Expert Systems with Applications 39 (2012) 6129–6139

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Hybrid morphological methodology for software development cost estimation

Ricardo de A. Araújo *, Sergio Soares, Adriano L.I. Oliveira

Informatics Center, Federal University of Pernambuco, Recife, PE, Brazil

ARTICLE INFO

Keywords:
Software development cost estimation
Morphological-rank-linear perceptrons
Genetic algorithms
Hybrid methodologies
Feature selection

ABSTRACT

In this paper we propose a hybrid methodology to design morphological-rank-linear (MRL) perceptrons in the problem of software development cost estimation (SDCE). In this methodology, we use a modified genetic algorithm (MGA) to optimize the parameters of the MRL perceptron, as well as to select an optimal input feature subset of the used databases, aiming at a higher accuracy level for SDCE problems. Besides, for each individual of MGA, a gradient steepest descent method is used to further improve the MRL perceptron parameters supplied by MGA. Finally, we conduct an experimental analysis with the proposed methodology using six well-known benchmark databases of software projects, where two relevant performance metrics and a fitness function are used to assess the performance of the proposed methodology, which is compared to classical machine learning models presented in the literature.

1. Introduction

The aim of software development is to build software of high quality in time and within budget to assure competitiveness. The absence of planning and project management processes before the project' start and in its execution frequently produce several problems, such as (i) an increase in the software conclusion time, (ii) higher production costs, and (iii) inadequate performance, amongst others (Oliveira, 2006; Oliveira, Braga, Lima, & Cornelio, 2010).

The Standish Group Chaos Report (Group, 2004) recently reported that a wide number of software projects (around 66%) are delivered with time delay, overbudget and many are not even finished. The main causes of these problems is a failure of the software development cost estimation (SDCE) (de Araújo, de Oliveira, & Soares, 2009; de Araújo, Oliveira, & Soares, 2011). Therefore, the SDCE represents an important research area and the most relevant task in software planning and management, where machine learning methods have been applied to solve the estimation (prediction) problems (Braga, Oliveira, Ribeiro, & Meira, 2007, 2008; de Araújo et al., 2009, 2011; Elish, 2009; Li, Wang, Ye, & Li, 2008; Li, Xie, & Goh, 2009; Oliveira, 2006; Shin & Goel, 2000).

Nonlinear operators have been successful applied to signal processing. An important class of nonlinear systems is based on the framework of mathematical morphology (MM) (Matheron, 1975; Serra, 1982). Morphological operators are nonlinear signal transformations that locally modify the geometrical features of signals (Maragos, 1989), and are related to the basic operations of set theory and integral geometry. The idea of the MM of describing the signal structures through the geometrical patterns composition at several signal places is based on the structuring element concept, which is a set or function that describes a simple shape (Maragos, 1989).

Recently, morphological neural networks (MNNs) have been proposed for a wide range of applications (de Araújo, Madeiro, de Sousa, Pessoa, & Ferreira, 2006; de Araújo et al., 2009, 2011; Pessoa & Maragos, 2000; Sussner & Esmi, 2009). MNNs are based on the framework of MM (Matheron, 1975; Serra, 1982) whose algebraic foundations can be found in lattice theory (Heijmans, 1994; Ronse, 1990; Serra, 1988). In contrast to traditional ANN models, MNNs perform operations of MM instead of conventional semi-linear operations (de Araújo & Sussner, 2010). In the strict, lattice-algebraic sense this means that every node of MNN executes an elementary operation of MM between complete lattices, possibly followed by the application of an activation function. In a broader, geometrical sense, MNNs are used to analyze objects using structuring elements that correspond to the synaptic weights of the network (Won, Gader, & Coffield, 1997).

Furthermore, as in practical SDCE applications the number of features used grows, it is necessary to determine best particular features, since all features are not equally important to optimize the prediction model performance (de Araújo, 2007; de Araújo, 2010a, 2010b, 2010c, 2011, in press; de Araújo & Ferreira, 2009, in press). In prediction or estimation problems, a better performance may be achieved by discarding some variables or features, that is, reducing the data dimensionality. However, an exhaustive search through the space of feature subsets is an intractable problem. This is often referred to as the curse of dimensionality

* Corresponding author.
E-mail addresses: raa@cin.ufpe.br (R. de A. Araújo), scbs@cin.ufpe.br (S. Soares), alio@cin.ufpe.br (A.L.I. Oliveira).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.11.077

problem for databases (Braga et al., 2007, Braga, de Oliveira, & Meira, 2008; de Araújo et al., 2009, 2011; Oliveira, 2006; Oliveira et al., 2010).

In this sense, we propose a hybrid methodology to design morphological-rank-linear (MRL) perceptrons (Pessoa & Maragos, 1998) (hybrid morphological model composed of a linear operator (Pessoa & Maragos, 1998) and a nonlinear morphological-rank (MR) (Salembier, 1992a, 1992b) operator) to solve the SDCE problem. In this methodology, we use a modified genetic algorithm (MGA) (Leung, Lam, Ling, & Tam, 2003) to optimize the parameters of the MRL perceptron, as well as to select an optimal input feature subset of the used data sets, aiming at a higher accuracy level for SDCE problems. Besides, for each individual of MGA, we employ a gradient steepest descent method (Pessoa & Maragos, 1998) (using the least mean squares (LMS) algorithm with a systematic approach to overcome the problem of nondifferentiability of the morphological-rank operator) to optimize the MRL perceptron parameters supplied by MGA.

Furthermore, we conduct an experimental analysis with the proposed methodology using Nasa, Desharnais, Cocomo, Albrecht, Kemerer and KotenGray databases of software projects, where two relevant performance metrics and a fitness function are used to assess the performance of the proposed methodology, which is compared to machine learning models presented in the literature (Braga et al., 2007, 2008; de Araújo et al., 2009, 2011; Oliveira, 2006; Oliveira et al., 2010).

This paper is organized as follows. In Section 2 are presented the fundamentals of the SDCE problem. Section 3 presents the MRL perceptron definition and its training algorithm. Section 4 shows the performance measures used to assess performance of the proposed methodology. Section 5 presents the fundamentals and describes the proposed methodology. In Section 6 it is presented the simulations and the experimental results with the proposed model, as well as a comparison between the obtained results with those given by the models previously presented in literature. At the end, in Section 7, it is presented the conclusions of this work.

2. The software development cost estimation problem

The software development cost estimation (SDCE) problem consists of building a database (composed by features of previous software projects, such as features related to products, processes and resources) and quantify its subjective and objective attributes into numerical values (de Araújo et al., 2009, 2011). The estimation (or prediction) term is applied when it used to predict a future attribute value, where in the particular case of SDCE, the attribute which will be predicted is the cost (Braga et al., 2007, 2008). A typical example of SDCE is to use attributes which characterize the software project to predict (or estimate) the cost, in terms of programmers-by-month, to conclude the software development, and consequently it is possible to predict the needed time to deliver the product (Oliveira, 2006; Oliveira et al., 2010).

Therefore, in order to meet the new demands of development, maintenance or customization of software, the project team has to estimate beforehand the software project development time and the project cost for the company. According to de Araújo et al. (2009, 2011), the main risk factors for software projects are the schedule and effort (cost) to finish it; the particularities of software project requisites, project team and the employed technology make the process of cost estimation too hard.

Due to these and other peculiarities of each project, it is known in practice that the accurate measurement of the cost and development time of software is only possible when the project is finished (Braga et al., 2007, 2008; de Araújo et al., 2009, 2011; Oliveira, 2006; Oliveira et al., 2010; Shin & Goel, 2000). However, it is necessary to perform estimations before the project begins. There is a wide number of techniques and methods which can be employed to estimate such variables.

3. The morphological-rank-linear perceptron

3.1. Preliminaries

Definition 1 (*Rank function*). The r-the rank function of a vector $\mathbf{t} = (t_1, t_2, \ldots, t_n) \in \mathbb{R}^n$ is the rth vector element $\mathbf{t}$ ordered by decreasing way ($t_{(1)} \geqslant t_{(2)} \geqslant \cdots \geqslant t_{(n)}$). In this way, the rth rank function of vector $\mathbf{t}$ is given by Pessoa and Maragos (2000)

$$\mathcal{R}_r(\mathbf{t}) = t_{(r)}, \quad r = 1, 2, \ldots, n. \tag{1}$$

For example, given the vector $\mathbf{t} = (3,0,5,7,2,1,3)$, its 4th rank function is $\mathcal{R}_4(\mathbf{t}) = 3$.

Definition 2 (*Unit sample function*). The unit sample function, $q(v)$, is given by Pessoa and Maragos (2000)

$$q(v) = \begin{cases} 1, & \text{if } v = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

where $v \in \mathbb{R}$.

Then, when applying the unit sample function to a vector $\mathbf{v} = (v_1, v_2, \ldots, v_n) \in \mathbb{R}^n$, it returns a vector of unit sample function ($Q(\mathbf{v})$), given by Pessoa and Maragos (2000)

$$Q(\mathbf{v}) = [q(v_1), q(v_2), \ldots, q(v_n)]. \tag{3}$$

Definition 3 (*Rank indicator vector*). The rth rank indicator vector $\mathbf{c}$ of a vector $\mathbf{t}$ is given by Pessoa and Maragos (2000)

$$\mathbf{c}(\mathbf{t}, r) = \frac{Q(z \cdot \mathbf{1} - \mathbf{t})}{Q(z \cdot \mathbf{1} - \mathbf{t}) \cdot \mathbf{1}^T}, \tag{4}$$

where $z = \mathcal{R}_r(\mathbf{t}), \mathbf{1} = (1, 1, \ldots, 1)$ and the symbol $\cdot^T$ denotes transposition. Then, the rank indicator vector mark the locations of vector $\mathbf{t}$ where the value of z occurs. Therefore, it also have a unit area.

For example, given a vector $\mathbf{t} = (3,0,5,7,2,1,3)$, its 4th rank indicator vector is $\mathbf{c}(\mathbf{t}, 4) = \frac{1}{2}(1,0,0,0,0,0,1)$. Below, it will be presented some properties of rank indicator vector.

Preposition 1. *Let $\mathbf{t} \in \mathbb{R}^n, r \in \{1, 2, \ldots, n\}$ and $\mathbf{c} = \mathbf{c}(\mathbf{t}, r)$. Then, we have*

(1) $\mathbf{c} \cdot \mathbf{1}^T = 1$ *(unit area)*
(2) $\mathbf{c} \cdot \mathbf{t}^T = \mathcal{R}_r(\mathbf{t})$ *(representation via inner-product)*
(3) $\mathbf{c} \cdot (z\mathbf{1} - \mathbf{t})^T = 0$ *where* $z = \mathcal{R}_r(\mathbf{t})$
(4) *If r is fixed, then $\mathbf{c}$ is a piecewise constant function $\mathbf{t}$ with exactly $2^n - 1$ distinct possible values. Besides, for all points $\mathbf{t}_0 \in \mathbb{R}^n$ with unequal components, $t_{0,i} \neq t_{0,j} \forall i \neq j$, there are a neighborhood around them so that*

$$\|\mathbf{t} - \mathbf{t}_0\|_\infty < \frac{1}{2} \min_{i \neq j} |t_{0,i} - t_{0,j}|, \tag{5}$$

in which $\mathbf{c}$ is constant, that is, $\mathbf{c}(\mathbf{t}, r) = \mathbf{c}(\mathbf{t}_0, r)$.

Preposition 2. Let $\mathbf{c} = \mathbf{c}(\mathbf{t}, r)$. For a given fixed r, if $\mathbf{c}$ is constant in a neighborhood of some $\mathbf{t}_0$, then the rth rank function $\mathcal{R}_r(\mathbf{t})$ is differentiable in $\mathbf{t}_0$ and

$$\left.\frac{\partial \mathcal{R}_r(\mathbf{t})}{\partial \mathbf{t}}\right|_{\mathbf{t}=\mathbf{t}_0} = \left.\frac{\partial}{\partial \mathbf{t}}(\mathbf{c}(\mathbf{t}_0, r)\cdot \mathbf{t}^T)\right|_{\mathbf{t}=\mathbf{t}_0} = \mathbf{c}(\mathbf{t}_0, r). \quad (6)$$

Otherwise, the rank function $\mathcal{R}_r(\mathbf{t})$ is not differentiable.When the function $z = \mathcal{R}_r(\mathbf{t})$ is not differentiable, is possible to make it differentiable through a methodology, that is to assign to vector $\mathbf{c}$ an unilateral value of discontinuity $\frac{\partial z}{\partial \mathbf{t}}$. Besides, since the rank indicator vector is used to estimate the derivatives and is based on discontinuity of unit sample function, a simple approach to prevent abrupt changes and to achieve numerical robustness that is to replace the unit sample function by smoothed impulses, $q_\sigma(v)$, which depends of a scale factor $\sigma \geqslant 0$, having at least the following properties:

$$q_\sigma(v) = q_\sigma(-v) \text{ (symmetry)}, \quad (7)$$

$$q_\sigma(v) \to q(v) \quad \forall v \quad \text{as } \sigma \to 0, \quad (8)$$

and

$$q_\sigma(v) \to 1 \quad \forall v \quad \text{as } \sigma \to \infty. \quad (9)$$

Functions of kind $e^{\frac{1}{2}\left(\frac{v}{\sigma}\right)^2}$ or $sech^2\left(\frac{v}{\sigma}\right)$ are, in this way, natural choices for $q_\sigma(v)$. Of a similar way to unit sample function $Q(\mathbf{v})$, the smoothed unit sample function is given by Pessoa and Maragos (2000)

$$Q_\sigma(\mathbf{v}) = [q_\sigma(v_1), q_\sigma(v_2), \ldots, q_\sigma(v_n)]. \quad (10)$$

Definition 4 (*Smoothed rank function).* If $q_\sigma(v)$ is a smoothed impulse function which satisfies the imposed conditions presented in Eqs. (7)–(9), then the *r*th smoothed rank function is defined by Pessoa and Maragos (2000)

$$\mathcal{R}_{r,\sigma}(\mathbf{t}) = \mathbf{c}_\sigma \cdot \mathbf{t}^T, \quad (11)$$

where

$$\mathbf{c}_\sigma(\mathbf{t}, r) = \frac{Q_\sigma(z\cdot \mathbf{1} - \mathbf{t})}{Q_\sigma(z\cdot \mathbf{1} - \mathbf{t})\cdot \mathbf{1}^T}, \quad z = \mathcal{R}_r(\mathbf{t}). \quad (12)$$

In this way, the smoothed rank function $\mathbf{c}_\sigma$ represents an approximation of rank function $\mathbf{c}$. The smoothed rank function $\mathbf{c}_\sigma$, using ideas of fuzzy sets theory, can be interpreted as a membership function vector (Pessoa & Maragos, 2000).

For example, given a vector $\mathbf{t} = (3, 0, 5, 7, 2, 1, 3)$, $q_\sigma(v) = sech^2\left(\frac{v}{\sigma}\right)$ with $\sigma = 0.5$, then

$$\mathbf{c}_\sigma(\mathbf{t}, 4) = \frac{1}{2}(0.9646, 0, 0.0013, 0, 0.0682, 0.0013, 0.9646)$$

Below, it will be presented some or properties of smoothed rank indicator vector.

Preposition 3. Let $\mathbf{t} \in \mathbb{R}^n, r \in \{1, 2, \ldots, n\}$ and $\mathbf{c} = \mathbf{c}(\mathbf{t}, r)$. Then, we have

(1) $\mathbf{c}_\sigma \cdot \mathbf{1}^T = 1$ (unit area)
(2) $\lim_{\sigma\to 0}\mathbf{c}_\sigma = \mathbf{c}$
(3) $\lim_{\sigma\to 0} z_\sigma = z$
(4) $\lim_{\sigma\to\infty}\mathbf{c}_\sigma = \frac{1}{n}$
(5) $\lim_{\sigma\to\infty} z_\sigma = \frac{1}{n}\sum_{j=1} n t_j$

3.2. Definition

Let $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ represent the input signal inside an *n*-point moving window and let *y* be the output from the perceptron. Then, the morphological-rank-linear (MRL) perceptron is defined as a shift-invariant system whose local signal transformation rule $\mathbf{x} \to y$ is given by Pessoa and Maragos (1998)

$$y = \lambda\alpha + (1-\lambda)\beta, \quad (13)$$

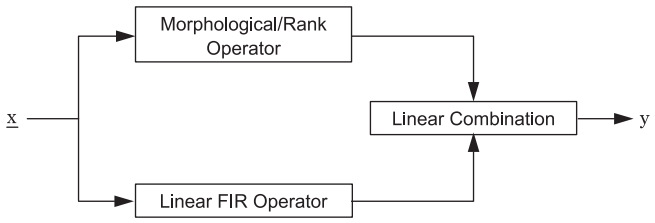

Fig. 1. Structure of the MRL perceptron.

with

$$\alpha = \mathcal{R}_r(\mathbf{x} + \mathbf{a}) = \mathcal{R}_r(x_1 + a_1, x_2 + a_2, \ldots, x_n + a_n), \quad (14)$$

and

$$\beta = \mathbf{x}\cdot \mathbf{b}' = x_1 b_1 + x_2 b_2 + \cdots + x_n b_n, \quad (15)$$

where $\lambda \in \mathbb{R}$, $\mathbf{a}$ and $\mathbf{b} \in \mathbb{R}^n$. Terms $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_n)$ represent both the MR operator and linear FIR operator coefficients, respectively. Term $\mathbf{a}$ is usually referred to "structuring element" because for $r = 1$ or $r = n$ the rank operator becomes the morphological dilation and erosion by a structuring function equal to $\pm\mathbf{a}$ within its support (Pessoa & Maragos, 1998). The structure of the MRL perceptron is illustrated in Fig. 1.

3.3. Training algorithm

According to the DEP definition, it is seen that the main objective is to determine a set of parameters defined by $\mathbf{a}$, $\mathbf{b}$, r and λ. However, instead of using the integer rank parameter r directly in the MRL perceptron definition Eqs. (13)–(15), they argued that it is possible to work with a real variable ρ implicitly defined through the following rescaling,

$$r = round\left(n - \frac{n-1}{exp(-\rho)}\right), \quad (16)$$

where $\rho \in \mathbb{R}, n$ is the dimension of the input signal vector $\mathbf{x}$ inside the moving window and round $(\cdot)$ denotes the usual symmetrical rounding operation.

Therefore, the weight vector to be used in the training process is given by

$$\mathbf{w} = (\mathbf{a}, \mathbf{b}, \rho, \lambda). \quad (17)$$

The framework of the MRL perceptron adaptive design is viewed as a learning process where the perceptron parameters are iteratively adjusted. The usual approach to adaptively adjust the vector $\mathbf{w}$, and therefore designing the perceptron, is to define a cost function $J(\mathbf{w})$, estimating its gradient $\nabla J(\mathbf{w})$, and updating the vector $\mathbf{w}$ by the iterative formula

$$\mathbf{w}(i+1) = \mathbf{w}(i) - \mu_0 \nabla J(\mathbf{w}), \quad (18)$$

where $\mu_0 > 0$ (usually called step size) and $i \in \{1, 2, \ldots\}$. The term μ_0 is responsible for regulating the tradeoff between stability and speed of convergence of the iterative procedure. The iteration of Eq. (18) starts with an initial guess $\mathbf{w}(0)$ and stops when some desired condition is reached. This approach is known as the method of gradient steepest descent (Pessoa & Maragos, 1998).

The cost function *J* must reflect the solution quality achieved by the parameters configuration of the system. A cost function *J*, for example, can be any error function, such as

$$J[\mathbf{w}(i)] = \frac{1}{M}\sum_{k=i-M+1}^{i} e^2(k), \quad (19)$$

where $M \in \{1, 2, \ldots\}$ is a memory parameter and $e(k)$ is the instantaneous error, given by

$$e(k) = d(k) - y(k), \tag{20}$$

where $d(k)$ and $y(k)$ are the desired output signal and the actual perceptron output for the training sample k, respectively. The memory parameter M controls the smoothness of the updating process. If we are processing noiseless signals, $M = 1$ is recommended (Pessoa & Maragos, 1998). However, when we use $M > 1$, the updating process tends to reduce the noise influence of noisy signals during the training (Pessoa & Maragos, 1998).

Hence, the resulting adaptation algorithm is given by Pessoa and Maragos (1998)

$$\mathbf{w}(i+1) = \mathbf{w}(i) + \frac{\mu}{M} \sum_{k=i-M+1}^{i} e^2(k) \frac{\partial y(k)}{\partial \mathbf{w}}, \tag{21}$$

where $\mu = 2\mu_0$ and $i \in \{1,2,\ldots\}$. From Eqs. (13)–(15), term $\frac{\partial y(k)}{\partial \mathbf{w}}$ (Pessoa & Maragos, 1998) may be calculated as

$$\frac{\partial y}{\partial \mathbf{w}} = \left(\frac{\partial y}{\partial \mathbf{a}}, \frac{\partial y}{\partial \mathbf{b}}, \frac{\partial y}{\partial \rho}, \frac{\partial y}{\partial \lambda}\right) \tag{22}$$

with

$$\frac{\partial y}{\partial \mathbf{a}} = \lambda \frac{\partial \alpha}{\partial \mathbf{a}}, \tag{23}$$

$$\frac{\partial y}{\partial \mathbf{b}} = (1-\lambda)\mathbf{x}, \tag{24}$$

$$\frac{\partial y}{\partial \rho} = \lambda \frac{\partial \alpha}{\partial \rho}, \tag{25}$$

$$\frac{\partial y}{\partial \lambda} = (\alpha - \beta), \tag{26}$$

where

$$\frac{\partial \alpha}{\partial \mathbf{a}} = \mathbf{c} = \frac{Q((\alpha \cdot \mathbf{1}) - \mathbf{x} - \mathbf{a})}{Q((\alpha \cdot \mathbf{1}) - \mathbf{x} - \mathbf{a}) \cdot \mathbf{1}'}, \tag{27}$$

$$\frac{\partial \alpha}{\partial \rho} = 1 - \frac{1}{n} Q((\alpha \cdot \mathbf{1}) - \mathbf{x} - \mathbf{a}) \cdot \mathbf{1}', \tag{28}$$

where n is the dimension of $\mathbf{x}$ and $\alpha = \mathcal{R}_r(\mathbf{x} + \mathbf{a})$.

Fig. 2 presents the training algorithm steps. It is worth mentioning that three termination conditions are used in the proposed training algorithm:

(1) The maximum epoch number: *epoch* = 10000;
(2) The decrease in the training error process training (*Pt*) (Prechelt, 1994) of the cost function: $Pt \leqslant 10^{-6}$.
(3) The increase in the validation error or generalization loss (*Gl*) (Prechelt, 1994) of the cost function: $Gl > 5\%$;

According to the Fig. 2, in the weight vector initialization it is used a random process to determine the vectors **a** and **b** (normalized in range $[-1,1]$), the mix term λ (normalized in range $[0,1]$) and the term ρ (normalized in range $[-n,n]$).

```
begin Training Algorithm Steps
    initialize parameters (μ, epoch, w and termination condition);
    τ = 0;              // τ: actual epoch
    while not termination condition do
        τ = τ+1;
        for m = 1 to M do
            calculate output y_1 for m-th input pattern;
            calculate the instantaneous error (e(m));
            calculate the gradient (∇J(w));
            calculate the cost function (J(w));
            update weight vector w;
        end
    end
end
```

Fig. 2. MRL perceptron training algorithm.

4. Performance metrics

The first metric used in this work is able to accurately identifying model deviations, referred to as mean magnitude of relative error (MMRE), which is given by

$$\text{MMRE} = \frac{1}{N} \sum_{i=1}^{N} \frac{|\text{target}_i - \text{output}_i|}{\text{target}_i}. \tag{29}$$

where N is the number of patterns, target_i is the desired output for pattern i and output_i is the predicted value for pattern i.

The second metric used in this work is defined as the percentage of predictions falling within the actual known value, referred to as PRED, which is given by

$$\text{PRED}(x) = \frac{100}{N} \sum_{i=1}^{N} D_i, \tag{30}$$

where

$$D_i = \begin{cases} 1, & \text{if } (\text{MMRE}_i) < \frac{x}{100}, \\ 0, & \text{otherwise}, \end{cases} \tag{31}$$

in which term x is equals to 25 (then the PRED metric is defined as PRED(25)).

5. The proposed method

The proposed method, referred to as morphological-rank-linear hybrid design (MRLHD), is proposed to solve the software development cost estimation (SDCE) problem. It uses an evolutionary search procedure (Leung et al., 2003) to determine the elements necessary for building an accurate system: (i) the underlying information necessary to estimate the cost (the minimum dimensionality, or the minimum number of features, for representing the data), and (ii) the model structure, parameters and training algorithm. It is important to consider the minimum dimensionality in the data because the model must be as parsimonious as possible.

The proposed method uses a modified genetic algorithm (MGA) (Leung et al., 2003) (using optimal genetic operators to accelerate its convergence) to design morphological-rank-linear (MRL) perceptrons. The purpose of using the MGA (Leung et al., 2003) is to identify the minimum data dimensionality and their corresponding specific features to represent the data (initially, a maximum dimensionality (*MaxDim*) is defined by the number of data features and then the MGA can select any value in the interval [1,*MaxDim*] for each individual of the population), as well as to search for the best initial MRL perceptron parameters ($\mathbf{a}, \mathbf{b}, \rho$ and λ) to solve the software development cost estimation problem. For each MGA individual, it uses the least mean squares (LMS) algorithm (Pessoa & Maragos, 1998) to optimize the MRL perceptron parameters supplied by MGA, since it has proved to be effective in speeding up the training process while limiting its computational complexity.

The MGA used here is based on the work of Leung et al. (2003). The MGA procedure consists of selecting a parent pair of chromosomes and then performing crossover and mutation operators (generating the offspring chromosomes – the new population) until the termination condition is reached; then the best individual in the population is selected as a solution to the problem. The MGA algorithm is described in Fig. 3.

The crossover operator is used for exchanging information from two parents (vectors $\mathbf{p}^1$ and $\mathbf{p}^2$) obtained in the selection process by a roulette wheel approach (Leung et al., 2003). The recombination process to generate the offsprings (vectors $\mathbf{C}^1$, $\mathbf{C}^2$, $\mathbf{C}^3$ and $\mathbf{C}^4$) is

```
1  begin MGA
2      τ = 0;                    // τ: actual iteration
3      initialize population;
4      evaluate f(population);   // f(·): fitness function
5      while not termination condition do
6          τ = τ+1;
7          select a parent pair of chromosomes (p1 and p2) from population;
8          begin crossover operator
9              generate the offsprings C1, C2, C3 and C4 by Equations (32)-(35);
10             the offspring with the best fitness function is denoted C^best;
11         end
12         begin mutation operator with C^best
13             generate the mutated offsprings MC1, MC2 and MC3 by
               Equation (36);
14         end
15         // generate a new population
16         insert C^best in the population;
17         if random number < p_mut then
18             the one among MC1, MC2 and MC3 with largest fitness function
               value replaces the individual of the population with the smallest
               fitness function value;
19         else
20             if f(MC1) > smallest fitness function value in the population
               then
21                 MC1 replaces the individual of the population with the
                   smallest fitness function value;
22             end
23             if f(MC2) > smallest fitness function value in the population
               then
24                 MC2 replaces the individual of the population with the
                   smallest fitness function value;
25             end
26             if f(MC3) > smallest fitness function value in the population
               then
27                 MC3 replaces the individual of the population with the
                   smallest fitness function value;
28             end
29         end
30         evaluate f(population);
31     end
32 end
```

Fig. 3. The modified genetic algorithm procedure.

done by four crossover operators, which are defined by the following equations (Leung et al., 2003):

$$\mathbf{C}^1 = \frac{\mathbf{p}^1 + \mathbf{p}^2}{2}, \tag{32}$$

$$\mathbf{C}^2 = w(\mathbf{p}^1 \vee \mathbf{p}^2) + (1-w)\mathbf{p}_{\max}, \tag{33}$$

$$\mathbf{C}^3 = w(\mathbf{p}^1 \wedge \mathbf{p}^2) + (1-w)\mathbf{p}_{\min}, \tag{34}$$

$$\mathbf{C}^4 = \frac{w(\mathbf{p}^1 + \mathbf{p}^2) + (1-w)(\mathbf{p}_{\max} + \mathbf{p}_{\min})}{2}. \tag{35}$$

The symbol $w \in [0,1]$ (in this paper, we used 0.9) denotes the crossover weight (the closer w is to 1, the greater is the direct contribution from parents). The symbols $\mathbf{p}^1 \vee \mathbf{p}^2$ and $\mathbf{p}^1 \wedge \mathbf{p}^2$ denote the vectors whose elements are respectively the element-wise maximum and minimum of $\mathbf{p}^1$ and $\mathbf{p}^2$. The terms $\mathbf{p}_{\max}$ and $\mathbf{p}_{\min}$ denote the vectors with the maximum and minimum possible gene values, respectively. After the offspring generation by crossover operators, the son exhibiting the greatest fitness value will be chosen as the offspring generated by the crossover process. The resulting vector is denoted using the symbol $\mathbf{C}^{best}$, which replaces the individual of the population with the smallest fitness value.

After conclusion of the crossover process, three new mutated offsprings $\mathbf{MC}^1$, $\mathbf{MC}^2$, and $\mathbf{MC}^3$ are generated from $\mathbf{C}^{best}$ as follows (Leung et al., 2003):

$$\mathbf{MC}^j = \mathbf{C}^{best} + \mathbf{\Gamma}^j \Delta\mathbf{M}^j, \quad j = 1,2,3. \tag{36}$$

Here, the vectors $\Delta\mathbf{M}^j$ satisfy the inequalities $\mathbf{p}_{\min} \leqslant \mathbf{C}^{best} + \Delta\mathbf{M}^j \leqslant \mathbf{p}_{max}$ for $j = 1, 2, 3$. The vectors $\mathbf{\Gamma}^j$ have entries in $\{0,1\}$ and satisfy the following additional conditions: The vector $\mathbf{\Gamma}^1$ has only one randomly chosen non-zero entry, $\mathbf{\Gamma}^2$ represents a random binary vector, and $\mathbf{\Gamma}^3$ is the constant vector $\mathbf{1}$ (consisting only of ones).

The mutated offsprings are incorporated into the population according to the following scheme. We generate a random element r of the unit interval $[0,1]$ and compare it with 0.1. If $r < 0.1$ then the mutated offspring exhibiting the largest fitness replaces the individual of the current population that has the smallest fitness value. Otherwise, we perform the following steps for $j = 1, 2, 3$. If

the fitness value of $\mathbf{MC}^j$ exceeds the one of the least fit individual (the one that yields the smallest fitness value) of the current population then we substitute the latter with $\mathbf{MC}^j$.

Many performance evaluation criteria are found in literature. However, most of the existing literature frequently employ only one performance criterion for prediction evaluation. The Mean Squared Error (MSE) may be used to drive the prediction model in the training process, but it cannot be considered a conclusive measure for comparison of different prediction models (Clements & Hendry, 1993). For this reason, it is defined a fitness function, which is a combination of two well-known performance measures: MMRE and PRED(25) defined in the previous section. The proposed fitness function is given by

$$\text{Fitness function} = \frac{\text{PRED}(25)}{1 + \text{MMRE}}. \tag{37}$$

This fitness function was built to globally evaluate the prediction model, where the main idea is to maximize the PRED(25) metric and to minimize the MMRE metric. In this way, it will be used as performance indicator of the prediction model.

The idea used here is to conjugate a local search method (LMS) (to train the MRL perceptron) to a global search method (MGA) (to determine the minimum data dimensionality). While the MGA makes possible the testing of varied solutions (particular features of the data) in different areas of the solution space, the LMS acts on the initial solution (MRL perceptron) to produce a fine-tuned model. These processes offer the effective capacity to seek the most compact MRL perceptron (due to data dimensionality reduction), reducing computational cost and probability of model overfitting. Fig. 4 depicts the proposed method.

The termination conditions for the MGA are: (i) the maximum number of generations, (ii) the increase in the validation error or generalization loss (Gl) (Prechelt, 1994), and (iii) The decrease in the training error process training (Pt) (Prechelt, 1994).

Each individual of the MGA population is represented by chromosomes that have the following genes:

- **a**: MR operator coefficients;
- **b**: linear FIR operator coefficients;
- ρ: variable used to determine the rank r;
- λ: mixing parameter;
- **features**: a vector having size *MaxDim*, where each position has a real-valued codification, which is used to determine if a feature will be used ($features_i \geqslant 0$) or not ($features_i < 0$).

6. Experimental results

For all the experiments, the following initialization parameters were used: the MGA parameters used in the proposed MRLHD method are a maximum number of MGA generations, corresponding to 10^4, crossover weight $w = 0.9$ (used in the crossover operator), mutation probability equals to 0.1. The MR operator coefficients and the linear FIR operator coefficients (**a** and **b**, respectively) were normalized in the range $[-0.5, 0.5]$. The MRL perceptron parameters λ and ρ were in the range $[0, 1]$ and $[-MaxDim, MaxDim]$, respectively. The MRL perceptron was trained via the LMS algorithm using a convergence factor $\mu = 0.01$. The termination conditions for the LMS algorithm are the maximum number of epochs (10^4), the increase in the validation error or generalization loss ($Gl > 5\%$) and the decrease in the error of the process training ($Pt < 10^{-6}$).

Next, the simulation results involving the proposed model will be presented. In order to establish a fair performance comparison, results previously published in the literature with support vector regression (SVR) (Oliveira, 2006) (SVR-RBF and SVR-Linear), bagging (Braga et al., 2007), ga-based with SVR (Braga et al., 2008; Oliveira et al., 2010) (SVR-RBF and SVR-Linear) and morphological-rank-linear (MRL) (de Araújo et al., 2009, 2011) were examined

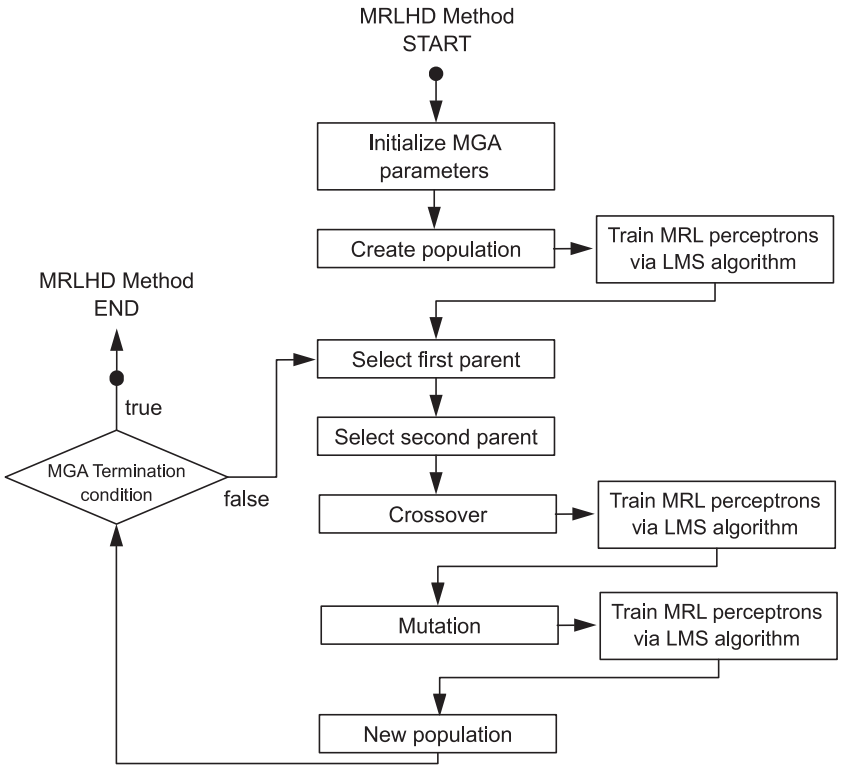

Fig. 4. The proposed method.

in the same context and under the same experimental conditions presented in Oliveira et al. (2010).

6.1. Nasa database

The Nasa software project data set (Oliveira, 2006) is used as a benchmarking for evaluation of the proposed model. It is composed of two independent features: Developed Lines (DL) and MEthodology (ME), and a dependent feature, which will be predicted/estimated, named Software Development Cost (SDC).

The DL feature is computed by the software's LOC (lines of code) value, which considers both implemented and re-used lines of code, giving a more realistic size measurement. It is worth mentioning that DL measure includes comments plus 20% of re-used lines (Oliveira, 2006). The ME feature is given according to methodologies applied in the software project development. Some of the factors included to compute the ME feature are: (i) the adoption of formal test plans, (ii) formal documentation, and (iii) formal training (Oliveira, 2006). Finally, the SDC is measured using the criterion man-by-months. It is possible to notice that SDC includes both programming and management time, being measured from the start of the design phase until acceptance testing, including programming, management, and support hours (Oliveira, 2006). For the prediction model built purpose, all Nasa data set was normalized to lie within the range [0,1]. Also, the Leave-One-Out Cross-Validation (LOOCV) procedure is used to estimate the generalization error, as Shin and Goel (2000). The LOOCV procedure is too simple. The data set is partitioned into k subsets (k represents the data set samples amount). The training set is defined by the $k-1$ subsets samples, and the test set is defined by the kth subset (used for prediction purpose). This process is performed k times, where for each experiment it uses a different software project as test set. Also, it is worth mentioning that the training set is partitioned into two sets (training set and validation set). The size of each set is given by the values proposed in Prechelt (1994), where it uses for training set (50% of the data) and for validation set (25% of the data).

Table 1 presents the Nasa software project database.

For the Nasa database, the proposed MRLHD model chose both features DL and ME as the best features to estimate the cost. Table 2 presents the experimental results (LOOCV) obtained by the models presented in literature (Braga et al., 2007, 2008; de Araújo et al., 2009; Oliveira, 2006), as well as those achieved by the proposed MRLHD model.

According to Table 2, it is possible to notice that the best model found in literature is the MRL (fitness function ~ 87). Regarding the proposed MRLHD model, it outperforms (in terms of MMRE and fitness function), having better performance (0.0738 and 87.9493, respectively), the other analyzed models. The proposed MRLHD model have 8.32% and 0.62% of improvement in MMRE and fitness function, respectively, over the results achieved using the MRL model (best model found in literature so far). However, regarding PRED(25) metric, the proposed MRLHD obtained the same performance of the MRL model.

Table 1
Nasa software project database.

Software project	DL	ME	SDC
1	90.2	30	115.8
2	46.2	20	96.0
3	46.5	19	79.0
4	54.5	20	90.8
5	31.1	35	39.6
6	67.5	29	98.4
7	12.8	26	18.9
8	10.5	34	10.3
9	21.5	31	28.5
10	3.1	26	7.0
11	4.2	19	9.0
12	7.8	31	7.3
13	2.1	28	5.0
14	5.0	29	8.4
15	78.6	35	98.7
16	9.7	27	15.6
17	12.5	27	23.9
18	100.8	34	138.3

Table 2
Prediction results for the Nasa database.

Model	PRED(25)	MMRE	Fitness function
SVR-Linear (Oliveira, 2006)	88.89	0.1650	76.3004
SVR-RBF (Oliveira, 2006)	83.33	0.1780	70.7385
Bagging (Braga et al., 2007)	94.44	0.1771	80.2311
GA-based with SVR-RBF (Braga et al., 2008)	94.44	0.1624	81.2456
GA-based with SVR-Linear (Braga et al., 2008)	94.44	0.1636	81.1619
MRL (de Araújo et al., 2009)	94.44	0.0805	87.4108
MRLHD	94.44	0.0738	87.9493

6.2. Desharnais database

The Desharnais database (Oliveira et al., 2010) consists of 81 software projects described by eleven features, nine independent and two dependent. The first dependent feature is the cost. The second dependent feature (length) was ignored because the goal of this work is to estimate the software development cost. The nine independent features of this database are TeamExp (team experience), ManagerExp (manager experience), YearEnd (year project ended), Transactions (count of basic logical transactions in the system – function points), Entities (Entities is the number of entities in the systems data model – function points), PointsNonAdjust (This is Transactions + Entities – function points), Adjustment (Function point complexity adjustment factor – Total Processing Complexity), PointsAdjust (This is the function points adjusted by the *adjustment factor* = 0.65 + (0.01 ∗ *PointsNonAdjust*).) and Language (programming language). The database was normalized to lie within the range [0,1] and divided in three sets according to Prechelt (1994): training set (50% of the points), validation set (25% of the points) and test set (25% of the points).

For the Desharnais database, the proposed MRLHD model chose the features ManagerExp, Transactions, Entities and PointsAdjust as the best features to estimate the cost. Table 3 presents the experimental results (of the test set) obtained by the models presented in literature (Braga et al., 2007, 2008; de Araújo et al., 2009; Oliveira, 2006), as well as those achieved by the proposed MRLHD model.

According to Table 3, it is possible to notice that the best model found in literature is the MRL (fitness function ~ 74). Regarding the proposed MRLHD model, it outperforms (in terms of PRED(25), MMRE and fitness function), having much better performance (90.00, 0.0981 and 81.9597, respectively), the other analyzed models. The proposed MRLHD model have 5.88%, 34.99% and 10.97% of improvement in PRED(25), MMRE and fitness function, respectively, over the results achieved using the MRL model (best model found in literature so far).

6.3. Cocomo database

The Cocomo database (Oliveira et al., 2010) consists of 63 software projects described by twenty-two features, twenty

Table 3
Prediction results for the Desharnais database.

Model	PRED(25)	MMRE	Fitness function
SVR-Linear (Oliveira, 2006)	55.00	0.4829	37.0895
SVR-RBF (Oliveira, 2006)	60.00	0.4543	41.2570
Bagging (Braga et al., 2007)	65.00	0.4076	46.1779
GA-based with SVR-RBF (Braga et al., 2008)	80.00	0.3302	60.1413
GA-based with SVR-Linear (Braga et al., 2008)	80.00	0.3154	60.8180
MRL (de Araújo et al., 2009)	85.00	0.1509	73.8552
MRLHD	90.00	0.0981	81.9597

Table 4
Prediction results for the Cocomo database.

Model	PRED(25)	MMRE	Fitness function
SVR-Linear (Oliveira, 2006)	81.82	0.1573	70.6990
SVR-RBF (Oliveira, 2006)	72.73	0.1802	61.6251
Bagging (Braga et al., 2007)	72.73	0.1754	61.8768
GA-based with SVR-RBF (Braga et al., 2008)	72.73	0.1729	62.0087
GA-based with SVR-Linear (Braga et al., 2008)	81.82	0.1481	71.2656
MRL (de Araújo et al., 2009)	81.82	0.1436	71.5460
MRLHD	90.90	0.1298	80.4567

independent and one dependent. The dependent feature is ACTMM (the estimated software development cost in by man per hour). The twenty independent features of this database are LANG (programming language), RELY (software reliability), DATA (database size), CPLX (product complexity), TIME (time execution constraint), STOR (main storage constraint), VIRT (virtual machine volatility), TURN (runtime machine), ACAP (analyst capacity), AEXP (applications experience), PCAP (programmer capacity), VEXP (virtual machine experience), LEXP (language programming experience), MODP (modern programming practices), PLATFORM (development platform), TOOL (software tools), SCHED (schedule development), RVOL (requirements volatility), MODE (development mode) and ADJKDSI (adjusted code lines number). The database was normalized to lie within the range [0,1] and divided in three sets according to Prechelt (Prechelt, 1994): training set (50% of the points), validation set (25% of the points) and test set (25% of the points).

For the Cocomo database, the proposed MRLHD model chose the features LANG, CPLX, TIME, ACAP, AEXP, PCAP, MODP, PLATFORM, SCHED and RVOL as the best features to estimate the cost. Table 4 presents the experimental results (of the test set) obtained by the models presented in literature (Braga et al., 2007, 2008; de Araújo et al., 2009; Oliveira, 2006), as well as those achieved by the proposed MRLHD model.

According to Table 4, it is possible to notice that the best model found in literature is the MRL (fitness function $\sim$ 72). Regarding the proposed MRLHD model, it outperforms (in terms of PRED(25), MMRE and fitness), having much better performance (90.90, 0.1298 and 80.4567, respectively), the analyzed models. The proposed MRLHD model have 11.10%, 9.61% and 12.45% of improvement in PRED(25), MMRE and fitness, respectively, over the results achieved using the MRL model (best model found in literature so far).

6.4. Albrecht database

The Albrecht software project data set (Oliveira et al., 2010) is used as a benchmarking for evaluation of the proposed model. It is composed of six independent features: Input Count (IC), Output Count (OC), Query Count (QC), File Count (FC), Function Points (FP) and Lines of Source Code (SLOC), and a dependent feature, which will be predicted/estimated, named Software Development Cost (SDC). The SDC is recorded in 1000 person hours and is the targeting feature of cost estimation.

Table 5
Albrecht software project database.

Software project	IC	OC	QC	FC	FP	SLOC	SDC
1	25	28	22	4	15	500	3.6
2	70	27	12	0	20	428	11.1
3	69	112	39	21	110	1572	61.2
4	61	68	11	0	24	694	11.8
5	28	38	9	24	96	512	15.8
6	28	41	11	16	24	417	7.5
7	41	27	5	29	57	512	10.8
8	13	19	23	0	28	283	10
9	48	66	50	13	94	1235	38.1
10	42	57	5	12	40	606	18.3
11	34	14	5	0	35	205	8
12	25	150	60	75	130	1750	102.4
13	7	12	8	13	40	209	4.1
14	15	15	3	6	3	199	0.5
15	193	98	36	70	318	1902	105.2
16	40	60	12	20	54	759	21.1
17	33	17	5	8	22	224	2.9
18	45	64	16	14	48	680	12.9
19	27	20	6	24	52	400	8.9
20	17	17	5	15	30	289	4.9
21	40	60	15	20	93	794	19
22	10	69	9	1	62	431	28.8
23	12	15	15	0	29	260	6.1
24	43	40	35	20	42	682	12

For the prediction model built purpose, all Albrecht data set was normalized to lie within the range [0,1]. Also, the Leave-One-Out Cross-Validation (LOOCV) procedure is used to estimate the generalization error, as Oliveira et al. (2010). Also, it is worth mentioning that the training set is partitioned into two sets (training set and validation set). The size of each set is given by the values proposed in Prechelt (1994), where it uses for training set (50% of the data) and for validation set (25% of the data).

Table 5 presents the Albrecht software project database.

For the Albrecht database, the proposed MRLHD model chose the features IC, OC, FP and SLOC as the best features to estimate the cost. Table 6 presents the experimental results (LOOCV) obtained by the models presented in literature (Braga et al., 2007, 2008; de Araújo et al., 2009; Oliveira, 2006; Oliveira et al., 2010), as well as those achieved by the proposed MRLHD model.

It is possible to verify that, according to Table 6, the best model found in literature is the MRL (fitness function $\sim$ 50). Regarding the proposed MRLHD model, it outperforms (in terms of PRED(25), MMRE and fitness function), having better performance (75.00, 0.3810 and 54.3085, respectively), the other analyzed models. The proposed MRLHD model have 5.89%, 6.78% and 8.01% of improvement in PRED(25), MMRE and fitness function, respectively, over the results achieved using the MRL model (best model found in literature so far).

6.5. Kemerer database

The Kemerer software project data set (Oliveira et al., 2010) is used as a benchmarking for evaluation of the proposed model. It is composed of six independent features: Language (LANG), Hardware (HARD), Duration (DUR), KSLOC, Adjusted Function Points (ADJFP), Raw Function Points (RAWFP), and a dependent feature, which will be predicted/estimated, named Software Development Cost (SDC).

Table 6
Prediction results for the Albrecht database.

Model	PRED(25)	MMRE	Fitness function
SVR-Linear (Oliveira, 2006)	58.33	0.6719	34.8884
SVR-RBF (Oliveira, 2006)	66.66	0.5072	44.2277
Bagging (Braga et al., 2007)	70.83	0.4178	49.9577
GA-based with SVR-RBF (Oliveira et al., 2010)	70.42	0.4465	48.6830
GA-based with SVR-Linear (Oliveira et al., 2010)	56.25	0.6628	33.8285
MRL (de Araújo et al., 2009)	70.83	0.4087	50.2804
MRLHD	75.00	0.3810	54.3085

Table 7
Kemerer software project database.

Software project	LANG	HARD	DUR	KSLOC	ADJFP	RAWFP	SDC
1	1	1	17	253.6	1217.1	1010	5.66
2	1	2	7	40.5	507.3	457	4.41
3	1	3	15	450	2306.8	2284	7.01
4	1	1	18	214.4	788.5	881	4.46
5	1	2	13	449.9	1337.6	1583	5.82
6	1	4	5	50	421.3	411	4.43
7	2	4	5	43	99.9	97	3.14
8	1	2	11	200	993	998	4.87
9	1	1	14	289	1592.9	1554	4.75
10	1	1	5	39	240	250	4.28
11	1	1	13	254.2	1611	1603	5.56
12	1	5	31	128.6	789	724	5.44
13	1	6	20	161.4	690.9	705	5.06
14	1	1	26	164.8	1347.5	1375	5.51
15	3	1	14	60.2	1044.3	976	4.25

Table 8
Prediction results for the Kemerer database.

Model	PRED(25)	MMRE	Fitness function
SVR-Linear (Oliveira, 2006)	60.00	0.4608	41.0734
SVR-RBF (Oliveira, 2006)	60.00	0.4439	41.5541
Bagging (Braga et al., 2007)	60.00	0.4297	41.9668
GA-based with SVR-RBF (Oliveira et al., 2010)	66.67	0.3695	48.6820
GA-based with SVR-Linear (Oliveira et al., 2010)	60.00	0.4373	41.7449
MRL (de Araújo et al., 2009)	66.67	0.3014	51.2294
MRLHD	73.33	0.2779	57.3832

For the prediction model built purpose, all Kemerer data set was normalized to lie within the range [0,1]. Also, the Leave-One-Out Cross-Validation (LOOCV) procedure is used to estimate the generalization error, as in Oliveira et al. (2010). Also, it is worth mentioning that the training set is partitioned into two sets (training set and validation set). The size of each set is given by the values proposed in Prechelt (1994), where it uses for training set (50% of the data) and for validation set (25% of the data).

Table 7 presents the Kemerer software project database.

For the Kemerer database, the proposed MRLHD model chose the features DUR, KSLOC, ADJFP and RAWFP as the best features to estimate the cost. Table 8 presents the experimental results (LOOCV) obtained by the models presented in literature (Braga et al., 2007, 2008; de Araújo et al., 2009; Oliveira, 2006; Oliveira et al., 2010), as well as those achieved by the proposed MRLHD model.

Note that, according to Table 8, the best model found in literature is the MRL (fitness function ~ 51). Regarding the proposed MRLHD model, it outperforms (in terms of PRED(25), MMRE and fitness function), having better performance (73.33, 0.2779 and 57.3832, respectively), the other analyzed models. The proposed MRLHD model have 9.99%, 7.80% and 12.01% of improvement in PRED(25), MMRE and fitness function, respectively, over the results achieved using the MRL model (best model found in literature so far).

6.6. KotenGray database

The KotenGray software project data set (Oliveira et al., 2010) is used as a benchmarking for evaluation of the proposed model. It is composed of seven independent features: Productivity (PROD), FRMNUM, RPTNUM, GPHNUM, ENTFRM, ENTRPT and ENTGPH, and a dependent feature, which will be predicted/estimated, named Software Development Cost (SDC).

For the prediction model built purpose, all KotenGray data set was normalized to lie within the range [0,1]. Also, the Leave-One-Out Cross-Validation (LOOCV) procedure is used to estimate the generalization error, as in Oliveira et al. (2010). Also, it is worth mentioning that the training set is partitioned into two sets (training set and validation set). The size of each set is given by the values proposed in Prechelt (1994), where it uses for training set (50% of the data) and for validation set (25% of the data).

Table 9 presents the KotenGray software project database.

For the KotenGray database, the proposed MRLHD model chose the features PROD, FRMNUM, RPTNUM, GPHNUM and ENTGPH as the best features to estimate the cost. Table 10 presents the experimental results (LOOCV) obtained by the models presented in literature (Braga et al., 2007, 2008; de Araújo et al., 2009; Oliveira, 2006; Oliveira et al., 2010), as well as those achieved by the proposed MRLHD model.

According to Table 10, it is possible to verify that the best model found in literature is the MRL (fitness function ~ 87). Regarding the proposed MRLHD model, it outperforms (in terms MMRE and fitness function), having better performance (0.0689 and 88.0531, respectively), the other analyzed models. The proposed MRLHD model have 2.96% and 0.2% of improvement in MMRE and fitness function, respectively, over the results achieved using the MRL model (best model found in literature so far). However, regarding PRED(25) metric, the proposed MRLHD obtained the same performance of the MRL model.

7. Conclusion

In this paper, a hybrid methodology to design morphological-rank-linear (MRL) perceptrons, was proposed to solve the software development cost estimation (SDCE) problem. In this methodology, we used a modified genetic algorithm (MGA) to optimize the parameters of the MRL perceptron and to select an optimal input feature subset of the used data sets, aiming at a higher accuracy level for the SDCE. Furthermore, for each individual of MGA, it is used a gradient steepest descent method (using a Least Mean Squares (LMS) algorithm with a systematic approach to overcome the problem of nondifferentiability of the morphological-rank operator) to optimize the MRL perceptron parameters supplied by MGA.

Two different metrics (PRED(25) and MMRE) were used to measure the performance of the proposed MRLHID model. A fitness function was designed with these two well-known statistic error measures in order to create a global indicator of the prediction model performance, where the main idea was to maximize the PRED(25) metric and to minimize the MMRE metric. The justification of the inclusion of this metrics to evaluate the proposed method is that most of the existing literature frequently employ the

Table 9
KotenGray software project database.

Software project	PROD	FRMNUM	RPTNUM	GPHNUM	ENTFRM	ENTRPT	ENTGPH	SDC
1	65.833	17	5	7	37	14	18	359.25
2	72.292	16	9	0	44	25	0	473.25
3	59.375	7	11	0	11	28	0	314
4	60.625	12	6	0	38	25	0	354
5	59.167	8	6	0	26	23	0	258.2
6	56.25	13	10	0	26	44	0	379
7	49.583	14	8	0	34	32	0	432.25
8	73.958	8	10	0	36	37	0	313.5
9	55.417	14	2	1	62	5	3	553.8
10	60.417	11	10	2	19	39	2	337.75
11	54.792	16	4	0	19	17	0	296.25
12	42.5	20	5	0	43	29	0	568.65
13	48.125	10	2	0	25	7	0	354.5
14	46.042	18	2	0	47	9	0	531.35
15	64.583	8	9	0	27	33	0	459.25
16	50.417	11	8	0	32	32	0	428.5
17	59.375	10	8	0	34	26	0	353.5

Table 10
Prediction results for the KotenGray database.

Model	PRED(25)	MMRE	Fitness function
SVR-Linear (Oliveira, 2006)	88.24	0.1133	79.2599
SVR-RBF (Oliveira, 2006)	88.24	0.1108	79.4382
Bagging (Braga et al., 2007)	94.12	0.1001	85.5559
GA-based with SVR-RBF (Oliveira et al., 2010)	94.12	0.0947	85.9779
GA-based with SVR-Linear (Oliveira et al., 2010)	94.12	0.0895	86.3883
MRL (de Araújo et al., 2009)	94.12	0.0710	87.8805
MRLHD	94.12	0.0689	88.0531

Mean Squared Error (MSE) for estimation evaluation, however it may be used to drive the model in the training process, but it cannot be considered a conclusive measure for comparison of different models.

An experimental validation of the method was carried out using Nasa, Desharnais, Cocomo, Albrecht, Kemerer and KotenGray databases, showing the robustness of the proposed MRLHID model through a comparison, according to two performance measures and a fitness function, of previous results found in the literature (SVR-Linear, SVR-RBF, Bagging, GA-based with SVR-Linear, GA-based with SVR-RBF and MRL). This experimental investigation indicates a better, more consistent global performance of the proposed MRLHID model, having around 0.6% of improvement (Nasa database), around 11% of improvement (Desharnais database), around 12% of improvement (Cocomo database), around 8% of improvement (Albrecht database), around 12% of improvement (Kemerer database) and around 0.2% of improvement (KotenGray database) regarding the best result reported in the literature.

It is possible to notice that the main advantages of the proposed MRLHID model, apart from its superior predictive performance compared to all analyzed models, are (i) it has both linear and nonlinear components (it means that the model can use a distinct percentage of the linear and nonlinear components according to the linearity or nonlinearity of the problem), and (2) it is quite attractive due to its simpler computational complexity, according to analysis presented in (de Araújo & Ferreira, 2009).

Future works will consider the development of further studies in order to formalize properties of the proposed model, as well as the use of other data sets and in other software engineering classification and regression problems such as fault and maintainability prediction.

Acknowledgment

This work was partially supported by the National Institute of Science and Technology for Software Engineering (INES), funded by CNPq and FACEPE, Grants 573964/2008-4 and APQ-1037-1.03/08.

References

Braga, P. L., de Oliveira, A. L. I., & Meira, S. R. L. (2008). A ga-based feature selection and parameters optimization for support vector regression applied to software effort estimation. In *SAC '08: Proceedings of the 2008 ACM symposium on applied computing* (pp. 1788–1792). New York, NY, USA: ACM.

Braga, P. L., Oliveira, A. L. I., Ribeiro, G. H. T., & Meira, S. R. L. (2007). Software effort estimation using machine learning techniques with robust confidence intervals. In *IEEE international conference on tools with artificial intelligence (ICTAI)* (pp. 1595–1600).

Clements, M. P., & Hendry, D. F. (1993). On the limitations of comparing mean square forecast errors. *Journal of Forecasting, 12*(8), 617–637.

de Araújo, A. R. (2007). Swarm-based hybrid intelligent forecasting method for financial time series prediction. *Learning and Nonlinear Models, 5*(2), 137–154.

de Araújo, A. R. (2010a). Hybrid intelligent methodology to design translation invariant morphological operators for brazilian stock market prediction. *Neural Networks, 23*(10), 1238–1251.

de Araújo, A. R. (2010b). A hybrid intelligent morphological approach for stock market forecasting. *Neural Processing Letters, 31*(3), 195–217.

de Araújo, A. R. (2010c). Swarm-based translation-invariant morphological method for financial time series forecasting. *Information Sciences, 180*(24), 4784–4805.

de Araújo, A. R. (2011). Translation invariant morphological time-lag added evolutionary forecasting method for stock market prediction. *Expert Systems with Applications, 38*(3), 2835–2848.

de Araújo, A. R., & Ferreira, T. A. E. (in press). A morphological-rank-linear evolutionary method for stock market prediction. *Information Sciences*.

de Araújo, A. R., & Sussner, P. (2010). An increasing hybrid morphological-linear perceptron with pseudo-gradient-based learning and phase adjustment for financial time series prediction. In *IEEE international joint conference on neural networks*. IEEE.

de Araújo, A. R., Madeiro, F., de Sousa, R.P., Pessoa, L. F. C., & Ferreira, T. A. E. (2006). An evolutionary morphological approach for financial time series forecasting. In *Proceedings of the IEEE congress on evolutionary computation* (pp. 2467–2474).

de Araújo, A. R., de Oliveira, A. L. I., & Soares, S. C. B. (2009). A morphological-rank-linear approach for software development cost estimation. In *IEEE international conference on tools with artificial intelligence*. IEEE.

de Araújo, A. R., & Ferreira, T. A. E. (2009). An intelligent hybrid morphological-rank-linear method for financial time series prediction. *Neurocomputing, 72*(10–12), 2507–2524.

de Araújo, A. R. (in press). A robust automatic phase-adjustment method for financial forecasting. *Knowledge-Based Systems*.

de Araújo, R. A., Oliveira, A. L. I., & Soares, S. C. B. (2011). A shift-invariant morphological system for software development cost estimation. *Expert Systems with Applications, 38*(4), 4162–4168.

Elish, M. O. (2009). Improved estimation of software project effort using multiple additive regression trees. *Expert Systems with Applications, 36*(7), 10774–10778.

Group, S. (2004). *Project success rates improved over 10 years*. Tech. rep., <http//www.softwaremag.com/L.cfm?Doc=newsletter/2004-01-15/Standish>.

Heijmans, H. J. A. M. (1994). *Morphological image operators*. New York, NY: Academic Press.

Leung, F. H. F., Lam, H. K., Ling, S. H., & Tam, P. K. S. (2003). Tuning of the structure and parameters of the neural network using an improved genetic algorithm. *IEEE Transactions on Neural Networks, 14*(1), 79–88.

Li, Q., Wang, Q., Ye, Y., & Li, M. (2008). Reducing biases in individual software effort estimations: A combining approach. In *ESEM '08: Proceedings of the second ACM–IEEE international symposium on empirical software engineering and measurement* (pp. 223–232). New York, NY, USA: ACM.

Li, Y. F., Xie, M., & Goh, T. N. (2009). A study of the non-linear adjustment for analogy based software cost estimation. *Empirical Software Engineering, 14*(6), 603–643.

Maragos, P. (1989). A representation theory for morphological image and signal processing. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 11*, 586–599.

Matheron, G. (1975). *Random sets and integral geometry*. New York: Wiley.

Oliveira, A. L. I. (2006). Estimation of software project effort with support vector regression. *Neurocomputing, 69*(13–15), 1749–1753.

Oliveira, A. L. I., Braga, P. L., Lima, R. M. F., & Cornelio, M. L. (2010). Ga-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation. *Information and Software Technology, 52*(11), 1155–1166.

Pessoa, L. F. C., & Maragos, P. (1998). MRL-filters: A general class of nonlinear systems and their optimal design for image processing. *IEEE Transactions on Image Processing, 7*, 966–978.

Pessoa, L. F. C., & Maragos, P. (2000). Neural networks with hybrid morphological rank linear nodes: A unifying framework with applications to handwritten character recognition. *Pattern Recognition, 33*, 945–960.

Prechelt, L. (1994). *Proben1: A set of neural network benchmark problems and benchmarking rules*. Tech. Rep. 21/94.

Ronse, C. (1990). Why mathematical morphology needs complete lattices. *Signal Processing, 21*(2), 129–154.

Salembier, P. (1992a). Adaptive rank order based filters. *Signal Processing, 27*(1), 1–25.

Salembier, P. (1992b). Structuring element adaptation for morphological filters. *Journal for Visual Communication and Image Representation, 3*(2), 115–136.

Serra, J. (1982). *Image analysis and mathematical morphology*. London: Academic Press.

Serra, J. (1988). *Image analysis and mathematical morphology. Theoretical advances* (Vol. 2). New York: Academic Press.

Shin, M., & Goel, A. L. (2000). Empirical data modeling in software engineering using radial basis functions. *IEEE Transactions on Software Engineering, 26*(6), 567–576.

Sussner, P., & Esmi, E. L., (2009). An introduction to morphological perceptrons with competitive learning. In *Proceedings of the international joint conference on neural networks, Atlanta, GA* (pp. 3024–3031).

Won, Y., Gader, P. D., & Coffield, P. (1997). Morphological shared-weight networks with applications to automatic target recognition. *IEEE Transactions on Neural Networks, 8*(5), 1195–1203.