Swarm and Evolutionary Computation 38 (2018) 158–172

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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

DABE: Differential evolution in analogy-based software development effort estimation

Tirimula Rao Benala[a,*], Rajib Mall[b]

[a] Department of Information Technology, JNTUK, University College of Engineering, Vizianagzaram , Andhra Pradesh 535003, India
[b] Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, West Bengal 721302, India

ARTICLE INFO

Keywords:
Software development effort estimation
Differential evolution
Analogy-based estimation
Feature weight optimization

ABSTRACT

Several feature weight optimization techniques have been proposed for similarity functions in analogy-based estimation (ABE); however, no consensus regarding the method and settings suitable for producing accurate estimates has been reached. We investigate the effectiveness of differential evolution (DE) algorithm, for optimizing the feature weights of similarity functions of ABE by applying five successful mutation strategies. We have named this empirical analysis as DE in analogy-based software development effort estimation (DABE). We have conducted extensive simulation study on the PROMISE repository test suite to estimate the effectiveness of our proposed DABE technique. We find significant improvements in predictive performance of our DABE technique over ABE, particle swarm optimization-based feature weight optimization in ABE, genetic algorithm-based feature weight optimization in ABE, self-adaptive DE-based feature weight optimization ABE, adaptive differential evolution with optional external archive-based feature weight optimization ABE, functional link artificial neural network,artificial neural network with back propagation learning based software development effort estimation (SDEE), and radial basis function-based SDEE.

1. Introduction

Software project managers require reliable techniques for performing activities, such as feasibility studies, project bidding, planning, resource allocation, and cost estimation, at the initial stages of software development. Over the past two decades, substantial research efforts have been directed toward accurate cost prediction for more effective schedule and product quality management. Analogy-based estimation (ABE) introduced by Sternberg [32], is a case-based reasoning (CBR) model, which mimics the instinctive human decision-making procedure. ABE was partially inspired by the strong relationship between expert judgment approach [125] and conventional use of analogies in CBR [48]. In 1997, Shepperd and Schofield reported ABE in SDEE, which since then has become a popular alternative to SDEE. The accuracy of effort prediction using ABE strongly relies on four key components—similarity function, historical database, the number of closest analogs, and solution function: Initially, one or more past projects that are similar to the project under development are retrieved from the historical project base (PB) by using a similarity function. A heuristic function then predicts the effort for a new project by using effort information of retrieved projects.

In conventional ABE, the Euclidean distance, Manhattan distance, or Minkowski distance are considered as the similarity function for calculating the distance between two projects. These methods treat each project as a point of the high-dimensional Euclidean space. Therefore, the features in each project become independent of each other, thus having similar influences on cost estimation. This phenomenon contradicts the real-life scenario; where the project features are often depend on each other. Tosun et al. [69] suggested that various features have different levels of influence on the accuracy of estimation. This results in estimation errors. Therefore, if we model the unequal influence of various attributes, our estimation would become more precise. The weighted distance models the real-life software projects more accurately. Several researchers have proposed feature selection methods (Bener et al., 2007) and feature weight assignment methods using Euclidean distance ([24]; Mendes et al., 2003; [23]), fuzzy logic [61], rough set analysis (Li and Ruhe 2006, [85]), and genetic algorithms (GAs) [25].

Researchers have also proposed several prediction methods based on machine learning, swarm intelligence, and soft computing for accurate SDEE. Swarm intelligence techniques could be a promising alternative to conventional software effort prediction methods for achieving accurate prediction ([37,38,92], Bardsiri et al., 2015; [147]). Li et al. [37] reported that GAs may be appropriate to search

* Corresponding author.
E-mail addresses: btirimula.it@jntukucev.ac.in (T.R. Benala), rajib@cse.iitkgp.ernet.in (R. Mall).

http://dx.doi.org/10.1016/j.swevo.2017.07.009
Received 12 May 2016; Received in revised form 29 March 2017; Accepted 10 July 2017
Available online 13 July 2017
2210-6502/ © 2017 Elsevier B.V. All rights reserved.

for a subset of instances yielding the most favorable effort. Bardsiri et al. [38] applied particle swarm optimization for optimal feature weighting in similarity function. Slowik [8] reported that because of its inherent properties, GA could not be used for solving real-world optimization problems. At the first International Contest on Evolutionary Optimization, held in Nagoya, Japan, the differential evolution (DE) algorithm, initially advocated by Storn and Price [6], was considered the most suitable genetic type of algorithm [56]. In the classical DE algorithm, differential operator is incorporated in place of conventional crossover and mutation operators in GA. The DE algorithm has the following inherent benefits over the traditional GA: more straightforward implementation in any computer language, increased cost-effective memory utilization, lower computational complexity, and decreased computational effort (faster convergence) [58,93,95,113]. In particular, the DE algorithm is considerably successful in nonlinear constraint optimization problems and is also suitable for optimizing multimodal problems [94]. The DE algorithm is considered a successful alternative for solving complex real-world optimization problems.

The DE algorithm is succinct, can be concisely coded, and requires few control parameters. It has been extensively studied in domains, such as nonlinear system identification, pattern recognition, medical diagnosis, and computational finance [56,111]. In this context, we explore the usage of the DE algorithm in analogy-based SDEE. We propose a novel ABE technique by applying various popular mutation schema of the DE algorithm for feature weight optimization of similarity function (five schema, mentioned in Section 2.2.1) by using each scheme to explore feature weight search. Each mutation scheme has its own characteristics regarding search space exploration versus exploitation, as described in Section 2.2.1. To the best of our knowledge,in contrast to that in other domains,only one study (Bardsiri et al., 2015) has reported the exploitation of the characteristics of popular mutation strategies of DE in the SDEE domain.

Rest of this paper is organized as follows: In Section 2, we present a concise critique on the background of this study, followed by a summary of the related work in Section 3. Next, we discuss the proposed differential evolution in analogy-based SDEE (DABE) framework in Section 4 and present real-world datasets and our experimental setup in Section 5. In Section 6,we report the empirical analysis of PROMISE repository data sets. In Section 7,we state the statistical performance evaluation of the proposed model on the related models. Finally, in Section 8, we present our study conclusion.

2. Background

In this section, we first review a few background concepts and terminologies. In Section 2.1, we discuss analogy-based SDEE. Next, we mention DE as a new stochastic population-based computing paradigm and present the DE pseudo code in Section 2.2.

2.1. ABE

Here, we report an ABE technique. We also describe similarity function, k-nearest neighbors, and solution function in Sections 2.1.1–2.1.3, respectively.

ABE is a framework of CBR with cases described as a generalization of phenomena constrained in time and space [23]. The ABE model functions with pre-assumptions in line with the CBR taxonomy. The historical project base (PB) contains similar projects with relevant features. Any new project presented to the ABE model for estimating effort is considered a new case. Previous projects stored in historical PB are called case repository. The process of analogy-based effort estimation by consists of the following three steps:

1. The new project to be estimated is presented by a set of relevant features.
2. Conforming to a predefined similarity function, one or more similar projects are retrieved from the historical project dataset. The commonly used similarity functions are Euclidean distance,Manhattan distance, and Minkowski distance.
3. Finally, the effort of the new project is predicted by combining the effort values of similar projects (typically by calculating their mean, median, or weighted mean).

2.1.1. Similarity function

The similarity function, which assesses the degree of similarity between the new and past projects, is the core of ABE model. The choice of similarity measure influences the selection of k-nearest neighbors. The following is the general form of the similarity function [85]:

$$sim(p, p') = f((Lsim(f_1, f_1'), Lsim(f_2, f_2'), \ldots, Lsim(f_n, f_n')) \quad (1)$$

where p and p' represents the two respective projects (new project and old project in historical PB, f_i and f_i'denotes denote the ith feature values of the corresponding projects, n indicates the total number of features in each project, and $Lsim(.)$ function accounts for local similarity measure between two corresponding features of the two respective projects. $Lsim(.)$ and $f(.)$ functions defines the structure of the generalized similarity function. Various types of similarity functions exist, namely Euclidean similarity (ES), Manhattan distance-based similarity (MS), Minkowski similarity, Mahalanobis distance-based similarity [84], maximum distance-based similarity [34], and rank mean similarity (Walkerden and Jeffery 1997). Among these, ES and MS are most commonly used in the SDEE domain. The project similarities are positioned on the Euclidean distance between two projects as depicted in Eq. (2).

$$sim(p, p') = 1/\left[\sqrt{\sum_{i=1}^{n} w_i Dis(f_i f_i')} + \delta\right] \delta = 0.0001 \quad (2)$$

$$Dis(f_i f_i') = \begin{cases} (f_i - f_i')^2 iff_i \; and f_i' \; are numerical or ordinal \\ 1 iff_i \; and f_i' \; are nominal and f_i = f_i' \\ 0 iff_i \; and f_i' \; are nominal and f_i \neq f_i' \end{cases}$$

where w_i is the weight of ith feature with values between 0 and 1; $\delta = 0.0001$ is a small positive constant intentionally introduced to avoid denominator being zero.

MS, also known as city block distance similarity, is a variant of ES and is defined as the summation of the absolute difference between a pair of features of two respective projects. Eq. (3) represents the MS function:

$$sim(p, p') = 1/\left[\sum_{i=1}^{n} w_i Dis(f_i f_i') + \delta\right] \delta = 0.0001 \quad (3)$$

$$Dis(f_i f_i') = \begin{cases} |f_i - f_i'| iff_i \; and f_i' \; are numerical or ordinal \\ 1 iff_i \; and f_i' \; are nominal and f_i = f_i' \\ 0 iff_i \; and f_i' \; are nominal and f_i \neq f_i' \end{cases}$$

2.1.2. K-nearest neighbors

A crucial parameter influencing the prediction accuracy of a new project is the number of k-nearest neighbors. A few studies have suggested that k should be predefined at fixed values ([33]; Leung, 2002; Mendes et al., 2003; Jorgensen et al., 2003; [24,36,66]). However, some recent studies (Li and Ruhe, 2008a, 2008b) have reported that dynamic k value can be obtained by selecting a distance threshold and that smaller k values are suitable for non-homogeneous datasets. Here, we assumed k values to be 1–5 for our experiments.

2.1.3. Solution functions

The solution function incorporates certain statistical evaluation techniques for the final effort prediction of a new project. We used the

following evaluation methods as the basis for the solution function in our study: the closet analogy (most similar project) [33], unweighted mean [23], the median [34], and the inverse distance weighted mean [51]. The mean—a classical measure of central tendency—is the average of the software efforts (k values) of the most similar projects, where k > 1. The median—another standard measure of central tendency—is the median of the software efforts (k values) of the most similar projects, where k > 2. The median is a more robust statistics than is the mean because the median is insensitive toward outliers, the number of which increases with increase in the number of the most comparable projects [34]. The inverse distance weighted mean (IWM) states that more similar projects will have more importance than less similar ones [51]. The weighted mean formula is depicted in Eq. (4):

$$\hat{C}_p = \sum_{k=1}^{n} \frac{Sim(P, P_k)}{\sum_{i=1}^{n} Sim(P, P_k)} C_{p_k} \tag{4}$$

Where p denotes the project being estimated, P_k represents the kth most similar project, $Sim(P, P_k)$ is the similarity between projects P andP_k, C_{p_k} is the cost of the most similar project P_k, and k is the total number of most similar projects.

2.2. DE algorithm

The DE algorithm [100,5,7,99] is a population-based stochastic meta-heuristic algorithm, which aims to evolve a population of NP D-dimensional parameter vector or chromosome, called individuals, which encode the candidate solutions; $x_{i,j}^G i = 1, \ldots, NP$, toward global optimum inspired by the Nelder–Mead simplex method [98]. A schematic block diagram of the DE algorithm is shown in Fig. 1. The pseudo-code of the DE algorithm with the binomial crossover operation is presented in Table 1. In the DE algorithm, each vector $x_{i,j}^G$ consists of D variables $x_{i,j}^G$ in the range $[x_{\min\,(j)}, x_{\max\,(j)}]$, $j = 1, \ldots, D$. The initial population should more effectively cover the entire search space as much as possible by uniformly randomized individuals with search space constrained using the prescribed minimum and maximum parameter bounds [97]. Initial vectors are randomly generated. We can initialize the jth parameter in the ith decision vector at the generation $G = 0$ as follows:

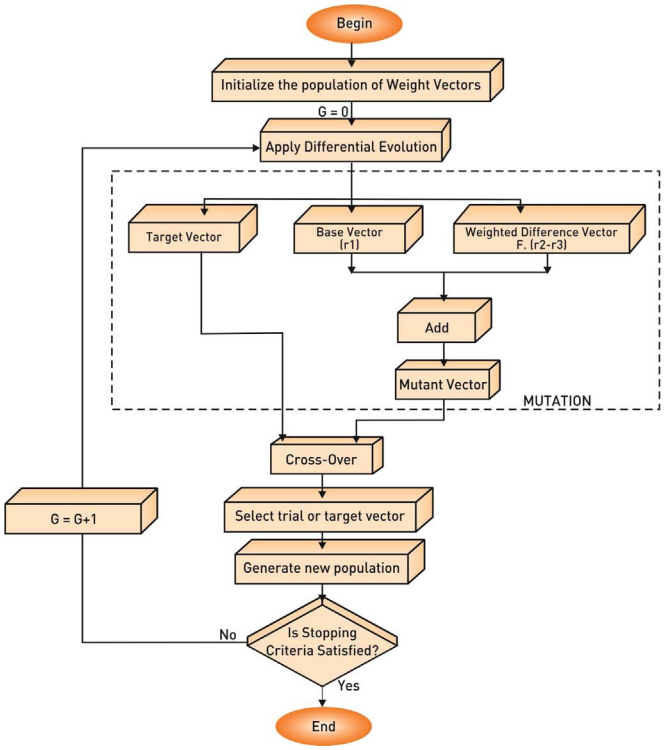

Fig. 1. Schematic block diagram of the DE algorithm.

Table 1
Pseudo-code of the DE algorithm with the binomial crossover operation.

Algorithm 1: DE Algorithm with Binomial Crossover Operation.

Input:	$f(.)$:	Objective function
NP (*pop*):	Population Size	
	$\boldsymbol{x_{min}}$:	Lower Bound
	$\boldsymbol{x_{max}}$:	Upper Bound
	$\boldsymbol{F}$:	Scaling Factor
	CR:	Crossover Rate
	$\boldsymbol{G_{max}}$:	Maximum Generation number

Step 1: Initialization- Set the generation number G = 0. Randomly initialize a population of NP target vectors
$P_G = \{X_1^G, X_2^G, \ldots, X_{NP}^G\}$, $with X_i^G = \{x_{i,1}^G, x_{i,2}^G, \ldots, x_{i,D}^G\}$, $i = 1,2,\ldots, NP$, uniformly distributed in the range $[\boldsymbol{x_{min}}, \boldsymbol{x_{max}}]$.
Step 2: Evaluate all target vectors.
Step 3: For each target vector:
Step 3.1: Perform mutation and create mutant vectors using the mutation operations as described in Section 2.2.1.
Step 3.2: Perform crossover and yield trial vectors using the crossover operation as mentioned in Section 2.2.2.
Step 3.3: Perform greedy selection for next generation. Generate the members of the target population as reported in Section 2.2.3.
Step 4: Increment the generation=G+1 .
Step 5: If a stopping criterion is met ($\boldsymbol{G=G_{max}}$), then output the most satisfactory/most favorable solution demonstrated/revealed/discovered, otherwise go back to Step 3.
Output: The candidate solution with smallest objective function value.

$$x_{i,j}^0 = x_{\min\,(j)} + \text{rand}_{i,j}(0, 1).\,(x_{\max(j)} - x_{\min(j)}),\ j = 1, \ldots, D \tag{5}$$

where $rand_{i,j}(0,1)$, represents a uniformly distributed random variable within the range [0,1].

The control parameters of the classical DE algorithm are population size NP, mutation scaling factor F, and crossover probability Cr. The DE algorithm incorporates simple arithmetic operators with traditional operators of crossover, mutation, and selection to evolve global optima (minima/maxima) from randomly generated candidate solutions. At every generation G during the optimization process, the DE algorithm maintains a constant population $x_{i,j}^G$.

2.2.1. Mutation operation

In the DE algorithm, mutation is a random change or perturbation of the population to approach a favorable solution in the search space. The parent vector (also called target vector) in the DE algorithm is mutated into a mutant vector V. Mutation is performed by perturbing a base vector with the scaled difference of two or more pairs of vectors adopting concept from the Nelder–Mead simplex method [113,58,93,95,96]. This technique as an inherent advantage of automatically adapting the resulting "step" size and orientation during the perturbation process to fitness function landscape. The DE algorithm employs mutation operation to produce a mutant (donor) vector v_i^G, $i = (1,2,\ldots..,N)$ with respect to each individual x_i^G. The following are the originally proposed and most frequently used mutation strategies in the DE literature:

(a) "DE/rand/1/bin":

$$v_i^{G+1} = x_{\mathrm{r1}}^G + F.\,(x_{\mathrm{r2}}^G - x_{\mathrm{r3}}^G), r1 \neq \mathrm{r2} \neq \mathrm{r3} \neq i \tag{6}$$

(b) "DE/best/1/bin" :

$$v_i^{G+1} = x_{best}^G + F.\ (x_{r1}^G - x_{r2}^G),\ r1 \neq r2 \neq i \tag{7}$$

(c) "DE/rand/2/bin" :

$$v_i^{G+1} = x_{r1}^G + F.\ (x_{r2}^G - x_{r3}^G) + F.\ (x_{r4}^G - x_{r5}^G),\ r1 \neq r2 \neq r3 \neq r4 \neq r5 \neq i \tag{8}$$

(d) "DE/best/2/bin":

$$v_i^{G+1} = x_{best}^G + F.\ (x_{r1}^G - x_{r2}^G) + F.\ (x_{r3}^G - x_{r4}^G),\ r1 \neq r2 \neq r3 \neq r4 \neq i \tag{9}$$

(e) "DE/ran-to-best/1/bin" :

$$v_i^{G+1} = x_i^G + F.\ (x_{best}^G - x_i^G) + F.\ (x_{r1}^G - x_{r2}^G),\ r1 \neq r2 \neq i \tag{10}$$

Note: Types (a) and (c) are suitable for global search, whereas types (b), (d), and (e)are applicable for local search ([113–116,126,137,141,58,93,95]; Zhang et al., 2009).

2.2.2. Crossover operation

Crossover in the DE algorithm is similar to the crossover occurring in biological evolution by randomly preserving some dimensions of the parent vector in the offspring and including the remaining dimensions from the mutant vector. The crossover operation creates a trial vector by mixing the target and mutant (donor) vectors according to a predefined probability. The most popularly used crossover methods in the DE family of algorithms are exponential (or two points modulo) and binomial (uniform). We adopted the widely applied variant binomial crossover technique [118], which is described as follows:

$$u_{i,j}^{G+1} = \begin{cases} v_{i,j}^{G+1} if\ (rand_j[0,\ 1)) \leq CR or\ (j = j_{rand}) \\ x_{i,j}^G otherwise \end{cases}$$

$$j = 1,\ \ldots..,\ D \tag{11}$$

where $u_{i,j}^{G+1}$ means indicates that the jth number/dimension of trial vector u_i^{G+1}, $rand_j[0,1)$ is a random number between [0, 1], the crossover rate (CR) is a user-defined constant within the range [0,1), guiding the fraction of parameter values reproduced from the mutant vector, j_{rand} is the randomly generated integer in the range [1,D] to assure that at least one dimension is selected from the mutant vector

Note: The binomial crossover operator replicates the jth component of the mutant vector v_i^{G+1} to the corresponding element in the trial vector $u_i^{G+1} if\ (rand_j[0,1)) \leq CR\ or\ (j = j_{rand})$. Otherwise, it is reproduced from the corresponding target vector x_i^G.

2.2.3. Selection operation

Selection operation decides whether the target (parent) or trial (offspring) vector survives into the next generation. The objective function value of each trial vector $f(u_i^{G+1})$ is compared with that of its corresponding target vector $f(x_i^G)$ in the current population. If the trial vector yields lower objective function value compared with the corresponding target vector, the trial vector replaces the target vector and enters the population of the next generation. Otherwise, the target vector is retained in the population into the next generation. The selection operation is represented as follows:

$$x_i^{G+1} = \begin{cases} u_{i,G+1} iff\ (u_i^{G+1}) \leq f(x_i^G) \\ x_i^G otherwise \end{cases} \tag{12}$$

Where $f(u_i^{G+1})$ and $f(x_i^G)$are the function values ofu_i^Gandx_i^G, respectively, and $f(u_i^{G+1}) \leq f(x_i^G)$ is employed to resolve the minimization problems.

3. Related work

In their systematic literature review (SLR) of 304 Software development effort/cost estimation papers from 76 journals, Jørgensen [19] recognized several software cost estimation techniques, broadly classified into two models: parametric models (predominantly based on numerical and/or statistical analysis of historical project data) and machine learning models (based on black box optimization principle, such as artificial neural networks (ANNs), support vector machine, decision trees, GA, and ABE or CBR). The major advantage of ML techniques is that these provide minimal approximation regarding the function to be modeled. Hence, they can easily hypothesize the complicated relationship between software cost drivers and effort. In their SLR, Wen et al. (2012) identified eight machine learning models for SDEE; of these, ANN and ABE were the most frequently used SDEE techniques. This inference is consistent with that of the aforementioned SLR by Jorgensen and Shepperd [50]. However, research findings suggest ANN purely follow the black-box optimization principle: the rules (reasoning) framed by the learned neural network model is unknown. Because the ABE model mimics human reasoning, it is easy to understand [121]. Hence; we adopted ABE technique to construct our SDEE model.

ABE follows the fundamental principle of calculating SDEE of a project on the basis of the similarity of software project feature characteristics on that of historical PB. It incorporates precise expert opinion, and data-driven modeling approaches through experience learning and exposure to similar projects. The data-driven ABE technique includes four components [23] as described in Section 2.1.The application of ABE for more accurate effort prediction incorporates some critical design choices, broadly categorized by researchers into feature subset selection, scaling, similarity measure, number of analogies to use (i.e., finding a suitable k-value), and solution function [119,23,62].

The selection of a similarity function plays a crucial role, which influences the selection of similar projects for a given training/testing project. The problem of formulation of similarity function has been an area of interest in the SDEE domain. The literature suggests different techniques formulating the similarity function ([23,47,85,62]; Mendes et al. 2003a, 2003b). Various researchers (e.g., [61]; Chen et al. 2005; [85]; Kirsopp et al. 2003; [23,37]) have extensively studied the impact feature subset selection on prediction accuracy. Idri et al. [145] reported the effectiveness of Fuzzy and classical Analogy ensemble SDEE techniques. Idri et al. [146] investigated the impact of missing data on Fuzzy and Classical Analogy-based SDEE. Azzeh et al. [123] used Ensemble learning techniques for variants of adjustment methods used in Analogy based Estimation.

Walkerden and Jeffery [33] proposed the linear size extrapolation technique to tune nearest analogies by size extrapolation. This method provided improved performance compared with the baseline ABE method. Stamelos and Angelis [67] used statistical simulation techniques to evaluate confidence intervals for the effort required for a project portfolio. Jørgensen et al. [49,64] proposed RTM adjustments technique to improvise the ABE models. Auer et al. [24] proposed a novel approach, which exercises comprehensive search to optimally weigh individual project features, facilitating a more realistic measure of project similarity. Li et al. [85] reported various similarity measures for non-quantitative data types, such as ordinal, nominal, set, fuzzy, and range scale types; the authors annotated the framework as AQUA+. Huang and Chiu [35] investigated the performance of ABEs when the GA method was followed and calibrated reused effort depending on the similarity distances between a new project and its analogies on all project features. By contrast, Li et al. [37] used the ANNs to learn the differences in effort values between a new project and its analogies through learning effort differences between previous projects in the training dataset and then added the difference produced to the nearest analogy. Heung et al. (2008) proposed the prospects of the SDEE

model by integrating a GA to the GRA. The GA technique is used to evaluate the best fit of weights for each software effort driver in similarity measures. Mittas et al., [68] exploited the relation of the ABE technique to the nearest neighbor nonparametric regression as a resampling procedure, termed as iterated bagging, for decreasing the prediction error. Tosun et al. [92] presented two weight assignment heuristics motivated by a popularly used statistical technique called PCA. Azzeh [65] proposed a new adaptation strategy by using model tree-based attribute distance to fine tune ABE and obtain new estimates.

The DE algorithms have evolved by refining the control parameters, namely learning/mutation strategy, CR, and scaling factor. These state-of-the-art learning strategies are opposition-based DE and generalized opposition-based DE, which incorporate the contrasting number of initializing population for new generation creation and local improvement [131,133] proposed a novel crossover operator improvisation technique by mixing eigenvector-based crossover with binomial crossover operator. In their study, Wang et al. [135] adapted orthogonal crossover operator to improve performance. Adaptive differential evolution with optional external archive (JADE) (Zhang et al., 2009) is a well-known scaling factor adaptation technique; other relevant competitive techniques include those proposed by Islam et al. [136], Gong et al.[114,137,141], Gong et al. [138],and Aalto and Lampinen [139]. Self-adaptive DE (SaDE) [112] is a widely accepted crossover adaptation techniques, while other competitive methods include those proposed by Islam et al. [136], Gong et al. [114,137,141], Zhao et al. [140], Gong et al. [114,137,141], and Gong et al. [138]. All these preceding methods induce crossovers at respective generations according to the normal distribution with the mean value gained by successful crossovers. Several DE algorithm variants have on the basis of the pooling of multiple learning strategies, such as EPSDE [142] and jDElscop [143]. Piotrowski [130] investigated the effect of population size on the performance of the DE algorithm. For more information on contemporary DE techniques, please refer to the works of Das et al. [118], Guo et al. [131], and Penunuri et al. [132].

4. DABE framework

In this section, we first present our evaluation criteria, followed by the DABE framework.

4.1. Evaluation criteria

The performance indicator popularly used for measuring the efficacy of the software prediction models are the magnitude of relative error (MRE), which computes the absolute percentage of error to the actual effort as shown in Eq. (14); mean of MRE (MMRE), as shown in Eq. (15);and performance indicator PRED(x), defined as the percentage of predictions falling within the actual known value x, specified as PRED as shown in Eq. (16). For obtaining a more robust prediction model, Araújo et al. [55] proposed an evaluation function (EF), a combination of two well-known performance measures MMRE and PRED(25), as given by Eq. (17). EF was constructed to globally assess the prediction model, where the key concept is to maximize the PRED(25) metric and simultaneously minimize the MMRE metric. Thus, EF can be used for the performance metrics of the prediction model [55].

The measures based on MRE have inherent limitations as performance metrics because of their asymmetric distribution, biasing toward prediction model that are. MMRE and PRED are built on MRE. Thus, they are also biased measures underestimated [52–54]. Another test, called the mean absolute error (MAE), has also been reported; it is not biased and does not exhibit asymmetric distribution as does MMRE. The MAE is calculated averaging absolute error (AE; Eqs. (13) and (18)). However, its interpretation was difficult because its residuals are not standardized. Hence, Shepperd and MacDonell [54] reported a new measure called as standardized accuracy (SA; Eq. (19)), which evaluates the effect size, as shown in Eq. (20). The SA ratio is mainly used to test whether a prediction model outperforms a baseline of random guessing and generates meaningful predictions. If not, this prediction model cannot be deemed useful. The reliability of a prediction model can be quantified using SA performance metrics. The interpretation of SA provides the effectiveness of given model compared with random guessing. An SA value close to zero indicates lower reliability; furthermore, a negative value is considered unacceptable.The effect size (Δ)is used to verify whether the predictions of a given model are estimated by chance and to confirm whether there has been a substantial improvement in effectiveness compared with random guessing, because statistical analysis alone is insufficient when the prediction models are significantly different. The value of Δ can be explained in terms of small (0.2), medium (0.5), and large (0.8) categories, where a value ≥ 0.5 is considered more favorable [122,54].

$$AE_i = |y_i - \hat{y}_i| \tag{13}$$

$$MRE_i = \frac{AE_i}{y_i} \tag{14}$$

$$MMRE = \frac{1}{N}\sum_{i=1}^{N} MRE_i \tag{15}$$

$$PRED(x) = \frac{100}{N} \times \sum_{i=1}^{N} D_i \tag{16}$$

$$D_i = \begin{cases} 1 \; if \; MMRE < \frac{x}{100} \\ 0 \; otherwise \end{cases}$$

When $x = 25$ the PRED metric is defined as PRED(0.25).

$$EF = \frac{PRED(25)}{1+MMRE} \tag{17}$$

$$MAR = \frac{\sum_{i=1}^{N} AE_i}{N} \tag{18}$$

$$SA = 1 - \frac{MAR}{\overline{MAR_{P_0}}} \tag{19}$$

$$\Delta = \frac{MAR - \overline{\mathrm{MAR}_{\mathrm{P}_0}}}{\mathrm{s}_{\mathrm{P}_0}} \tag{20}$$

where $y_i and \hat{y}_i$ are the actual and estimated effort of a particular project, MAR is the mean absolute residual of the prediction model, $\overline{MAR_{P_0}}$ is the mean value of a large number (typically 1000) of runs of random guessing. This value predicts a$\hat{y}_i$for the target case t by randomly sampling (with equal probability) over all the remaining n −1 cases and considers$y_i = y_r$, where r is drawn randomly from 1,……..,$n \bigwedge r \neq t$. This randomization procedure is robust because it makes no assumptions and requires no knowledge concerning population.s_{P_0} is the sample standard deviation of the random guessing strategy.

4.2. Proposed DABE framework

We propose our DABE framework in this subsection. First, we discuss the training stage in Section 4.2.1 and then the testing stage in Section 4.2.2. A collection of training examples are fed to the model in the training stage; the ABE system is tuned by the candidate parameters (feature weights) and the DE algorithm explores the weight vector space to minimize the error (in terms of MMRE) of ABE on the training examples. The learned/trained DABE framework is presented with test data (new project) for effort prediction. The system architectures of DABE training and testing stages are illustrated in Figs. 2 and 3, respectively. The pseudo-code of DABE framework is presented in Table 2.

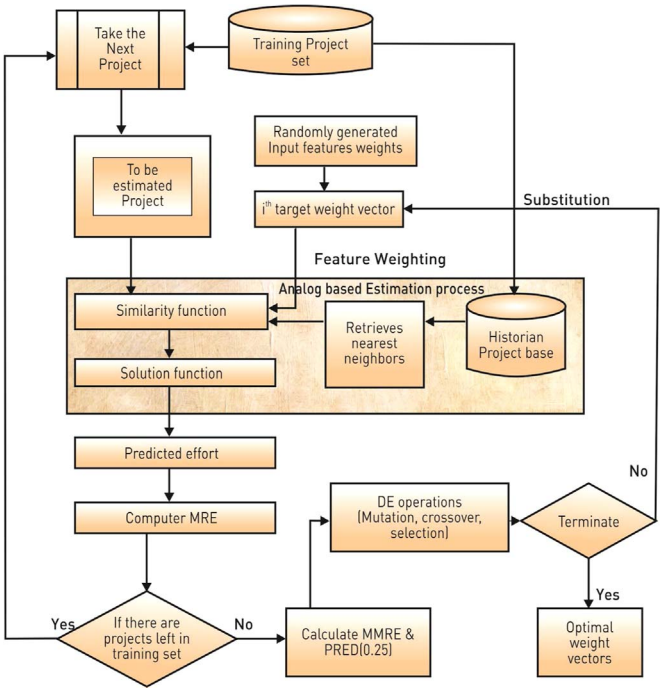

Fig. 2. The system architecture of DABE training stage.

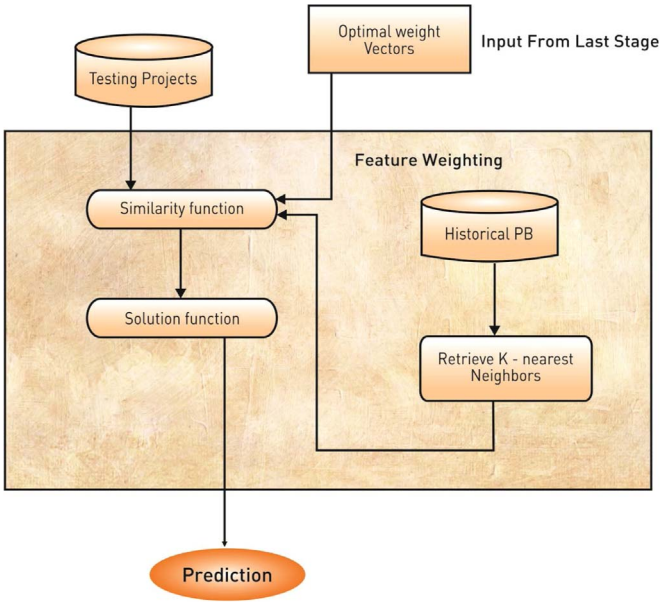

Fig. 3. The System architecture of DABE testing stage.

4.2.1. Training stage

The system architecture of the training stage is shown in Fig. 2. Historical PB was used to predict the efforts of the training dataset in the training stage. We created a population from randomly selected target vectors with dimension NP. Target vector encoding of the weights of the proposed DABE was given as follows:

The *i*th target vector of the current generation G is

$$P_G = \{W_1^G, W_2^G, \ldots, W_{NP}^G\},\ with W_i^G = \{w_{i,1}^G, w_{i,2}^G, \ldots, w_{i,D}^G\},\ i = 1, 2, \ldots, NP,$$

Where D is the dimension of the weight vector, W_i^G is the *ith* target (weight)vector of the population, and G is the generation to which the population belongs.

4.2.2. Testing stage of DABE framework

The testing stage is depicted in Fig. 3. In the testing stage, we combined the historical PB and training projects to act as the historical PB.A new project "*p*"was then submitted to the trained DABE system. We next retrieved a set of k analogies from the aforementioned historical PB by using Eqs. (2) and (3) to compute the similarities. After obtaining the K analogies, we used the retrieved solution function to predict the effort.

5. Data sets and experiment setup

In this section, we first present the dataset description in Section 5.1, followed by the SDEE techniques used for comparison and detailed experimental procedure in Sections 5.2 and 5.3, respectively.

5.1. Dataset description

To assess the performance of our method, we used six software measurement datasets, publically available at http://openscience.us/repo/, obtained from the following sources: Desharnais, Cocomo81, Nasa93, Maxwell, Albrecht, and China datasets. The Desharnais dataset includes Canadian software projects. The Cocomo81 and Nasa93 datasets contain projects developed in the United States. The China dataset comprises Chinese software projects. The Maxwell dataset contains Finnish banking software projects. The Albrecht dataset includes projects completed by IBM in the 1970s.

Dejaeger et al. [72] categorized the datasets into four classes: size features, development features, environment features, and project data. Table 3 lists the descriptive statistics of datasets. The skewness of the effort values of promise repository dataset is upto 6.6 [71,73], implying that the effort of each dataset is not normally distributed and poses a challenge for developing accurate SDEE methods.

5.2. SDEE techniques

The performance of the proposed DABE framework was validated and compared with the most efficient and commonly used variants of other popular machine learning models, namely conventional ABE, particle swarm optimization-based feature weight optimization in ABE (PSO-ABE),GA-based feature weight optimizationin ABE (GA-ABE), SaDE-based feature weight optimization in ABE (SaDE-ABE), JADE-based feature weight optimization in ABE(JADE-ABE) (Zhang et al. 2009), functional link ANN (FLANN) ([78,80,81]; Benala et al. 2009), ANN with back propagation learning-based SDEE [88–90], and radial basis function(RBF)-based SDEE [21], all obtained by training the estimating models using historical data and automatically adjusting algorithmic parameters.

The frameworks of PSO-ABE, GA-ABE, SaDE-ABE, and JADE-ABE were equivalent to the structure of the DABE framework. This was deliberately considered to construct four swarm intelligence techniques as equal level players such that their performance can be validated and compared.

SaDE, proposed by Qin et al. (2005), comprises self-adaptation of learning strategies and their associated control parameters during evolution. We selected two learning strategies "DE/rand/1″ and "DE/current-to-best/1," with first equal probabilities represented by $p1 and\ p2$, respectively for generating trial vectors. The probabilities $p1 and\ p2$ are updated as per the formulation suggested by Qin et al. [126]. We configured the control parameters as follows: The population size NP was set to 10 and mutation factor F was independently spawned at each generation with a mean of 0.5 and variance of 0.3 truncated to the interval (0, 2] with normal distribution. Cr was normally distributed with mean CR_m and variance 0.1. At first, CR was set to 0.5; then, we generated it after every five generations as suggested by Qin et al. (2005).

JADE, proposed by Zhang et al. (2009), instigates a new mutation strategy, namely "current-to-pbest/1,"with an optional archive and updated control parameters in an adaptive manner. The recently

Table 2
Pseudo-code of DABE framework.

Algorithm 2 DABE Framework

Input: $f(.)$: Objective function
NP (*pop*): Population Size
$\boldsymbol{x_{min}}$: Lower Bound
$\boldsymbol{x_{max}}$: Upper Bound
$\boldsymbol{F}$: Scaling Factor
CR: Crossover Rate
$\boldsymbol{G_{max}}$: Maximum Generation number

Step 1: Parameter setup:
Initialize population ***pop***, set ***pop*** = 10Dwhere D is the dimension of the objective function [124]
Scale Factor F = 0.8
Cross over rate =0.5
Set G=0 (Generation number)
Step 2: One training project is chosen from the training dataset as the new project to be estimated, and the remaining projects are considered as the historical projects in the ABE system.
Step 3: Encode the feature weights of the training project into parameter vector of DE.
Step 4: Randomly generate a target vector (weight vector) within the range [0, 1].
Step 5: Evaluate the similarity measure for the training project using Eqs. (1) and (2) with the randomly selected weight vector from ***pop***.
Step 6: Then, ABE uses the K closest analogies from the historical PB. Finally, the ABE model allocates a prediction value to the training project by choosing different solution functions.
Step 7: The steps 2–6 are repeated till all the training examples are processed using the same random target vector (generated for first training example).
Step 8: The performance indicators MMRE, PRED(0.25), and MdMRE are applied to evaluate the prediction accuracy of the training project set. In our work DE is intended to maximize the fitness function.The reciprocal of $(MMRE - PRED(0.25)+\epsilon)$ is employed as the objective function.
The ϵ is a small positive constant intentionally introduced to avoid denominator being zero.
Step 9: The Evolution Step //While stopping criteria is not satisfied
Step 9.1: Mutation Step
For each target vector in ***pop*** select random variables (target vectors) from ***pop*** $r_1, r_2, r_3 \in \{1,2,\ldots,NP\}$. Apply mutation operator to each target vector in population to generate mutant vectors using the mutation operations depicted by Eqs. (6)–(10).
Step 9.2: Crossover Step
Apply crossover and generate trial vectors using the crossover operation describes in Eq. (11).
Step 9.3: Selection Step
To keep the population size constant over subsequent generations, the next phase of the algorithm calls for selection to determine whether the target or the trial vector survives to the next generation. Determine the members of the target population using the formula represented in Eq. (12).
Note: The selection criteria used in DE [59] is a one-to-one survivor selection where one trial vector competes against one target vector, allowing keeping the population size. Constant. The vector which maximizes the objective function $f(W_i^G)$, survives into the next generation
Step 9.4: Increase the generation count.
G=G+1.
Step 10: Stopping criteria –The population is evolved by the DE algorithm until
G_{max}isequaltoorexcess1000orthebestfitnessvaluedidnotchangeinthelast200 generations.If the stopping criteria are satisfied then go to step 12.
Step 11: Go To Step 9.
Step 12: EXIT.
Output: The candidate solution with largest objective function value is selected as an optimal target (weight) vector for next stage (testing stage).

Table 3
Statistical properties of employed datasets.

Dataset	Features	Number of Projects	Effort Data					
			Unit	Min	Max	Mean	Median	Skew
Desharnais	12	77	Hours	546	23,940	5046	3647	2.0
Nasa	**3**	18	Months	5	138.3	49.47	26.5	0.57
Cocomo	**17**	63	Months	6	11,400	686	98	4.4
China	**18**	499	Hours	26	54,620	3921	1829	3.92
Maxwell	**27**	62	Hours	583	63,694	8223.2	5189.5	3.26
Albrecht	**7**	24	Months	1	105	22	12	2.2

Table 4
Results of effect size for Desharnais dataset.

Model	Min. Δ	Max. Δ	Avg. Δ	Std. Δ	# Small	# Medium	#Great/Immense/High/ Considerable/Long	# Medium +Great/Immense/High/ Considerable/Long
DABE1	0.075	0.858	0.494	0.229	2	10	12	22
DABE2	0.091	0.99	0.439	0.227	4	12	8	20
DABE3	0.11	1.126	**0.529**	**0.272**	4	11	10	**21**
DABE4	0.221	1.027	0.459	0.197	3	13	8	21
DABE5	0.11	0.919	0.429	0.224	6	9	9	18

explored inferior solutions were archived and used in this strategy to improve the performance. We adapted JADE with archive as suggested by Zhang et al. (2009) for simulation.

The ANN models are specified by network topology, neuron characteristics, and training or learning rules [91], with inputs, output(s), and hidden layer(s) with interconnections. The input layer depends on several features in a project; output layer comprises only one neuron as it is a function approximation optimization problem. The hidden layer is set to one to prevent an over fitting problem. The number of neurons in the hidden layer is obtained through trial and error. In this study, we set this number to 4. At first, the historical data were used to predict the costs of the training set during the training phase. We then combined the historical and training sets and used them to predict costs of the test set during the experimental phase.

FLANN was constructed as proposed by Benala et al. [78,80,81,]. FLANN can be incorporated with optimally reduced datasets to make it faster. Optimal reduced datasets reduce the whole project base into small subsets, comprising only representative projects. We used the Chebyshev polynomial as the functional expansion block in FLANN and provided the representative projects as input to it. Furthermore, we validated the system was and obtained performance measures, as mentioned in Section 4.1. We then implemented the structure of FLANN on the training set and used the most satisfactory performing values that were obtained for the comparison; the sets yielding most satisfactory SA and Δ values on training data were selected for comparisons.

RBF was constructed as presented by Shin et al. [21]. For a given global width σ (variance) and a measure δ of closeness between the interpolation matrix and its approximation by m vector terms, we first determined the minimum number of basis functions that provide the desired degree of approximation. Based on this (m; σ) combination, we next determined the centers of the basis functions to complete the design matrix. Finally, we compute the weights to the output node by using the pseudo inverse. We first develop an RBF model for $\delta = 1\%$. According to the heuristic argument of Shin et al. [21], we selected values of variance as 0.25–0.75 in increments of 0.05. For each, we next applied the SG algorithm to determine the value of m that satisfies the δ criterion. For the (DL-Y) data, we obtained the computed values of m for a variance from 0.25 to 0.75. Similar to the training procedure for ANN, we trained all RBFs with different σ values on the historical dataset and selected the RBF yielding most satisfactory SA and Δ values on training data for comparisons.

5.3. Experiment setup

In this section, we describe the experimental set up that we have used in our experimentations. As a preprocessing step, we eliminated the different influences of the features by normalizing in the interval [0, 1] by using the min–max normalization approach [73].Each method was evaluated over all datasets by using 10-fold cross validation to identify testing and training data. We divided randomly permuted and divided the datasets into 10 approximately equal-sized bins. One bin was used as the test set, whereas the other bin was used as the training set and historical PB. In other words, we used nine-tenth of the project data to build the model and one-tenth to validate the prediction accuracy of the established model. The historical PB was used to predict the costs of the training set during the training phase. The historical PB and training sets were then combined and used to predict costs of the test set during the experimental phase. We repeated the cross validation as long as the standard deviation over the runs was greater than 1% of the mean accuracy or until 10 repetitions were completed [11]. We depicted the aggregation of the accuracies across all test datasets and the accuracies across all training datasets as testing and training results, respectively.

The three control parameters of ABE are similarity functions, k number of similar projects, and solution functions. First, we considered the effect of various combinations of these parameters on the performance of ABE by defining the search space of each parameter and selected the most suitable variant of our proposed model for comparing with other cost estimation methods. We then validated DABE variants (DABE-1 for “DE/rand/1/bin”; DABE-2 for “DE/best/1/bin”; DABE-3 for “DE/rand/2/bin”;DABE-4 for “DE/best/2/bin”; and DABE-5 for DE/ran-to-best/1/bin”),SaDE-ABE, JADE-ABE, PSO-ABE, and GA-ABE as per the DABE framework procedure. The experimental results and analysis are given in Section 7.

6. Experiment results

In this section, we present the experimental results obtained from a simulation study and their analyses. First, we explored the most suitable variant of DABE by using the effect size Δ on the basis of the experimental report of Minku and Yao [144]. Next, we used the simulation study to explore the possibility of obtaining the most favorable ABE configuration { k value, similarity measure, solution function}by using five performance indicators: MMRE, MdMRE,

Table 5
Results of effect size for Maxwelldataset.

Model	Min. Δ	Max. Δ	Avg. Δ	Std. Δ	# Small	# Medium	#Great/Immense/High/ Considerable/Long	# Medium +Great/Immense/High/ Considerable/Long
DABE1	0.133	1.691	0.538	0.351	3	9	12	21
DABE2	0.118	1.435	0.574	0.325	2	7	15	22
DABE3	0.21	2.048	**0.614**	**0.370**	2	7	14	**21**
DABE4	0.151	2.107	0.586	0.693	4	9	12	21
DABE5	0.109	1.758	0.612	0.374	2	9	13	22

Table 6
Results of DABE-3 for Maxwell dataset.

Similarity	K	Solution	Training			Testing		
			MMRE	PRED	MdMRE	MMRE	PRED	MdMRE
	K = 1	CA	0.003	0.516	0.016	0.010	1.000	0.010
	K = 2	Mean	0.072	0.025	0.072	**0.014**	**0.074**	**0.014**
		IWM	0.067	0.049	0.067	0.100	**0.074**	0.100
	K = 3	Mean	0.056	0.049	0.024	**0.085**	0.099	**0.028**
		IWM	0.067	0.074	0.067	0.100	**0.111**	0.100
		Median	0.049	0.049	0.050	0.086	0.099	**0.028**
Euclidean	K = 4	Mean	0.052	0.074	0.025	0.274	**0.111**	**0.100**
		IWM	0.067	0.099	0.067	0.100	0.148	**0.100**
		Median	0.048	0.074	0.017	**0.208**	**0.111**	0.106
	K = 5	Mean	0.051	0.099	0.026	0.313	0.123	0.172
		IWM	0.067	0.123	0.067	0.100	**0.185**	0.100
		Median	0.044	0.099	0.038	**0.203**	0.136	**0.078**
	K = 1	CA	0.024	0.554	0.016	0.009	1.000	0.009
	K = 2	Mean	0.072	0.025	0.072	**0.014**	**0.074**	**0.014**
		IWM	0.067	0.049	0.067	0.100	**0.074**	0.100
	K = 3	Mean	0.056	0.049	0.024	0.288	0.099	**0.028**
		IWM	0.067	0.074	0.067	**0.100**	**0.111**	0.100
		Median	0.049	0.049	0.050	0.307	0.099	**0.028**
Manhattan	K = 4	Mean	0.052	0.074	0.025	0.303	0.111	0.079
		IWM	0.067	0.099	0.067	**0.100**	**0.148**	0.100
		Median	0.048	0.074	0.017	0.276	0.123	**0.071**
	K = 5	Mean	0.051	0.099	0.026	0.336	0.123	0.131
		IWM	0.067	0.123	0.067	**0.100**	**0.185**	0.100
		Median	0.044	0.099	0.038	0.257	0.148	**0.078**

PRED(0.25),SA, and Δ. Finally we selected the best SDEE model by using all the performance measures described in Section 4.1. The most suitable prediction model was further confirmed through graphical analysis. We also selected the most suitable DABE variant on the basis of the fundamental notion of effect size Δ—larger Δ indicates higher practical significance of the model. We selected Desharnais and Maxwell datasets to perform the simulation study. Furthermore, to confirm the robustness of the potential model, we explored all possible ABE configurations while calculating Δ.The generalization capability of all the DABE variants was accounted for calculating Δ. Table 4 presents the results of Δ for Desharnais dataset. DABE-3 outperformed other variants with a mean (standard deviation) of 0.529(0.272).In 21 cases, Δ were ≥ 0.5. Table 5 depicts the results of Δ for Maxwell dataset. DABE-3 outperformed other variants with a mean (standard deviation) of 0.614 (0.37). Here as well, the Δ was ≥ 0.5 in 21 cases. Thus; DABE-3 was selected as the most suitable variant for comparison with other SDEE techniques.

Second, we investigated the most favorable ABE configuration by using the test performance of MMRE, MdMRE, PRED, and SA for Desharnais and Maxwell datasets. Tables 6 and 7 present the generalization errors for Maxwell and Desharnais datasets, respectively; the most significant values are marked bold. According to the MMRE and

Table 7
Results of DABE-3 for Desharnaisdataset.

Similarity	K	Solution	Training			Testing		
			MMRE	PRED	MdMRE	MMRE	PRED	MdMRE
	K = 1	CA	0.017	0.667	0.010	0.048	0.988	0.031
	K = 2	Mean	0.001	0.121	0.001	**0.020**	**0.182**	**0.020**
		IWM	0.067	0.121	0.067	0.100	**0.182**	0.100
	K = 3	Mean	0.048	0.121	0.003	**0.023**	**0.273**	**0.021**
		IWM	0.067	0.182	0.067	0.100	**0.273**	0.100
		Median	0.064	0.121	0.003	0.025	**0.273**	0.026
Euclidean	K = 4	Mean	0.038	0.182	0.002	0.046	**0.364**	**0.026**
		IWM	0.067	0.242	0.067	0.100	**0.364**	0.100
		Median	0.050	0.182	0.006	**0.037**	**0.364**	0.028
	K = 5	Mean	0.060	0.242	0.003	0.093	0.424	**0.025**
		IWM	0.067	0.303	0.067	0.100	**0.455**	0.100
		Median	0.075	0.242	0.009	**0.065**	0.424	0.027
	K = 1	CA	0.015	0.667	0.010	0.036	0.988	0.025
	K = 2	Mean	0.001	0.121	0.001	**0.033**	**0.182**	0.033
		IWM	0.067	0.121	0.067	0.100	**0.182**	0.100
	K = 3	Mean	0.048	0.121	0.003	0.042	**0.273**	**0.015**
		IWM	0.067	0.182	0.067	0.100	**0.273**	0.100
		Median	0.064	0.121	0.003	**0.035**	**0.273**	0.028
Manhattan	K = 4	Mean	0.038	0.182	0.002	0.043	**0.364**	0.031
		IWM	0.067	0.242	0.067	0.100	**0.364**	0.100
		Median	0.050	0.182	0.006	**0.035**	**0.364**	**0.022**
	K = 5	Mean	0.060	0.242	**0.003**	0.090	0.424	0.037
		IWM	0.067	0.303	0.067	0.100	**0.455**	0.100
		Median	0.075	0.242	0.009	**0.063**	0.424	**0.025**

Table 8
Results of SA for Desharnaisdataset.

Similarity	K	Training				Testing			
		Min. SA	Max. SA	Avg.SA	Std.SA	Min. SA	Max. SA	Avg.SA	Std.SA
	3	10.602	**90.126**	**36.584**	26.206	28.165	**89.886**	**53.74**	19.18
Euclidean	4	10.03	88.438	38.077	22.325	25.39	83.669	53.99	17.887
	5	11.102	30.681	21.014	5.895	22.859	88.623	42.171	15.934
	3	11.539	63.363	31.873	16.996	22.398	79.268	51.367	17.755
Manhattan	4	11.803	68.415	30.112	17.092	16.102	80.788	38.788	11.165
	5	10.928	31.294	20.94	6.826	15.164	65.388	37.503	11.165

Table 9
Results of SA for Maxwell dataset.

Similarity	K	Training				Testing			
		Min. SA	**Max. SA**	**Avg.SA**	**Std.SA**	**Min. SA**	**Max. SA**	**Avg.SA**	**Std.SA**
	3	12.126	**89.657**	**50.484**	30.418	27.38	**96.183**	**59.954**	14.693
Euclidean	4	13.301	95.683	35.395	27.094	23.07	53.403	39.464	8.356
	5	15.175	27.556	22.055	3.826	32.329	85.308	50.43	12.522
	3	11.77	68.118	39.293	14.523	21.219	88.736	57.88	18.927
Manhattan	4	13.709	79.116	33.149	20.539	23.108	71.861	45.239	14.652
	5	13.284	63.447	26.114	10.045	26.837	75.489	50.177	12.898

Table 10
Results of solution function for Maxwell dataset for k =3.

Solution function	Training				Testing			
	Min. SA	**Max. SA**	**Avg. SA**	**Std. SA**	**Min. SA**	**Max. SA**	**Avg. SA**	**Std. SA**
Mean	26.348	83.84	42.43	21.68	44.36	**96.18**	**58.26**	18.77
IWM	12.13	84.92	33.89	33.74	27.38	84.18	58.59	18.12
Median	38.52	89.66	75.13	19.24	47.57	66.11	57.01	6.01

Table 11
Results of solution function for Desharnais dataset for k = 3.

Solution function	Training				Testing			
	Min. SA	**Max. SA**	**Avg. SA**	**Std. SA**	**Min. SA**	**Max. SA**	**Avg. SA**	**Std. SA**
Mean	27.5	39.9	30.54	4.46	30.96	**83.19**	**51.35**	22.58
IWM	10.03	88.44	33.79	30.05	25.39	83.66	50.6	19.02
Median	19.2	84.51	49.9	25.3	44.14	74.24	60.04	11.44

Table 12
Results and comparisons for Albrecht dataset.

Models	MMRE		PRED		MdMRE		SA		Δ	
	Training	**Testing**	**Training**	**Testing**	**Training**	**Testing**	**Training**	**Testing**	**Training**	**Testing**
ABE	0.050	0.221	0.367	0.875	0.037	0.091	16.584	27.811	0.633	3.471
GA-ABE	0.095	0.018	0.167	0.250	0.001	0.018	89.452	83.233	0.291	1.734
DABE−3	0.095	0.020	0.167	0.375	0.001	0.019	95.523	93.636	0.266	1.698
PSO-ABE	0.081	0.018	0.333	0.250	0.039	0.018	76.824	83.180	0.286	1.714
SADE-ABE	0.015	0.027	0.376	0.291	0.018	0.031	92.451	90.453	1.481	1.251
JADE-ABE	0.023	0.016	0.421	0.262	0.291	0.018	91.953	88.753	0.451	0.985

MdMRE performance measures, the most suitable ABE configuration for both Maxwell and Desharnais datasets was {ES, k = 3, mean solution function}.However, the PRED performance indicator test performance indicated that the most favorable ABE configuration for both data sets was {ES/MS measure, k = 5, IWM solution function}.Hence, we evaluated another performance measure SA for Maxwell and Desharnais datasets to confirm the most suitable ABE configuration. We analyzed test results to evaluate the minimum,

Table 13
Results and comparisons for China dataset.

Models	MMRE		PRED		MdMRE		SA		Δ	
	Training	Testing	Training	Testing	Training	Testing	Training	Testing	Training	Testing
ABE	0.131	0.116	0.259	0.868	0.116	0.041	16.035	12.493	1.313	0.042
GA-ABE	0.076	0.100	0.374	0.167	0.027	0.100	82.344	86.196	0.327	3.318
DABE−3	0.075	0.068	0.621	0.573	0.020	0.039	95.847	**96.509**	0.259	**1.818**
PSO-ABE	0.067	0.010	0.111	0.167	0.067	0.100	69.451	92.880	0.350	5.843
SADE-ABE	0.079	0.084	0.561	0.483	0.058	0.046	95.189	94.203	0.784	1.231
JADE-ABE	0.092	0.076	0.347	0.212	0.075	0.081	90.651	88.265	1.143	0.891

Table 14
Results and comparisons for Cocomo81 dataset.

Models	MMRE		PRED		MdMRE		SA		Δ	
	Training	Testing	Training	Testing	Training	Testing	Training	Testing	Training	Testing
ABE	0.010	1.826	0.667	0.492	0.01	0.274	10.617	24.531	0.533	0.794
GA-ABE	0.068	0.097	0.778	0.735	0.009	0.098	89.414	91.812	0.277	0.546
DABE−3	0.068	0.050	0.878	0.816	0.009	0.042	96.033	**98.940**	0.265	**0.158**
PSO-ABE	0.067	0.861	0.118	0.137	0.067	0.021	70.053	88.016	0.185	0.244
SADE-ABE	0.025	0.016	0.712	0.654	0.035	0.042	93.451	91.923	0.438	0.231
JADE-ABE	0.054	0.129	0.671	0.554	0.089	0.114	92.876	89.546	0.354	0.132

Table 15
Results and comparisons on Nasa93 dataset.

Models	MMRE		PRED		MdMRE		SA		Δ	
	Training	Testing	Training	Testing	Training	Testing	Training	Testing	Training	Testing
ABE	0.012	0.144	0.667	0.839	0.012	0.073	10.601	11.488	0.222	0.611
GA-ABE	0.049	0.009	0.137	0.118	0.036	0.009	67.605	90.324	0.288	0.546
DABE−3	0.066	0.074	0.157	0.157	0.066	0.018	96.591	94.234	0.265	0.240
PSO-ABE	0.066	0.099	0.118	0.176	0.066	0.099	89.379	93.010	0.291	0.266
SADE-ABE	0.032	0.057	0.295	0.185	0.0.21	0.045	95.341	93.451	0.348	0.154
JADE-ABE	0.054	0.063	0.342	0.178	0.055	0.051	93.423	91.891	0.321	0.176

Table 16
Prediction results for Desharanais dataset.

Model	PRED (25)	MMRE	EF
ABE	99.9	0.023	97.654
GA-ABE	86.5	0.032	83.818
DABE1	100	0.023	97.752
DABE2	100	0.018	98.232
DABE3	100	0.017	98.328
DABE4	100	0.023	97.752
DABE5	100	0.023	97.752
PSO-ABE	100	0.018	98.232
SADE-ABE	100	0.019	98.135
JADE-ABE	98.2	0.031	95.247
ANN	82.716	1.10E−05	82.715
FLANN	91.358	7.00E−06	91.357
RBF	96.8	0.24578	77.702

maximum, average, and standard deviation of SA. Table 8 presents the results of SA for Desharnais dataset: In the training stage, the best SA occurred at k = 3, with the maximum and average SA values of 90.126 and 36.584, respectively. We obtained the most suitable SA value at k = 3 in the testing stage, with a maximum and average values of 89.886 and 53.74, respectively.

Table 9 lists the results of SA for Maxwell dataset. In the training stage, the best value of SA occurred at k = 3, with the maximum and average SA reported to be 89.657 and 50.484, respectively. In the testing stage, the most favorable value of SA was at k = 3, and the maximum and average SA values were 90.183 and 59.954, respectively. In the simulation study, k = 1 and 2 were not used for comparison because all solution functions were not covered at these values. Finally, to strengthen the solution function most suitable for the optimal value of k (i.e., 3), we performed the simulation study on SA for Maxwell and Desharnais datasets. We selected the generalization capability of SA as the benchmark for result assessment. Table 10 lists the minimum, maximum, average, and standard deviation of SA for the three solution functions mean, IWM, and median for Maxwell dataset. Our result analysis suggested optimal maximum and average SA values of 96.18 and 58.26, respectively, for "mean" solution function. Table 11 lists the minimum, maximum, average, and standard deviation of SA for the three solution functions mean, IWM, and median for Desharnais dataset. Our analysis report indicated optimal maximum and average SA values of 83.19 and 51.35, respectively, for the "mean" solution function.

All analyses performed in this section concluded that the most suitable ABE configuration is {ES, k = 3, mean solution function}.

Tables 12–16 summarize the results and comparison of ABE, GA-ABE, DABE-3, PSO-ABE, SaDE-ABE, and JADE-ABE on four PROMISE test suits, namely Albrecht, China, Cocomo81, and Nasa93,respectively, with the {ES, k = 3, mean solution function} configuration. The values of the < SA,Δ > pair for DABE-3 for each dataset in respective training and testing stages are listed as follows: Albrecht, < 95.523,0.266 > and < 93.636,1.698 > (Table 12);China, < 95.847,0.259 > and < 96.509,1.818 > (Table 13);Cocomo81, < 96.033,0.265 > and < 98.940,0.158 > (Table 14); Nasa93, < 96.591,0.265 > and < 94.234,0.240 > (Table 15). The analysis

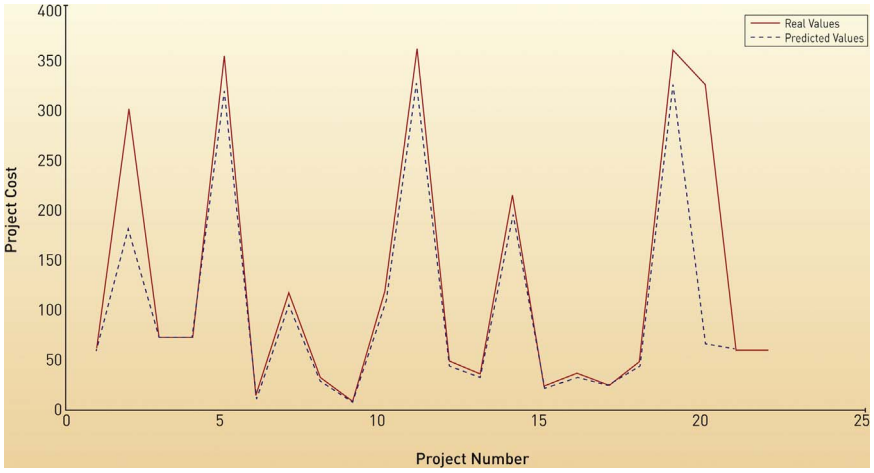

Fig. 4. Prediction results for Nasa93 data set (test set) using DABE-3 algorithm: actual values (solid lines) and predicted values (dashed lines).

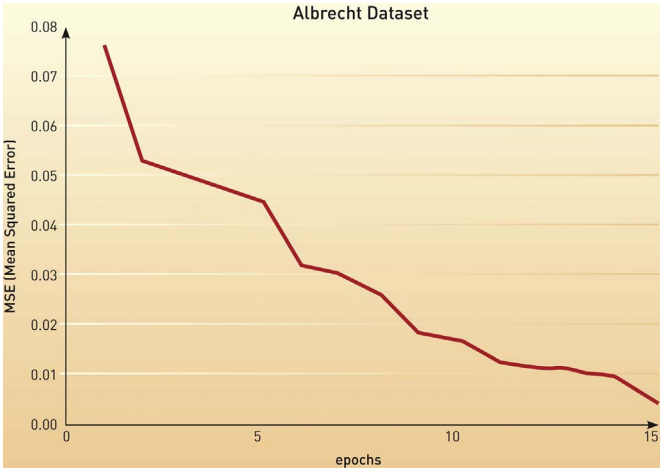

Fig. 5. Mean square error versus iteration – Albrecht dataset.

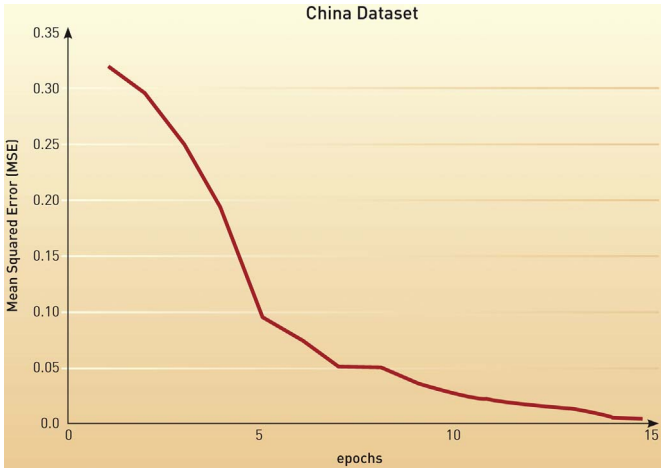

Fig. 7. Mean square error versus iteration – China dataset.

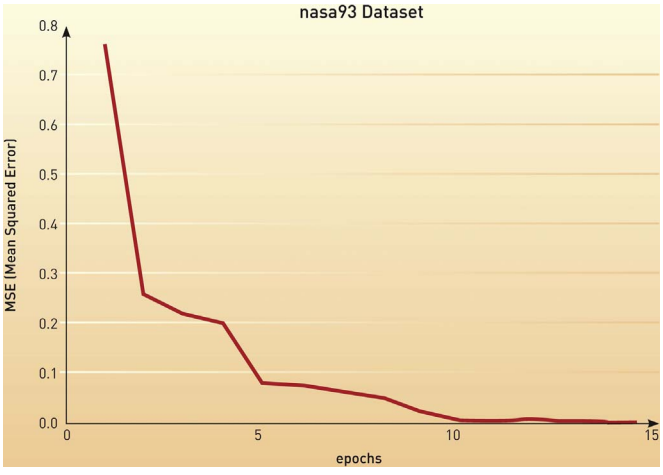

Fig. 6. Mean square error versus iteration – Nasadataset.

report based on SA and Δ for each table clearly indicated that DABE-3 exhibits promising results. Table 16 summarizes EF of the most suitable variants of all effort estimation techniques for Desharnais dataset. Among the DABE variants, DABE-3 provides highest testing performance with an EF value of 98.328.

Fig. 4 illustrates the plot of real (solid line) and predicted (dashed line) values generated by the proposed DABE model, specifically

Table 17
Wilcoxon signed rank test results.

Comparison	R+	R-	p-value
DABE–3 versus ABE	21	0	0.0432
DABE–3 versus GA-ABE	17	4	0.2251
DABE–3 versus PSO-ANE	21	0	0.0432
DABE–3 versus ANN	11	10	0.0689
DABE–3 versus FLANN	11	10	0.0689
DABE–3 versus RBF	21	0	0.0432
DABE–3 versus JADE-ABE	18	3	0.0541
DABE–3 versus SADE-ABE	20	1	0.0475

Note: DE exhibited more satisfactory performance compared with ABE, PSO, RBF, SADE, and JADE, with α = 0.05; ANN and FLANN, with α = 0.1, and GA, with α = 0.5.

DABE-3, because it has the highest of SA and Δvalues for the Nasa93 data set. Figs. 5–7 illustrate the errors calculated during different epochs of the experimental study of DABE-3, indicating that at least 15 iterations are required for reaching an error in a stable state for Albrecht, Nasa93, and China datasets. Although all datasets have inherent non-normal characteristics, DABE could reduce the local optima and provide promising accuracy.

7. Statistical performance evaluation

Because the SDEE datasets exhibit heteroscedasticity (i.e., each subpopulation in the dataset has non constant variance), we applied

Table 18
Error values of various prediction models.

Error	ABE	GA-ABE	DABE-3	PSO-ABE	ANN	FLANN	RBF	JADE-ABE	SADE-ABE
Albrecht	0.221	0.018	0.017	0.018	0.020	0.024	0.122	0.018	0.016
cocomo81	0.116	0.100	0.018	0.029	0.016	0.035	0.246	0.014	0.129
China	0.826	0.097	0.050	0.861	0.016	0.004	0.717	0.016	0.047
Desharnais	0.048	0.042	0.023	0.033	0.011	0.066	0.534	0.066	0.031
Maxwell	0.091	0.100	0.010	0.257	0.057	0.035	0.654	0.500	0.016
nasa93	0.144	0.009	0.074	0.099	0.005	0.002	0.434	0.099	0.063

Table 19
Represents the ranks evaluated through the Friedman test.

Friedman Ranks	ABE	GA-ABE	DABE-3	PSO-ABE	ANN	FLANN	RBF	JADE-ABE	SADE-ABE
Albrecht	9.0	4.0	2.0	4.0	6.0	7.0	8.0	4.0	1.0
cocomo81	7.0	6.0	3.0	4.0	2.5	5.0	9.0	1.5	8.0
China	8.0	6.0	5.0	9.0	2.5	1.0	7.0	2.5	4.0
Desharnais	6.0	5.0	2.0	4.0	1.0	7.5	9.0	7.5	3.0
Maxwell	5.0	6.0	1.0	7.0	4.0	3.0	9.0	8.0	2.0
nasa93	8.0	3.0	5.0	6.5	2.0	1.0	9.0	6.5	4.0
Average	7.2	5.0	3.0	5.8	3.0	4.1	8.5	5.0	3.7

Note: DE and RBF were the best and worst performing algorithms, respectively.

nonparametric statistical tests to evaluate the performance of the prediction models. When analogizing the prediction models, contemplating the degree to which the No Free Lunch Theorem [101] restricts the conclusion—under no limited comprehension, any two algorithms are identical when their performance is averaged across all plausible problems—is imperative.

To implement nonparametric tests to a multi-model framework, a result per algorithm model pair must be rendered. This is commonly estimated as the average of a specified performance measure over several runs and performed on every single model. A typical instance could be the average error on 100 runs of an algorithm over 15 benchmark functions.

A null or no-effect hypothesis should be developed before implementing the test. This frequently confirms the equivalence or absence of differences among the results of the algorithms and facilitates alternative hypotheses supporting the opposite situation to be evaluated [102]. The null and alternative hypotheses are characterized by H_0, and the alternative hypotheses by and $H_1, \ldots H_n$, respectively. The application of this test aids in computation of statistics, which can be used to reject the hypothesis at a given level of significance α.

For a fine-grained analysis, estimating the smallest level of significance that culminates in the rejection of the null hypothesis is desirable. This level is the p value, which is the probability of attaining a conclusion at least as extreme as the one that was indeed ascertained, presuming that the null hypothesis is true. The use of p value is often recommended over using only fixed α levels because it yields a clearer estimate of the significance of the result (smaller the p value, stronger is the substantiation against the null hypothesis) [106].

The nonparametric tests can be categorized according to their abilities to enforce pair-wise and multiple comparisons. Indicating that the p values realized through pair-wise and multiple comparisons are self-reliant is essential; thus, multiple comparison methodologies should be adopted when analyzing more than two algorithms [104,105].

Various nonparametric tests can be applied to correlate the final results of the prediction models: The Wilcoxon signed-rank test can assist in performing pair-wise comparisons, whereas the Friedman test can be used for conducting multiple comparisons [103,104].

Here, we considered the following null hypothesis:
H_o: All prediction models are equivalent.

Our statistical analysis was divided into three segments. In the first segment, we presented the results obtained by Wilcoxon signed-rank test, followed by illustrating the results of Friedman test in the second segment and highlighting the null hypothesis procedure by using the Friedman test in the final segment.

Table 17 shows all pair-wise comparisons concerning the DE algorithm: DABE-3 demonstrated a significant improvement over ABE, PSO-ABE, SaDE-ABE, JADE-ABE, and RBF, with α = 0.05, and over ANN and FLANN, with α = 0.1.

The objective of the Friedman test [107,108] is to infer the presence of difference among various prediction models for deducted sample of results (error values depicted in Table 18) found with respect to the given datasets. The first step in estimating the test statistic involves changing the original results to ranks; thus, the algorithms are ranked for each dataset independently, and the best performing algorithm is assigned the rank of 1, the second best is assigned rank 2, and so on. Table 19 presents the ranks computed using the Friedman test. The results indicated that DABE-3 and RBF were the best and worst performing algorithms, respectively.

To test our null hypothesis, we incorporated the Freidman test reported by Demšar [105] and Garćia et al. [103]. As per the study conducted by the authors, we assign ranks r_i^j to the jth of k prediction models on the ith of N datasets based on their predictive accuracy. We then obtained the Freidman statistic (F_F) given by equation

$$F_F = \frac{(N-1)\chi_F^2}{N(K-1) - \chi_F^2} \tag{21}$$

where

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{22}$$

The χ^2value was found to be 20.57; in turn, F_F was 6.6667. By using nine prediction models and six datasets, F_F was distributed in compliance with F-distribution, with 7 − 1 = 6 and (9 − 1) × (6 − 1) = 40 degrees of freedom. The critical value of F (5, 40) for α = 0.01 was 3.514—less than the obtained F_F. Thus, we can reject our null hypothesis H_o.

8. Conclusions

We have proposed a feature weight optimization technique called DABE model ABE. Our proposed DABE was tested rigorously on the PROMISE repository test suit with various mutation schema suggested

in DE algorithm literature. A few interesting observations for DABE are the following: The mutation strategy "DE/rand/2/bin" was a version efficient for SDEE. Our experimental studies demonstrated that the performance of the proposed DABE model for ABE was promising. The performance measures SA, Δ, and EF is the unique global error indicators added in our study for assessing the prediction model performance. In most cases, the results obtained using the DABE model was as satisfactory as or more satisfying than the most favorable results demonstrated by the PSO-ABE, GA-ABE, SaDE-ABE, JADE-ABE, FLANN, ANN, and RBF. The inherent advantages of the DE algorithm are efficient memory utilization, lower computational complexity, and lower computational effort, making it attractive for researchers working swarm intelligence application to SDEE tasks. In this study, we have focused on a fixed value of control parameters and single objective function, but some objective functions are extremely sensitive to the appropriate selection of control parameters. Hence, in future works, we would explore state-of-art DE schemes such as cultural algorithm and differential evolution [129]. Furthermore; we would incorporate DE-based multi-objective optimization techniques for software effort prediction using ABE. Also, we would investigate our proposed DABE model with simultaneous project selection and feature weight optimization. MMRE and PRED were used for objective function design and criticized as biased error indicators. Objective function consisting of SA and Δ must be constructed and its performance investigated.

References

[5] R. Storn, K. Price, Differential Evolution – A Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces 3, ICSI, Berkeley, 1995.
[6] R. Storn, K.V. Price Minimizing the real functions of the ICEC'96 contest by differential evolution, in: International Conference on Evolutionary Computation pp. 842–844, 1996.
[7] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (4) (1997) 341–359.
[8] A. Slowik, Application of an adaptive differential evolution algorithm with multiple trial vectors to artificial neural network training, IEEE Trans. Ind. Electron. 58 (8) (2011) 3160–3167.
[11] R. Kohavi, G.H. John, Wrappers for feature subset selection, Artif. Intell. 97 (1) (1997) 273–324.
[19] M. Jørgensen, Top-down and bottom-up expert estimation of software development effort, Inf. Softw. Technol. 46 (1) (2004) 3–16.
[21] M. Shin, A.L. Goel, Empirical data modeling in software engineering using radial basis functions, IEEE Trans. Softw. Eng. 26 (6) (2000) 567–576.
[23] M. Shepperd, C. Schofield, Estimating software project effort using analogies, IEEE Trans. Softw. Eng. 23 (11) (1997) 736–743.
[24] M. Auer, A. Trendowicz, B. Graser, E. Haunschmid, S. Biffl, Optimal project feature weights in analogy-based cost estimation: improvement and limitations, IEEE Trans. Softw. Eng. 32 (2) (2006) 83–92.
[25] S.J. Huang, N.H. Chiu, Optimization of analogy weights by genetic algorithm for software effort estimation, Inf. Softw. Technol. 48 (11) (2006) 1034–1045.
[32] R.J. Sternberg, Component processes in analogical reasoning, Psychol. Rev. 84 (4) (1977) 353.
[33] F. Walkerden, R. Jeffery, An empirical study of analogy-based software effort estimation, Empir. Softw. Eng. 4 (2) (1999) 135–158.
[34] L. Angelis, I. Stamelos, A simulation tool for efficient analogy based cost estimation, Empir. Softw. Eng. 5 (1) (2000) 35–68.
[35] S.J. Huang, N.H. Chiu, Optimization of analogy weights by genetic algorithm for software effort estimation, Inf. Softw. Technol. 48 (11) (2006) 1034–1045.
[36] N.H. Chiu, S.J. Huang, The adjusted analogy-based software effort estimation based on similarity distances, J. Syst. Softw. 80 (4) (2007) 628–640.
[37] Y.F. Li, M. Xie, T.N. Goh, A study of project selection and feature weighting for analogy based software cost estimation, J. Syst. Softw. 82 (2) (2009) 241–252.
[38] V.K. Bardsiri, D.N.A. Jawawi, S.Z.M. Hashim, E. Bardsiri, A PSO-based model to increase the accuracy of software development effort estimation, Softw. Qual. J. 21 (3) (2013) 501–526.
[47] M. Shepperd, G. Kadoda, Comparing software prediction techniques using simulation, IEEE Trans. Softw. Eng. 27 (11) (2001) 1014–1022.
[48] J. Kolodner, Case-based Reasoning, Morgan Kaufmann, 2014.
[49] M. Jørgensen, U. Indahl, D. Sjøberg, Software effort estimation by analogy and "regression toward the mean", J. Syst. Softw. 68 (3) (2003) 253–262.
[50] M. Jorgensen, M. Shepperd, A systematic review of software development cost estimation studies, IEEE Trans. Softw. Eng. 33 (1) (2007).
[51] G. Kadoda, M. Cartwright, L. Chen, M. Shepperd, . Experiences using case-based reasoning to predict software project effort, in: Proceedings of the EASE 2000 conference, Keele, UK, 2000.
[52] T. Foss, E. Stensrud, B. Kitchenham, I. Myrtveit, A simulation study of the model evaluation criterion MMRE, IEEE Trans. Softw. Eng. 29 (11) (2003) 985–995.
[53] I. Myrtveit, E. Stensrud, M. Shepperd, Reliability and validity in comparative studies of software prediction models, IEEE Trans. Softw. Eng. 31 (5) (2005) 380–391.
[54] M. Shepperd, S. MacDonell, Evaluating prediction systems in software project estimation, Inf. Softw. Technol. 54 (8) (2012) 820–827.
[55] R.D.A. Araújo, A.L. Oliveira, S. Soares, A shift-invariant morphological system for software development cost estimation, Expert Syst. Appl. 38 (4) (2011) 4162–4168.
[56] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, Evolut. Comput. IEEE Trans. 15 (1) (2011) 4–31.
[58] K. Price, R.M. Storn, J. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, U.S. Government Printing Office, 2005.
[59] A.L. Custodio, J.F.A. Madeira, A.I.F. Vaz, L.N. Vicente, Direct multisearch for multiobjective optimization, SIAM J. Optim. 21 (3) (2011) 1109–1140.
[61] M. Azzeh, D. Neagu, P.I. Cowling, Analogy-based software effort estimation using Fuzzy numbers, J. Syst. Softw. 84 (2) (2011) 270–284.
[62] M. Azzeh, D. Neagu, P.I. Cowling, Fuzzy grey relational analysis for software effort estimation, Empir. Softw. Eng. 15 (1) (2010) 60–90.
[64] M. Jørgensen, U. Indahl, D. Sjøberg, Software effort estimation by analogy and "regression toward the mean", J. Syst. Softw. 68 (3) (2003) 253–262.
[65] M. Azzeh, August). Model tree based adaption strategy for software effort estimation by analogy, in: Proceedings of the 11th International Conference on Computer and Information Technology (CIT), 2011 IEEE pp. 328–335. IEEE, 2011.
[66] N.H. Chiu, S.J. Huang, The adjusted analogy-based software effort estimation based on similarity distances, J. Syst. Softw. 80 (4) (2007) 628–640.
[67] I. Stamelos, L. Angelis, Managing uncertainty in project portfolio cost estimation, Inf. Softw. Technol. 43 (13) (2001) 759–768.
[68] N. Mittas, M. Athanasiades, L. Angelis, Improving analogy-based software cost estimation by a resampling method, Inf. Softw. Technol. 50 (3) (2008) 221–230.
[69] A. Tosun, B. Turhan, A.B. Bener, Feature weighting heuristics for analogy-based effort estimation models, Expert Syst. Appl. 36 (7) (2009) 10325–10333.
[71] T. Menzies, B. Caglayan, E. Kocaguneli, J. Krall, F. Peters, B. Turhan, The PROMISE Repository of Empirical Software Engineering Data, Department of Computer Science, West Virginia Universit, 2012 ⟨http://openscience.us/repo/⟩.
[72] K. Dejaeger, W. Verbeke, D. Martens, B. Baesens, Data mining techniques for software effort estimation: a comparative study, IEEE Trans. Softw. Eng. 38 (2) (2012) 375–397.
[73] E. Kocaguneli, T. Menzies, Software effort models should be assessed via leave-one-out validation, J. Syst. Softw. 86 (7) (2013) 1879–1890.
[78] T.R. Benala, S. Dehuri, R. Mall, Computational intelligence in software cost estimation: an emerging paradigm, ACM SIGSOFT Softw. Eng. Notes 37 (3) (2012) 1–7.
[80] T.R. Benala, S. Dehuri, S.C. Satapathy, S. Madhurakshara. Genetic algorithm for optimizing functional link artificial neural network based software cost estimation, in: Proceedings of the International Conference on Information Systems Design and Intelligent Applications 2012 (INDIA 2012) held in Visakhapatnam, India, January 2012 pp. 75–82. Springer Berlin Heidelberg, 2012b.
[81] T.R. Benala, R. Mall, S. Dehuri, V.L. Prasanthi, December). Software effort prediction using fuzzy clustering and functional link artificial neural networks, in: International Conference on Swarm, Evolutionary, and Memetic Computing pp. 124–132. 2012, Springer Berlin Heidelberg, 2012c.
[84] P.C. Mahalanobis, On the generalized distance in statistics, Proc. Natl. Inst. Sci. 2 (1936) 49–55.
[85] J. Li, G. Ruhe, A. Al-Emran, M.M. Richter, A flexible method for software effort estimation by analogy, Empir. Softw. Eng. 12 (1) (2007) 65–106.
[88] C. López-Martín, Predictive accuracy comparison between neural networks and statistical regression for development effort of software projects, Appl. Softw. Comput. 27 (2015) 434–449.
[89] I.F. de Barcelos Tronto, J.D.S. da Silva, N. Sant'Anna, An investigation of artificial neural networks based prediction systems in software project management, J. Syst. Softw. 81 (3) (2008) 356–367.
[90] H. Park, S. Baek, An empirical validation of a neural network model for software effort estimation, Expert Syst. Appl. 35 (3) (2008) 929–937.
[91] R. Lippmann, An introduction to computing with neural nets, IEEE Assp Mag. 4 (2) (1987) 4–22.
[92] A. Tosun, B. Turhan, A.B. Bener, Feature weighting heuristics for analogy-based effort estimation models, Expert Syst. Appl. 36 (7) (2009) 10325–10333.
[93] K.V. Price, R.M. Storn, J.A. Lampinen, Differential Evolution—A Practical Approach to Global Optimization, Springer-Verlag, Berlin, Germany, 2005.
[94] D. Karaboga, S. Okdem, A simple and global optimization algorithmfor engineering problems: differential evolution algorithm, Turk. J. Elect. Eng. Comput. Sci. 12 (1) (2004) 53–60.
[95] K. Price, R. Storn, J. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, Springer, Berlin, Germany, 2005.
[96] S. Hui, P.N. Suganthan, Ensemble and Arithmetic Recombination-Based Speciation Differential Evolution for Multimodal Optimization, 2015.
[97] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, Evolut. Comput. IEEE Trans. 13 (2) (2009) 398–417.
[98] R. Storn, System design by constraint adaptation and differential evolution, IEEE Trans. Evol. Comput. 3 (1) (1999) 22–34.
[99] R. Storn, K. Price, Differential Evolution-a Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces 3, ICSI, Berkeley, 1995.

[100] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (4) (1997) 341–359.
[101] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, Evolut. Comput. IEEE Trans. 1 (1) (1997) 67–82.
[102] D.J. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, CRC Press, 2003.
[103] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Inf. Sci. 180 (10) (2010) 2044–2064.
[104] J. Derrac, S. García, S. Hui, P.N. Suganthan, F. Herrera, Analyzing convergence performance of evolutionary algorithms: a statistical approach, Inf. Sci. 289 (2014) 41–58.
[105] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[106] J.H. Zar, Biostatistical Analysis, Pearson Education, India, 1999.
[107] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, J. Am. Stat. Assoc. 32 (200) (1937) 675–701.
[108] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, Ann. Math. Stat. 11 (1) (1940) 86–92.
[111] C.S.K. Dash, P. Sahoo, S. Dehuri, S.B. Cho, An empirical analysis of evolved radial basis function networks and support vector machines with mixture of kernels, Int. J. Artif. Intell. Tools 24 (4) (2015) 1550013.
[112] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy daptation for global numerical optimization, Evolut. Comput., IEEE Trans. 13 (2) (2009) 398–417.
[113] K. Price, R.M. Storn, J.A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series), 1st ed., Springer, New York, 2005.
[114] W.Y. Gong, Z.H. Cai, C.X. Ling, H. Li, Enhanced differentialevolution with adaptive strategies for numerical optimization, IEEE Trans. Syst. Man Cybern. B Cybern. 41 (2) (2011) 397–413.
[115] J.H. Zhong, M. Shen, J. Zhang, H.S.H. Chung, Y.H. Shi, Y. Li, A differential evolution algorithm with dual populations for solving periodic railway timetable scheduling problem, Evolut. Comput., IEEE Trans. 17 (4) (2013) 512–527.
[116] E. Mezura-Montes, J. Velázquez-Reyes, C.A. CoelloCoello, , A comparative study of differential evolution variants for global optimization, in: Proceedings of the 8th annual conference on Genetic and evolutionary computation pp. 485–492. ACM, 2006.
[118] S. Das, S.S. Mullick, P.N. Suganthan, Recent advances in differential evolution – an updated survey, Swarm Evolut. Comput. (2016).
[119] M. Azzeh, A replicated assessment and comparison of adaptation techniques for analogy-based effort estimation, Empir. Softw. Eng. 17 (1–2) (2012) 90–127.
[121] A. Idri, F. azzahraAmazal, A. Abran, Analogy-based software development effort estimation: a systematic mapping and review, Inf. Softw. Technol. 58 (2015) 206–230.
[122] M. Azzeh, A.B. Nassif, S. Banitaan, F. Almasalha, Pareto efficient multi-objective optimization for local tuning of analogy-based estimation, Neural Comput. Appl. (2015) 1–25.
[123] M. Azzeh, A.B. Nassif, L.L. Minku, An empirical evaluation of ensemble adjustment methods for analogy-based effort estimation, J. Syst. Softw. 103 (2015) 36–52.
[124] F. Peñuñuri, C. Cab, O. Carvente, M.A. Zambrano-Arjona, J.A. Tapia, A study of the classical differential evolution control parameters, Swarm Evolut. Comput. (2015).
[125] B. Hughes, M. Cotterell, Rajib Mall, Software Project Management, Tata McGraw-Hill Education, 2011.
[126] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evolut. Comput. 13 (2) (2009) 398–417.
[129] N.H. Awad, M.Z. Ali, P.N. Suganthan, R.G. Reynolds, CADE: a hybridization of cultural algorithm and differential evolution for numerical optimization, Inf. Sci. 378 (2017) 215–241.
[130] A.P. Piotrowski, Review of differential evolution population size, Swarm Evolut. Comput. (2016).
[131] S.M. Guo, C.C. Yang, P.H. Hsu, J.S.H. Tsai, Improving differential evolution with a successful-parent-selecting framework, IEEE Trans. Evolut. Comput. 19 (5) (2015) 717–730.
[132] F. Penunuri, C. Cab, O. Carvente, M.A. Zambrano-Arjona, J.A. Tapia, A study of the classical differential evolution control parameters, Swarm Evolut. Comput. 26 (2016) 86–96.
[133] S. Rahnamayan, H.R. Tizhoosh, M.M. Salama, Opposition-based differential evolution, IEEE Trans. Evolut. Comput. 12 (1) (2008) 64–79.
[135] Y. Wang, Z. Cai, Q. Zhang, Enhancing the search ability of differential evolution through orthogonal crossover, Inf. Sci. 185 (1) (2012) 153–177.
[136] S.M. Islam, S. Das, S. Ghosh, S. Roy, P.N. Suganthan, An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization, IEEE Trans. Syst. Man Cybern. Part B (Cybern.) 42 (2) (2012) 482–500.
[137] W. Gong, Z. Cai, C.X. Ling, H. Li, Enhanced differential evolution with adaptive strategies for numerical optimization, IEEE Trans. Syst. Man Cybern. Part B (Cybern.) 41 (2) (2011) 397–413.
[138] W. Gong, Z. Cai, Y. Wang, Repairing the crossover rate in adaptive differential evolution, Appl. Softw. Comput. 15 (2014) 149–168.
[139] J. Aalto, J. Lampinen, , A mutation adaptation mechanism for differential evolution algorithm, in: 2013 IEEE Congress on Evolutionary Computation, 2013. pp. 55–62. IEEE.
[140] S.Z. Zhao, P.N. Suganthan, S. Das, Self-adaptive differential evolution with multi-trajectory search for large-scale optimization, Softw. Comput. 15 (11) (2011) 2175–2185.
[141] W. Gong, Á. Fialho, Z. Cai, H. Li, Adaptive strategy selection in differential evolution for numerical optimization: an empirical study, Inf. Sci. 181 (24) (2011) 5364–5386.
[142] R. Mallipeddi, P.N. Suganthan, Q.K. Pan, M.F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Appl. Softw. Comput. 11 (2) (2011) 1679–1696.
[143] Q.K. Pan, P.N. Suganthan, L. Wang, L. Gao, R. Mallipeddi, A differential evolution algorithm with self-adapting strategy and control parameters, Comput. Oper. Res. 38 (1) (2011) 394–408.
[144] L.L. Minku, X. Yao, Software effort estimation as a multiobjective learning problem, ACM Trans. Softw. Eng. Methodol. (TOSEM) 22 (4) (2013) 35.
[145] A. Idri, M. Hosni, A. Abran, Improved estimation of software development effort using classical and fuzzy analogy ensembles, Appl. Softw. Comput. 49 (2016) 990–1019.
[146] A. Idri, I. Abnane, A. Abran, Missing data techniques in analogy-based software development effort estimation, J. Syst. Softw. 117 (2016) 595–611.
[147] S.H.S. Moosavi, V.K. Bardsiri, Satin bowerbird optimizer: a new optimization algorithm to optimize ANFIS for software development effort estimation, Eng. Appl. Artif. Intell. 60 (2017) 1–15.