proposed a MRL approach for solving SDCE problem comprising of the hybrid morphological model. This model consists of a linear combination of finite impulse response (FIS) and a Morphological Rank operator. For adjusting the parameters of MRL model, a method, gradient steepest descent (GSD) is used. Araújo et al., [15] proposed a hybrid technique for designing MRL perceptron using least mean square (LMS) algorithm and MGA called as morphological rank linear hybrid intelligent design (MRLHID) to provide a solution for the SDCE problem. The performance of the perceptron was improved along with determining the best particular features using the MGA. For optimizing the parameters of MRL supplied by MGA, a GSD method is used. Araújo et al., [16] presented a shift-invariant morphological system to solve the SDCE problem using a MRL filter composed of a hybrid morphological model using MR and FIR operators. To optimize MRL parameters, a GSD method along with LMS algorithm is used. Araújo et al., [17] presented a morphological approach using gradient method based on MM having an underlying algebraic foundation in lattice theory. For solving the SDCE problem, this model uses a combination of dilation and erosion operators from MM.

The dilation and erosion operators suffer from the problem of non-differentiability which is the major drawback of the classical DEP model and an efficient methodology is needed to overcome this problem which makes the learning process unstable [17].

In this sense, we propose a swarm intelligence based chaotic morphological approach for SDCE problem. The proposed model uses the DEP with a swarm intelligence based algorithm, called as DEP(CMPSO). The model was tested over 5 different complex databases - Albrecht, KotenGray, Kemerer, COCOMO and Desharnais and 3 different performance metrics were used to compare its performance with results obtained previously in the literature.

III. Fundamentals of Mnns and Dep

A. Morphological Neural Network

MNNs are special artificial neural networks also including the classical neural networks (CNNs). The basic difference between CNNs and MNNs is the technique by which the nodes combine the numeric information algebraically [11]. The proposed model focuses on the MNNs with the algebraic foundation in complete lattice theory.

Lattice is a partially ordered set L having every non-empty finite subset's infimum and supremum in L. Arrangement of the type (L, $\vee$, $\wedge$) are called lattice where L is a set, $\vee$ and $\wedge$ are binary operations which satisfy absorption laws and are commutative and associative. Alternatively, lattices are defined as (L, $\leq$), $\leq$ being partial ordering in L having suprema and infima for arbitrary elements of L. Combinedly (L, $\vee$, $\wedge$, $\leq$) defines lattices with both parts being defined by each other [18]. For X $\subseteq$ L, $\wedge_{i\in I}x^i$ gives infimum of X and similarly $\vee_{i\in I}x^i$ gives its supremum where X=$\{x^i, i \in I\}$ [9].

For given lattices L_1 and L_2, a mapping Ψ: $L_1 \rightarrow L_2$ is said to be increasing if and only if [9] Eq. (1) is satisfied:

$$\forall p, q \in L_1, p \leq q \Rightarrow \Psi(p) \leq \Psi(q) \quad (1)$$

A partial ordering on L^n is defined by following Eq. (2):

$$(p_1, p_2, p_3 \dots p_n) \leq (q_1, q_2, q_3 \dots q_n) \Leftrightarrow p_i \leq q_i, i = 1, 2, 3 \dots n \quad (2)$$

The partially ordered set thus obtained is called product lattice [9].

Lattice L is said to be complete if for any arbitrary subset of L, its supremum and infimum exists and its every element can be represented as a join of compact elements. The product lattice L^n is complete lattice if L is complete [10, 19].

Complete lattices are accepted as a theoretical framework for MM. Mapping's decomposition in terms of basic morphological operators between complete lattices is a major issue here [9].

Several theorems have been proposed as a solution on the mapping decomposition on complete lattice based on MM elementary mappings: dilation, anti-dilation, erosionand anti-erosion [20].

Theorem 1. According to this theorem Ψ: $L_1 \rightarrow L_2$ is an increasing mapping between lattices L_1 and L_2 and can be represented by two decompositions, which are infimum and supremum. There exists dilations δ^j for

some index J and erosion $\varepsilon^{\mathbf{k}}$ for some index K given by Eq. (3) [21]:

$$\Psi = \bigwedge_{j\in J} \delta^j and\, \Psi = \bigvee_{k\in K} \varepsilon^k \quad (3)$$

Some specific kinds of MNNs were developed using this theorem in which the elementary operators used required some extra complete lattice assembly. Here the main focus is on complete lattices as the functions $\mathbb{R}^m_{\pm\infty} \rightarrow \mathbb{R}_{\pm\infty}$ can be used to model SDCE problem [21].

For matrices $C\in\mathbb{R}^{m\times s}$ and $D\in\mathbb{R}^{s\times n}_{\pm\infty}$, we define two different kind of matrix products $E=C \vee D$ as max product and $F=C \wedge D$ as min product given by Eq. (4) and (5) [9, 21]:

$$E_{ij=\bigvee_{k=1}^{s}(c_{ik}+d_{kj})} \quad (4)$$

$$F_{ij=\bigwedge_{k=1}^{s}(c_{ik}+d_{kj})} \quad (5)$$

So the matrix product can be used to define the operators of MM where δ_C and $\varepsilon_C : \mathbb{R}^m_{\pm\infty} \rightarrow R^n_{\pm\infty}$ for $C\in\mathbb{R}^{m\times n}$ are given asEq. (6) and (7):

$$\varepsilon_C(x) = C^T \wedge x \quad (6)$$

$$\delta_C(x) = C^T \vee x \quad (7)$$

where T is transposition.

It can be clearly observed that operators δ_C and ε_C represent the algebraic dilation and erosion from $\mathbb{R}^n_{\pm\infty}$ to $\mathbb{R}^m_{\pm\infty}$ which are complete lattices. Thus δ_C and ε_C represents dilation and erosion of every form [21].

In a similar manner $\Psi:\mathbb{R}^n_{\pm\infty} \rightarrow \mathbb{R}^m_{\pm\infty}$ can be assumed to be an increasing mapping. So some matrices C^i and D^j must exist for index I and J such that [21]Eq. (8) and (9):

$$\Psi = \bigvee_{i\in I} \varepsilon_{C^i} \quad (8)$$

$$\Psi = \bigwedge_{j\in J} \delta_{D^j} \quad (9)$$

It can be observed that some vectors u^i and $v^j \in \mathbb{R}$ can be used to approximate the function $\Psi:\mathbb{R}^n \rightarrow \mathbb{R}$ given as Eq. (10) and (11) [21]:

$$\Psi \cong \bigvee_{i\in I} \varepsilon_{u^i} \quad (10)$$

$$\Psi \cong \bigwedge_{j\in J} \delta_{v^j} \quad (11)$$

When indices $\bar{I}$ =1 and $\bar{J}$ =1 the function can be approximated as Eq. (12) and (13) [21]:

$$\Psi \cong \varepsilon_u \quad (12)$$

$$\Psi \cong \delta_v \quad (13)$$

The above equation is the basis for solving the SDCE problems by using morphological perceptrons [9].

B. Dilation-Erosion Perceptron

DEP [9, 21] is a class of hybrid morphological perceptron which is basically a convex combination of dilation and erosion operators MM on CLT which is used to solve the SDCE problem.

Assume a real-valued signal denoted by $y = y_1, y_2, y_3.., y_n \in \mathbb{R}^n$inside a n-point moving window. Here, z denotes the output of the DEP. A shift invariant morphological system is used to define DEP with y→z as local signal conversion given by Eq. (14), (15) and (16):

$$z = \lambda\alpha + (1-\lambda)\beta, \lambda\epsilon[0,1] \quad (14)$$

$$\text{with } \alpha = \delta_a(y) = \bigvee_{j=1}^{n}(y_j + a_j) \quad (15)$$

$$\text{and, } \beta = \varepsilon_b(y) = \bigwedge_{j=1}^{n}(y_j + b_j) \quad (16)$$

where,$\lambda\epsilon\mathbb{R}$, a,b$\in\mathbb{R}^n$, the constructing elements of dilation and erosion (morphological operators) are represented by $a = a_1, a_2, a_3.., a_n$, $b = b_1, b_2, b_3.., b_n$ and n is the input signal dimensionality. The DEP has a convex combination of inversely proportional elements such that when contribution of one element increases then the contribution of the other decreases and vice-versa.

IV. Proposed Swarm Intelligence Based Learning Method

In the DEP (CMPSO) learning process, the weight vector is obtained by Eq. (17):

$$w = (a.b, \lambda) \quad (17)$$

where a. b, λ are the parameters which are needed to be adjusted during the learning process of DEP (CMPSO) until the termination criteria is not satisfied; adjustments being done according to error criterion until convergence of CMPSO generations [9].

Here, weight vector $w_j^{(g)}$ represents a candidate weight vector from the gth generation's j^{th} iteration and a fitness function is used to adjust and asses the quality of weight vector given by Eq. (18):

$$ff\left(w_j^{(g)}\right) = \frac{1}{N}\sum_{k=1}^{N} e^2(k) \quad (18)$$

The count of input patterns is denoted by N and e(k) denotes the instantaneous error given as Eq. (19),

$$e(k) = d(k) - y(k) \quad (19)$$

where the desired output signal is represented by d(k) and y(k) represents the actual output of the sample k.

A. Chaotic Opposition Based Population Initialization

This paper uses a chaotic opposition based initialization to obtain better results instead of using random initialization of population. Chaotic maps are used because of its randomness and sensitivity

dependence on initial conditions. These are also used to further extract search space information and increasing population diversity. Using this method of initialization also increases the convergence speed of proposed CMPSO algorithm [22].

For this method, a sinusoidal iterator was selected given by Eq. (20):

$$cho_{k+1} = \sin(\pi cho_k), cho_k \in [0,1],\ k = 0, 1, 2 \ldots max_k \quad (20)$$

Where k represents the iteration counter and $\max_k$ represents the maximum count of iterations. Fig.1. Shows the basic steps of COPI algorithm.

Algorithm 1: Algorithm for Chaotic Opposition Based Population Initialization (COPI)

```
begin
    (COPI) [22];
    Data: Max chaotic iterations Max_k = 300
    Pop_size=N
    Result: Initial population
    for p=1 to N do
        for q=1 to D do
            Randomly initialize cho_0q ∈ [0, 1];
            for r=1 to Max_k do
                cho_rq = sin(π cho_(r-1,q))
            end
            x_pq = x_minq + cho_rq (x_maxq − x_minq)
        end
    end
    (—Start Opposition Based Method—);
    for p=1 to N do
        for q=1 to D do
            opx_pq = x_minq + x_maxq − x_pq
        end
    end
    Select N fittest individuals from (X(N) ∪ OPX(N)) as initial population
end
```

Fig.1. Basic Steps of PICO Algorithm

B. Linearly Descending Chaotic Inertia Weight (LDCIW)

In order to set the coefficient for inertia weight, mapping such as logistic mapping can be used given by Eq. (21) and (22) [23]:

$$u = 4 * u * (1 - u) \quad (21)$$

1. A specific strategy called chaotic descending inertia weight uses following steps to set the inertia weight (ω):
2. A random number u is selected in the interval [0,1]
3. Then logistic mapping: $u = 4 * u * (1 - u)$

4. $$\omega = (\omega_1 - \omega_2) * \left(\frac{Iter_{Max} - Iter}{Iter_{Max}}\right) + \omega_2 * u \quad (22)$$

Here, the initial and final values are represented by ω_1 and ω_2 respectively, $Iter_{Max}$ and $Iter$ represent the maximum and current iterative time. The use of LDCIW in PSO increases the convergence speed towards optima and also performs global search at the beginning [24].

C. Calculation of Velocity and Position of the Particle

The updation of velocities and positions of the particles is done as given below in Eq. (23) and (24) [25, 26, 27]

$$\vartheta_j^{g+1} = \omega \vartheta_j^g + c_1 r_1 \left(p_{best_j}^g - X_j^g\right) + c_2 r_2 \left(g_{best_j}^g - X_j^g\right) \quad (23)$$

$$X_j^{g+1} = X_j^g + \vartheta_j^{g+1} \quad (24)$$

Where ω is linearly descending chaotic inertia weight derived from Eq. 22 the c1 and c2 are the learning coefficient usually set to 2 and r1 and r2 are random number selected from the interval [0,1] CMPSO starts with population initialization using chaotic opposition based learning method given in Eq. 20 where the velocities are randomly assigned to the particles. In this work, we have taken 10 particles as the pop_{size}. Now the fitness of each particle is calculated using Eq. 18 and then p_{best} (local best of individual particle) and g_{best} (global best for the entire swarm) is calculated. Further the velocities and positions for each particle is updated using the obtained p_{best} and g_{best} values [28]. The steps for DEP(CMPSO) are given as follows in Fig.2.

Algorithm 2: Steps for DEP(CMPSO) Algorithm

```
begin
    DEP(CMPSO) [9, 22];
    Data: Chaotically initialized population according to COPI Algorithm
    Result: Predicted Value
    initialization of CMPSO parameters according to [17, 22];
    initialization of stopping criteria;
    g=0;
    while Termination criteria is not satisfied do
        g=g+1;
        for j=2 to pop_size do
            initialization of DEP parameters taking values from w_j^(g) as given in section 2;
            for all input patterns, calculate the value of y and instantaneous error;
            use Eq. 18 to assess the individual fitness ff(w_j^(g));
            from the population, select the p_best_j and g_best;
            if ff(p_best_j-1) < ff(p_best_j) then
                p_best_j = p_best_j-1
            end
            for j=1 to pop_size do
                Update velocity & position for each particle.
            end
        end
    end
end
```

Fig.2. Basic Steps of DEP(CMPSO) Algorithm

V. Performance Metrics

There are several performance measures available in the literature, but mostly one is used for prediction evaluation. According to [16, 29] mean squared error (MSE) can't be considered as an end measure for comparison between prediction models although it can be used in the training process for driving the prediction model. Because of this reason, the metric presented by [16] are considered, which allows a more robust performance evaluation.

Mean magnitude of relative error (MMRE) is the first metric which accurately identifies the model deviation given by Eq. (25) [9]:

$$MMRE = \frac{1}{N}\sum_{i=1}^{N}\frac{|target_i - predicted_i|}{target_i} \quad (25)$$

where N denotes the count of input patterns, $target_i$ denotes the output for i^{th} pattern and $predicted_i$ is the output value predicted for i^{th} pattern.

The percentage of predictions is used as a second metric also called as PRED (prediction), which falls within the known value, given by Eq. (26) and (27) [9]:

$$PRED(e) = \frac{100}{N}\sum_{i=1}^{N} S_i \quad (26)$$

$$\text{Where } S_i = \begin{cases} 1, & if\ (MMRE_i) < \frac{e}{100}, \\ 0, & otherwise \end{cases} \quad (27)$$

when e=25, them the PRED (e) metric is represented as PRED(25).

A combination of MMRE and PRED(25) called Evaluation function (EF), provides a much robust prediction model given by Eq. (28) [9]:

$$EF = \frac{PRED(25)}{(1+MMRE)} \quad (28)$$

The EF tries to improve the performance of given DEP (CMPSO) by maximizing the value of PRED(25) metric and minimizing the value of MMRE metric which is desired. Thus, EF is considered as a global performance indicator for the SDCE problem.

VI. Simulation and Experimental Analysis

For effective validation of the proposed algorithm, it was tested over five real-world complex SDCE problems (Albrecht, KotenGray, Kemerer, COCOMO and Desharnais). A range of [0,1] was selected for normalization of the datasets in order to maintain the large variations of predictions and to get values close to each other. Further, the datasets were divided into 3 categories: training set (50% of data), a validation set (25% of data) and test set (25% of data). The values of weight vector a, b initialized in the interval [-1, 1] and λ in the interval [0, 1] which is fed as initial input to the COPI algorithm. The parameters were set with the above values after performing extensive research and determining the best values for the constant parameters.

The results were compared with those obtained previously in literature in support vector regression-linear kernel (SVR-LK) [12], support vector regression with radial bias function-kernel (SVR-RBFK) [12], bagging [13] , GA with SVR-RBFK [12], GA with SVR-LK [12], morphological rank-linear (MRL) [17], MRLHID [17], DEP with back propagation algorithm(DEP-BP) [21] and DEP (MGA) [9], for better performance comparison under the same experimental conditions and the same context.

A. Albrecht Dataset

The proposed model was evaluated with Albrecht project dataset as a benchmark. It is built over the details of 24 software projects developed using 3rd generations languages. There are 6 independent features in this dataset: IFC (input feature count) which is the total program's input count without weights and processing complexity adjustment applied in functional point; OFC (output feature count) which is the total program's output count without weights and processing complexity adjustment applied in functional point; QFC (query function count) in which input output enquiry types are counted to measure query count; FPC (file processing count) which is the total count of logical groups of user data or control info and files passed or shared between applications; FP (function point) is a measurement unit for representing the degree of business functionality provided to a user by an information system and SLOC (source line of code) also known as LOC (lines of code) which is a software metric to determine the effort by counting the number of lines of text in the source code, along with a dependent feature which is software development cost (SDC) which needs to be estimated, counted as 1000 person-hour and is the end goal of SDCE problem.For generalization of error, the leave one out cross validation (LOOCV) method is used [1]. Table 1. represents the simulation results of LOOCV for different models present in the literature [9] including the proposed model.

Table 1. Albrecht Dataset Results

Model Name	PRED(25)	MMRE	EF
SVR-LK	58.33	0.6719	34.8884
SVR-RBFK	66.66	0.5072	44.2277
Bagging	70.83	0.4178	49.9577
GA-based with SVR-RBFK	70.42	0.4465	48.6830
GA-based with SVR-LK	56.25	0.6628	33.8285
MRL	70.83	0.4087	50.2804
MRLHID	75.00	0.3810	54.3085
DEP-BP	75.00	0.3699	54.7485
DEP-MGA	75.00	0.3512	55.5062
DEP-CMPSO	78.00	0.3211	59.0417

From this table, it can be observed that best model found in the literature is DEP (MGA) with EF ≅ 55 whereas the proposed DEP (CMPSO) has better performance value=59.0417. It can be seen that proposed model outperforms the DEP (MGA) model in terms of all three performance metrics i.e. PRED(25), MMRE and EF having values as 78.00, 0.3211 and 59.0417 correspondingly. Improvement of 4%, 3.01% and 6.36% was observed in PRED(25), MMRE and EF correspondingly in the proposed model over the DEP (MGA) which is observed as the best model in the literature. Fig.3. shows the graphical comparison of EF obtained for different existing and proposed techniques.

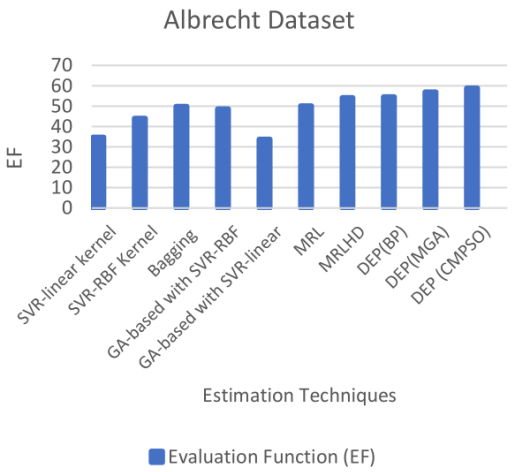

Fig.3. EF Obtained for Albrecht Dataset

B. KotenGray Dataset

The proposed model was evaluated with KotenGray dataset as a benchmark. It is built over the details of 17 different software projects. It contains 7 independent features: PRODUCT (development team's average marking in the practical assessment); FORNMUM (count of data entry forms in function hierarchy diagram (FHD)); RPTNUM (total count of data summary reports in FHD); ENTITYNUM (total database entries in entity relationship diagram (ERD)); ATTNUM (total number of data attributes in ERD); ENTFORM (count of database entities accessed by data entry forms); ENTREPT (database entities used by data summary reports) and a dependent feature which is software development cost (SDC) which needs to be estimated, counted as 1000 person-hour and is the end goal of SDCE problem.

For generalization of error, the LOOCV method is used [1]. Table 2. represents the simulation results of LOOCV for different models present in the literature [9] including the proposed model. From this table, it can be observed that best model found in the literature is DEP (MGA) with EF ≅ 89 whereas the proposed DEP (CMPSO) has better performance value=89.6295. The observation from the table shows that proposed model outperforms the DEP (MGA) model in terms of MMRE and EF with values as 0.0501 and 89.6295 correspondingly. The proposed model has 1.571%, 0.07% improvement in MMRE and EF correspondingly while the value obtained for PRED(25) remains the same i.e. 94.12 over the DEP (MGA) which is observed as the best model in the literature. Fig.4. shows the graphical comparison of EF obtained for different existing and proposed techniques.

Table 2. KotenGray Dataset Results

Model Name	PRED(25)	MMRE	EF
SVR-LK	88.24	0.1133	79.2599
SVR-RBFK	88.24	0.1108	79.4382
Bagging	94.12	0.1001	85.5559
GA with SVR-RBFK	94.12	0.0947	85.9779
GA with SVR-LK	94.12	0.0895	86.3883
MRL	94.12	0.0710	87.8805
MRLHID	94.12	0.0689	88.0531
DEP-BP	94.12	0.0572	89.0276
DEP-MGA	94.12	0.0509	89.5613
DEP-CMPSO	94.12	0.0501	89.6295

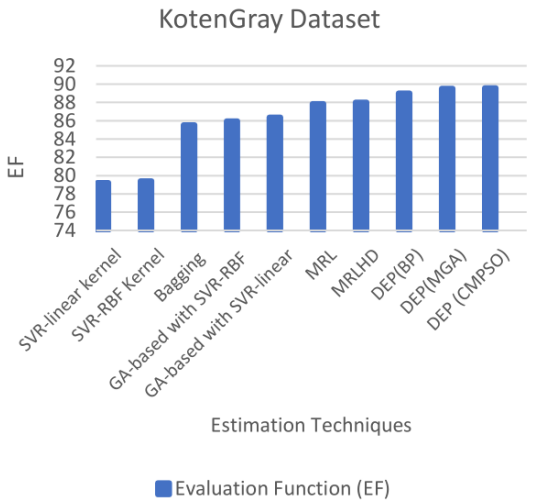

Fig.4. EF Obtained for KotenGray Dataset

C. Kemerer Dataset

The proposed model was evaluated with Kemerer dataset as a benchmark. It is built over the details of 14 different software projects. It contains 6 independent features: PLANG (programming language), HARD (hardware), DURA (duration), KSLOC (kilo source lines of code), Adj FP (adjusted function point), UFP (unadjusted function point) and a dependent feature which is software development cost (SDC) which needs to be estimated, counted as 1000 person-hour and is the end goal of SDCE problem.

For generalization of error, LOOCV method is used [1]. Table 3. represents the simulation results of LOOCV for different models present in the literature [9] including the proposed model. Observation taken from this table shows that best model found in the literature is DEP (MGA) with EF ≅ 58 whereas the proposed DEP (CMPSO) has better performance value=60.5375. it can be observed that proposed model outperforms the DEP (MGA) model in terms of PRED(25), MMRE and EF having values as 75.00, 0.2389 and 60.5375 correspondingly. Improvement of 2.277%, 7.331% and 3.837% in PRED(25), MMRE and EF correspondingly was observed with the proposed model over the DEP (MGA) which is observed as the best model in the literature. Fig.5. shows the graphical comparison of EF obtained for different existing and proposed techniques.

Table 3. Kemerer Dataset results

Model Name	PRED(25)	MMRE	EF
SVR-LK	60.00	0.4608	41.0734
SVR-RBFK	60.00	0.4439	41.5541
Bagging	60.00	0.4297	41.9668
GA with SVR-RBFK	66.67	0.3695	48.6820
GA with SVR-LK	60.00	0.4373	41.7449
MRL	66.67	0.3014	51.2294
MRLHID	73.33	0.2779	57.3832
DEP-BP	73.33	0.2619	58.1108
DEP-MGA	73.33	0.2578	58.3002
DEP-CMPSO	75.00	0.2389	60.5375

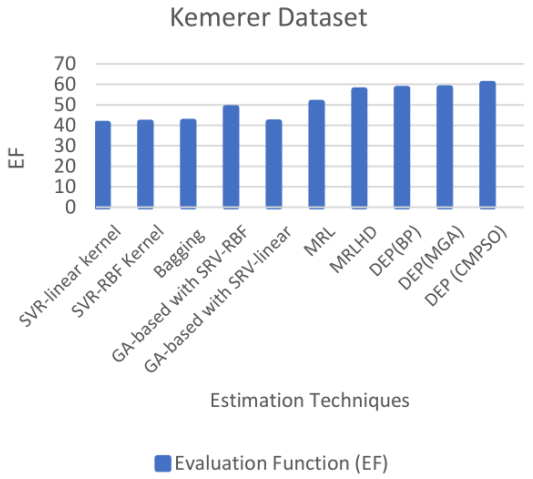

Fig.5. EF Obtained for Kemerer Dataset

D. COCOMO Dataset

The proposed model was evaluated with COCOMO dataset as a benchmark. It is built over the details of 63 different software projects. It contains 20 independent features: LANG (programming language), TEL (time execution limit), SRELY (software reliability), MSL (main storage limit), DATAS (database size), VMK (virtual machine experience), PCLX (product complexity), VMV (virtual machine volatility), RM (runtime machine), PCP (programmer capacity), ACP (analyst capacity), LPEX (language programming experience), AEX (application experience), DP (development platform), MOPM (modern programming methods), ST (software tools), DA (development approach), SCD (scheduled development), RVL (requirement volatility) and CALC (count of adjusted lines of code). EDCM (estimated development cost counted as man per hour) is considered as an only dependent feature.

Table 4. shows the simulation results different models present in the literature [9] including the proposed model. From this table, it can be observed that best model found in the literature is DEP (MGA) with EF ≅ 82 whereas the proposed DEP (CMPSO) has better performance value=82.4340. Observations from the result show that proposed model outperforms the DEP (MGA) model with respect to MMRE and EF having values as 0.1027 and 82.4340 correspondingly. The proposed model has

0.580% and 0.054% improvement in MMRE and EF correspondingly while the value obtained for PRED(25) remains the same i.e. 90.90 over the DEP (MGA) which is observed as the best model in the literature.Fig.6. shows the graphical comparison of EF obtained for different existing and proposed techniques.

Table 4. COCOMO Dataset results

Model Name	PRED(25)	MMRE	EF
SVR-LK	81.82	0.1573	70.6990
SVR-RBFK	72.73	0.1802	61.6251
Bagging	72.73	0.1754	61.8768
GA with SVR-RBFK	72.73	0.1729	62.0087
GA with SVR-LK	81.82	0.1481	71.2656
MRL	81.82	0.1436	71.5460
MRLHID	90.90	0.1298	80.4567
DEP-BP	90.90	0.1127	81.6932
DEP-MGA	90.90	0.1033	82.3892
DEP-CMPSO	90.90	0.1027	82.4340

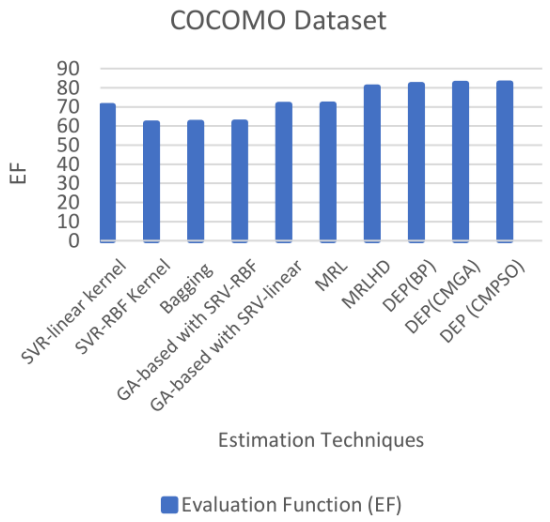

Fig.6. EF Obtained for COCOMO Dataset

E. Desharnais Dataset

The proposed model was evaluated with Desharnais dataset as a benchmark. It is built over the details of 81 different software projects. It contains 9 independent features: TManExpr (team manager experience), TeamExpr (team experience), YearFis (year project finished), Trans (basic logical count of transactions in system), Ent (number of entities in system data model), NonAdjPoint (theses are non-adjusted points calculated as: transactions + entities - FP), TPC(total processing complexity), AdjPoints (FP adjusted by adjustment factor = 0.65 + (0.01*NonAdjPoint) and PL (programming language used), and two dependent features: Cost and Length (ignored because the work focuses on SDCE).

Table 5. shows the simulation results for different models present in the literature [9] including the proposed model. From this table, it can be observed that best model found in the literature is DEP (MGA) with EF ≅ 83 whereas the proposed DEP (CMPSO) has better performance value=83.4982. Observations from the result show that proposed model outperforms the DEP (MGA) model in terms of PRED(25), MMRE and EF having values as 90.27, 0.0811 and 83.4982 correspondingly. Improvement of 0.3%, 2.171% and 1.346% in PRED(25), MMRE and EF correspondingly was observed for the proposed model over the DEP (MGA) which is observed as the best model in the literature. Fig.7. shows the graphical comparison of EF obtained for different existing and proposed techniques.

Table 5. Desharnais Dataset Results

Model Name	PRED(25)	MMRE	EF
SVR-LK	55.00	0.4829	37.0895
SVR-RBFK	60.00	0.4543	41.2570
Bagging	65.00	0.4076	46.1779
GA with SVR-RBFK	80.00	0.3302	60.1413
GA with SVR-LK	80.00	0.3154	61.8180
MRL	85.00	0.1509	73.8552
MRLHID	90.00	0.0981	81.9597
DEP-BP	90.00	0.0835	83.0641
DEP-MGA	90.00	0.0829	83.1102
DEP-CMPSO	90.27	0.0811	83.4982

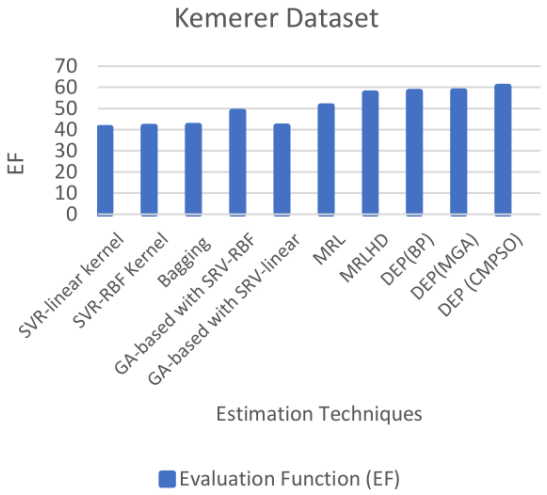

Fig.7. EF Obtained for Desharnais Dataset

VII. Conclusion

This paper presented a swarm intelligence based chaotic morphological approach for solving SDCE problem. In order to achieve a higher precision level for the SDCE problem, we have used CMPSO algorithm to improve and optimize the parameters of DEP perceptron.

To evaluate the performance of proposed CMPSO model, two performance metrics, MMRE and PRED(25) were used. For creating a global indicator of the performance of the proposed model, along with the two metrics, an evaluation function was used. An experimental validation of the proposed model was done over Albrecht, KotenGray, Kemerer, COCOMO and Desharnais datasets. A comparison with the results found in literature shows the robustness of the proposed model. The global indicator (EF) used in experimental validation

shows a more consistent global performance of proposed CMPSO model with 6.36% improvement in Albrecht, 0.07% improvement in KotenGray, 3.837% in Kemerer, 0.054% in COCOMO and 1.346% in Desharnais regarding the best experimental results available in the literature.

The results also show a 2.33345% of average global improvement with respect to DEP-MGA and 2.824% with respect to DEP-BP.

It can be clearly seen that the proposed model has improved and has better predictive performance over other models along with improvements in having simpler nonlinear components and fast convergence (better initial population setup through COPI method).

As future work, authors plan on considering simulation and analysis of proposed model with deeper studies and it will also be tested on other engineering problems such as regression and classification [30].

REFERENCES

[1] Ricardo de A. Araújo, Sergio Soares and Adriano L.I. Oliveira, "Hybrid morphological methodology for software development cost estimation", Expert Systems with Applications, Elsevier, Vol. 39, No. 6, 2012, pp. 6129-6139.

[2] Giuliano Casale, Cristina Chesta, et al., "Current and Future Challenges of Software Engineering for Services and Applications", In Proceedings of CLOUD FORWARD: From Distributed to Complete Computing, Elsevier, Vol. 97, No. 1, 2016, pp. 34-42.

[3] The Standish Group. (1994) CHAOS Report. [Online]. Available on:https://www.standishgroup.com/sample_research_files/chaos_report _1994. pdf.

[4] The Standish Group. (2013) CHAOS Manifesto. [Online]. Available on:https://larlet.fr/static/david/stream/ChaosMani festo2013.pdf.

[5] Fatemeh Zare, Hasan Khademi Zare and Mohammad Saber Fallahnezhad, "Software Effort Estimation based on the Optimal Bayesian Belief Network", Applied Soft Computing, Elsevier, Vol. 49, No. 1, 2016, pp. 968-980.

[6] Saurabh Bilgaiyan, Samresh Mishra and Madhabananda Das, "A Review of Software Cost Estimation in Agile Software Development Using Soft Computing Techniques", 2nd International Conference on Computational Intelligence and Networks (CINE), IEEE, 2016, pp. 112-117.

[7] Chong Wang, Zhong Luo, Luxin Lin and Maya Daneva, "How to Reduce Software Development Cost with Personnel Assignment Optimization: Exemplary Improvement on the Hungarian Algorithm", 21st International Conference on Evaluation and Assessment in Software Engineering, ACM, 2017, pp. 270-279.

[8] Oktay Adalier, Aybars Uğur, Serdar Korukoğlu and Kadir Ertaş, "A New Regression Based Software Cost Estimation Model Using Power Values", International Conference on Intelligent Data Engineering and Automated Learning, LNCS Springer, 2007, pp. 326-334.

[9] Ricardo de A. Araújo, Adriano L.I. Oliveira, Sergio Soares and Silvio Meira, "An Evolutionary Morphological Approach for Software Development Cost Estimation", Neural Networks, Elsevier, Vol. 32, No. 1, 2012, pp. 285-291.

[10] Peter Sussner and Estevão Laureano Esmi, "Morphological perceptrons with competitive learning: Lattice-theoretical framework and constructive learning algorithm", Information Sciences, Elsevier, Vol. 181, No. 10, 2011, pp. 1929-1950.

[11] Jennifer L. Davidson and Frank Hummer, "Morphology neural networks: An introduction with applications", Circuits, Systems and Signal Processing, Springer, Vol. 12, No. 2, 1993, pp. 177-210.

[12] Adriano L.I. Oliveira, Petronio L. Braga, Ricardo M.F. Lima, and Márcio L. Cornélio, "GA-Based Method for Feature Selection and Parameters Optimization for Machine Learning Regression Applied to Software Effort Estimation", Information and Software Technology, Elsevier, Vol. 52, No. 1, 2010, pp. 1155-1166.

[13] Petrônio L. Braga and Adriano L. I. Oliveira, "Software Effort Estimation using Machine Learning Techniques with Robust Confidence Intervals", Seventh International Conference on Hybrid Intelligent Systems, IEEE, 2007, pp. 352-357.

[14] Ricardo de A. Araujo, Adriano L. I. de Oliveira and Sergio C. B. Soares, "A Morphological-Rank-Linear Approach for Software Development Cost Estimation", 21st IEEE International Conference on Tools with Artificial Intelligence, 2009, pp. 630-636.

[15] Ricardo de A. Araujo, Adriano L. I. de Oliveira and Sergio Soares, "Hybrid Intelligent Design of Morphological-Rank-Linear Perceptrons for Software Development Cost Estimation", 22nd International Conference on Tools with Artificial Intelligence, IEEE, 2010, pp. 160-167.

[16] Ricardo de A. Araújo, Adriano L.I. Oliveira and Sergio Soares, "A shift-invariant morphological system for software development cost estimation", Expert Systems with Applications, Elsevier, Vol. 38, No. 4, 2011, pp. 4162-4168.

[17] Ricardo de A. Araújo, Adriano L. I. Oliveira, Sergio Soares and Silvio Meira, "Gradient-based Morphological Approach for Software Development Cost Estimation", Proceedings of International Joint Conference on Neural Networks, IEEE, 2011, pp. 588-594.

[18] Adam Grabowski, "Lattice Theory for Rough Sets - An Experiment in Mizar", CS&P, 2015, pp. 158-169.

[19] Taqseer Khan, "Distributive Lattices", M.S. thesis, Central European UniversityDepartment of Mathematics and its Applications, 2011.

[20] Gerald Jean FrancisBanon, "Decomposition of mappings between complete lattices by mathematical morphology, Part I. General lattices", Signal Processing, Elsevier, Vol. 30, No. 3, 1993, pp. 299-327.

[21] Ricardo de A. Araújo, "A class of hybrid morphological perceptrons with application in time series forecasting", Knowledge-Based Systems, Elsevier, Vol. 24, No. 4, 2011, pp. 513-529.

[22] Wei-feng Gao, San-yang Liu and Ling-ling Huang, "Particle Swarm Optimization with Chaotic Opposition-Based Population Initialization and Stochastic Search Technique", Communications in Nonlinear Science and Numerical Simulation, Elsevier, Vo. 17, No. 11, 2012, pp. 4316-4327.

[23] Yong Feng, Gui-Fa Teng, Ai-Xin Wang and Yong-Mei Yao, "Chaotic Inertia Weight in Particle Swarm Optimization", Second International Conference on Innovative Computing, Information and Control, IEEE, 2007, pp. 1-4.

[24] Martins Akugbe Arasomwan and Aderemi Oluyinka Adewumi, "Improved Particle Swarm Optimization with a Collective Local Unimodal Search for Continuous Optimization Problems", The Scientific World Journal,

2014, pp. 1-23.
[25] H-C. Tsai, "Unified particle swarm delivers high efficiency to particle swarm optimization", Applied Soft Computing, Elsevier, Vol. 55, No. 1, 2017, pp. 371-383.
[26] Saurabh Bilgaiyan, Santwana Sagnika, Samaresh Mishra, Madhabananda Das, "Study of Task Scheduling in Cloud Computing Environment Using Soft Computing Algorithms", International Journal of Modern Education and Computer Science (IJMECS), Vol. 7, No. 3, 2015, pp. 32-38.
[27] Santar Pal Singh, Subhash Chander Sharma, "A Particle Swarm Optimization Approach for Energy Efficient Clustering in Wireless Sensor Networks", International Journal of Intelligent Systems and Applications(IJISA), Vol. 9, No. 6, 2017, pp. 66-74.
[28] Ahmed A. Esmin, Rodrigo A. Coelho and Stan Matwin, "A Review on Particle Swarm Optimization Algorithm and Its Variants to Clustering High-dimensional Data", Artificial Intelligence Review, ACM, Vol. 44, No. 1, 2015, pp. 23-45.
[29] Sumit Goyal,Gyanendra Kumar Goyal, "Time – Delay Simulated Artificial Neural Network Models for Predicting Shelf Life of Processed Cheese", International Journal of Intelligent Systems and Applications(IJISA), Vol. 4, No. 5, 2012, pp. 30-37.
[30] Ayman E. Khedr, S.E.Salama, Nagwa Yaseen, "Predicting Stock Market Behavior using Data Mining Technique and News Sentiment Analysis", International Journal of Intelligent Systems and Applications(IJISA), Vol. 9, No. 7, 2017, pp. 22-30.

Authors' Profiles

Saurabh Bilgaiyan is currently working as a Teaching Associate at Kalinga Institute of Industrial Technology (KIIT) (Deemed to be University), Bhubaneswar, India. He is also pursuing Ph.D. (CSE) at Kalinga Institute of Industrial Technology (KIIT) (Deemed to be University), Bhubaneswar, India. He obtained his Bachelor's degree of B.E. (I.T.) in 2012 from B.I.R.T., Bhopal, India and Master's degree of M.Tech. (CSE) in 2014 from Kalinga Institute of Industrial Technology (KIIT) (Deemed to be University), Bhubaneswar, India. His area of interests includes soft computing, cloud computing, image processing, distributed database systems and software cost estimation.

Kunwar Aditya is currently pursuing B.tech. (CS&E) in Kalinga Institute of Industrial Technology (KIIT) (Deemed to be University), Bhubaneswar, India. His area of interests includes soft computing and software engineering

Samaresh Mishra M.Tech, Ph.D. (Computer Science) is the Dean of School Computer Engineering, Kalinga Institute of Industrial Technology (KIIT) (Deemed to be University), Bhubaneswar, India, India. He has a rich experience of teaching in the field of Computer Science at both UG and PG degree level and has also published many articles in International Conferences and Journals of repute. His area of interest includes Database Engineering, Cost Estimation, Software Reliability and Cloud Computing.

Madhabananda Das M.Tech., Ph.D. is a Senior Professor at the School Computer Engineering, Kalinga Institute of Industrial Technology (KIIT) (Deemed to be University), Bhubaneswar, India. He has a rich experience of teaching in the field of Computer Science at both UG and PG degree level and has also published many articles in International Conferences and Journals of repute. His area of interest includes soft computing, computational intelligence and image processing.

How to cite this paper: Saurabh Bilgaiyan, Kunwar Aditya, Samaresh Mishra, Madhabananda Das, "A Swarm Intelligence based Chaotic Morphological Approach for Software Development Cost Estimation", International Journal of Intelligent Systems and Applications(IJISA), Vol.10, No.9, pp.13-22, 2018. DOI: 10.5815/ijisa.2018.09.02

