IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 17 April 2024, accepted 5 May 2024, date of publication 8 May 2024, date of current version 17 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3399060

RESEARCH ARTICLE

Enhancing Software Effort Estimation With Self-Organizing Migration Algorithm: A Comparative Analysis of COCOMO Models

DARINA BAJUSOVA, PETR SILHAVY, AND RADEK SILHAVY
Faculty of Applied Informatics, Tomas Bata University in Zlín, 760 01 Zlin, Czech Republic

Corresponding author: Petr Silhavy (psilhavy@utb.cz)

This work was supported in part by the Faculty of Applied Informatics, Tomas Bata University in Zlín, under Grant RVO/FAI/2024/002 and Grant IGA/CebiaTech/2023/004.

ABSTRACT This study presents a comprehensive analysis of enhancing software effort estimation accuracy using a Self-Organizing Migration Algorithm (SOMA)-optimized Constructive Cost Model (COCOMO). By conducting a comparative study of traditional COCOMO models and SOMA-optimized variants across preprocessed datasets (NASA93, NASA63, NASA18, Kemerer, Miyazaki94, and Turkish), our research focuses on crucial evaluation metrics, including Mean Magnitude of Relative Error (MMRE), Prediction at 0.25 (PRED(0.25)), Mean Absolute Error (MAE), and Root Mean Square Error (RMSE). The analysis encompasses various configurations of COCOMO models—basic, intermediate, and post-architecture COCOMO II, supplemented with additional statistical testing and residual analysis for in-depth insights. The results demonstrate that the SOMA-optimized COCOMO models generally surpass traditional models in predictive accuracy, especially notable in metrics such as MMRE where an improvement of up to 12%, PRED(0.25) with an enhancement of 15%, MAE reduction by 18%, and a decrease in RMSE by 20% were observed. However, performance variances were identified in specific scenarios, highlighting areas for further refinement, particularly in large-scale estimations where residual plots suggested the potential for underestimation or overestimation. The study concludes that integrating the SOMA optimization algorithm into COCOMO models significantly enhances the accuracy of software effort estimations, providing valuable insights for future research to optimise estimations for larger projects and advance prediction models. This advancement addresses the technical challenge of parameter accuracy and offers a methodological improvement in model selection and application, underscoring the potential of metaheuristic optimization in software effort estimation.

INDEX TERMS Software effort estimation, COCOMO models, SOMA, metaheuristic optimization.

I. INTRODUCTION

The escalating complexity and magnitude of contemporary software development endeavours have significantly underscored the critical role of precise software effort estimation. Accurate estimations are indispensable for enabling informed decision-making and fostering successful project management. Despite a wide spectrum of available estimation methodologies, the persistent challenge remains in achieving precise predictions, with a common propensity towards either overestimation or underestimation of project efforts. The latter scenario often precipitates project failure due to constraints in time and budget [1], [2].

Within the extensive array of estimation models, algorithmic constructs such as the Constructive Cost Model (COCOMO) and its progeny, COCOMO II, continue to be favoured for their simplicity and versatile applicability across diverse project stages [3], [4]. Nevertheless, the accuracy

The associate editor coordinating the review of this manuscript and approving it for publication was Pinjia Zhang.

of these models is heavily reliant on the precision of input parameters, a dependency that can engender significant errors in estimation. To facilitate the reliability of these models, machine learning techniques, particularly metaheuristic algorithms, have been harnessed to refine parameter values, thereby enhancing estimation accuracy [5].

Metaheuristic algorithms are celebrated for their proficiency in identifying near-optimal solutions efficiently, rendering them highly suitable for addressing the intricate optimization challenges encountered in software effort estimation. These algorithms are dichotomized into single-solution-based and population-based methods, with the latter category, epitomized by strategies such as the Self-Organizing Migrating Algorithm (SOMA), demonstrating exceptional efficacy owing to their evolutionary approach to solution refinement [6].

A. CHALLENGES IN ACCURATE SOFTWARE EFFORT ESTIMATION

The endeavour to accurately estimate software effort is beleaguered by challenges emanating from software development projects' intrinsic variability and complexity. The predicament resides in the selection and implementation of estimation models. Although algorithmic models like COCOMO and COCOMO II proffer structured methodologies, their effectiveness is critically dependent on the precision of input parameters. Erroneous estimations of these parameters can precipitate substantial inaccuracies, further exacerbated by the complexity inherent in the estimated projects.

Moreover, human factors exert a significant influence. Estimation transcends a mere quantitative exercise, encompassing subjective judgments made by estimators. Cognitive biases, insufficient experience, and the exigencies imposed by business or market expectations can all culminate in overly optimistic or unduly conservative estimates.

Notwithstanding their recognized limitations, the industry's reliance on traditional estimation models remains steadfast. These models frequently neglect the non-linear and non-deterministic dimensions of software development, engendering either too optimistic or overly pessimistic estimates. Herein lies the potential for enhancement through metaheuristic algorithms like SOMA, which provide a more adaptable and robust mechanism for parameter optimization.

B. OBJECTIVES OF THE STUDY

The motivation for this study is the limited application of advanced metaheuristic optimization algorithms, precisely the Self-Organizing Migration Algorithm (SOMA), in improving the accuracy and adaptability of software effort estimation models like COCOMO. Previous efforts predominantly focused on conventional optimization techniques and did not thoroughly explore the comparative effectiveness of various models enhanced by advanced algorithms.

This study explores and addresses software effort estimation challenges by leveraging the SOMA optimization algorithm. SOMA exhibits distinctive features that are particularly well-suited to software effort estimation. Its ability to efficiently navigate complex, multidimensional search spaces and avoid local optima makes it an excellent choice for optimizing the COCOMO model parameters. These characteristics ensure a more thorough exploration of potential solutions, enhancing the likelihood of identifying optimal or near-optimal parameters for effort estimation. The study aims to:

1) Evaluate the impact of the SOMA optimization algorithm on the accuracy of effort estimation in the COCOMO and COCOMO II models, thereby addressing the technical challenge of parameter accuracy.
2) Determine whether the optimization of constants (a, b) in these models using SOMA can lead to statistically significant improvements in estimation precision, thus tackling the methodological challenge of model selection and application.
3) Compare the performance of the SOMA-optimized models against traditional models and other metaheuristic algorithms to establish a benchmark for estimation accuracy.
4) Enhance the understanding of how metaheuristic algorithms can mitigate the human factor challenge by reducing the reliance on subjective judgment and providing a more data-driven, objective approach to estimation.
5) Contribute to the body of knowledge in software effort estimation by providing empirical evidence of the effectiveness of metaheuristic optimization in improving estimation outcomes.

This study introduces a novel approach to software effort estimation by integrating the Self-Organizing Migration Algorithm (SOMA) with the Constructive Cost Model (COCOMO) and its successor, COCOMO II. The fusion of SOMA with these well-established models aims to address the persistent challenges in software effort estimation, notably the accuracy of parameter estimation and the adaptation of models to contemporary software development practices. The novelty of this research lies in several key areas:

1) Advanced Optimization with SOMA: Unlike traditional efforts focusing on linear or manual optimization techniques for model parameters, this study leverages SOMA, a metaheuristic optimization algorithm, for dynamic and sophisticated COCOMO and COCOMO II model parameters. This is one of the first studies to systematically analyze the impact of SOMA on these models, particularly targeting the optimization of the constants a and b, which are crucial for the models' accuracy.
2) Comprehensive Comparative Analysis: By conducting a detailed comparison of SOMA-optimized models against traditional COCOMO models and other metaheuristic algorithms, this research provides a broad

perspective on the performance of various optimization strategies. Including numerous datasets, such as NASA93, NASA63, NASA18, Kemerer, Miyazaki94, and Turkish, ensures the robustness and generalizability of the findings.

3) Contribution to Body of Knowledge: Beyond immediate practical applications, this research enriches the academic discourse on software effort estimation by offering empirical evidence of metaheuristic optimization's potential benefits. It opens new avenues for future research, particularly in optimizing estimations for larger projects and further advancing prediction models.

C. RESEARCH QUESTIONS AND HYPOTHESES

This research aims to address the following questions:

- RQ1: How does the SOMA optimization algorithm impact effort estimation accuracy in COCOMO and COCOMO II models?
- RQ2: Can optimising constants (a, b) in these models using SOMA lead to statistically significant improvements in estimation precision?

Based on these questions, the study hypothesizes that:

- H0: Applying the SOMA optimization algorithm to COCOMO and COCOMO II models does not result in more accurate software effort estimations (MMRE, PRED(0.25), MAE, and RMSE) than the original models.
- H1: Applying the SOMA optimization algorithm to COCOMO and COCOMO II models will result in more accurate software effort estimations (MMRE, PRED(0.25), MAE, and RMSE) than the original models.

To investigate these hypotheses, a comprehensive analysis was conducted using NASA93, NASA63, NASA18, Kemerer, Miyazaki94, and Turkish datasets. The MMRE metric was employed as a fitness function to evaluate the performance of the optimized models, supplemented by a 10-fold cross-validation method to prevent overfitting. The accuracy of the SOMA-optimized models was compared with that of the original COCOMO and COCOMO II models, as well as other metaheuristic algorithms and traditional models, using a range of performance metrics.

D. PAPER ORGANIZATION

The remainder of this paper is systematically organized into several sections: Section II reviews related works in the field. Section III details the methodology used (COCOMO and COCOMO II models, SOMA metaheuristic algorithm, datasets description, evaluation metrics, and statistical analysis). The proposed optimization approach is elaborated in Section IV, followed by an analysis of the results in Section V. Finally, Section VI concludes the paper with a summary of the findings and an outline of future research directions.

II. RELATED WORKS

The endeavor to accurately predict software development costs has spurred numerous studies on effective estimation methods. Recently, metaheuristic algorithms have gained prominence for enhancing the precision of established models such as COCOMO and COCOMO II. The body of literature encompasses a broad array of research that evaluates these conventional models and explores the augmentation of their accuracy through metaheuristics. This section surveys pivotal contributions to software effort estimation, focusing on algorithmic methodologies and metaheuristic algorithms.

Fadhil et al. [7] explored refining the COCOMO II model's estimation accuracy by applying the dolphin algorithm and introducing a hybrid dolphin and bat algorithm (DolBat). Their models underwent testing on the NASA93 and NASA60 datasets, assessed via the MMRE metric, and were benchmarked against the original COCOMO II model, the Genetic algorithm, the hybrid Cuckoo Optimization and Harmony Search algorithm (CSHS), and the Bat algorithm. Parwita et al. [8] also aimed at optimizing COCOMO II model coefficients using the Cuckoo Optimization Algorithm (COA), with their analysis conducted on the Turkish Software Dataset and comparisons drawn with the Fuzzy Local Calibration Model and the original COCOMO II model. Langsari and Sarno [9] utilized the same dataset for COCOMO II model optimization, focusing on the Particle Swarm Optimization (PSO) algorithm, which demonstrated superiority over the original COCOMO II coefficients and those generated by the Tabu Search algorithm. Sachan et al. [10] assessed the impact of a simplified genetic algorithm on optimizing the basic COCOMO model parameters, applying their methodology to the NASA18 dataset and employing Manhattan Distance (MD) as the fitness function.

Alsheikh and Munassar [11], in their research, aimed to improve the accuracy of effort estimation by adjusting the coefficients of three COCOMO-based models: basic COCOMO and its two modifications proposed by Sheta [12]. The authors conducted tests on NASA18 dataset and applied Grey Wolf Optimization (GWO), alongside four other optimization algorithms (Zebra Optimization, Moth-Flame Optimization, Prairie Dog Optimization, and White Shark Optimization). Evaluation metrics, including VAF, MSE, MAE, MMRE, RMSE, and R^2, were used to gauge the effectiveness of the optimized models, with results highlighting GWO's superiority over Zebra Optimization, Moth-Flame Optimization, Prairie Dog Optimization, White Shark Optimization. Results were also compared against the Firefly algorithm, Genetic algorithm, and PSO algorithm.

Gandomani et al. [13] used a hybrid GA-EA approach (Genetic algorithm (GA) and Environmental Adaptation (EA) techniques) to enhance the efficacy of the basic COCOMO model. The integration of EA into GA effectively addresses GA's convergence issue. The investigation utilizes the NASA93 dataset, allocating 80% of the data for training and 20% for testing. A comparative analysis was conducted

using the original COCOMO model, EA algorithm, and hybrid model GA-EA. The findings indicate that the hybrid GA-EA approach got the best results: MMRE was equal to 53%, PRED(0.25) was 23%, and evaluation function (EF) was 0.42. EF was defined as the ratio between PRED(0.25) and (1+MMRE).

Anwar ul Hassan and Sufyan Khan [14] tested the efficiency of the Strawberry Algorithm (SBA), Gray Wolf Algorithm (GWO), and Harmony Search Algorithm (HSA). These optimization techniques were applied to the basic COCOMO model using the NASA93 dataset and a 3-fold cross-validation approach. The experiment considered the whole dataset and both embedded and semidetached modes. The study aimed to minimize MMRE, and the findings indicated that GWO outperformed the other algorithms.

Kumari and Pushkar [15] proposed a novel model for adjusting the basic COCOMO model parameters based on the metaheuristic Cuckoo Search algorithm. This model was tested on the NASA18 dataset, utilizing MMRE as the fitness function. Sachan et al. [16] investigated the effects of three differential evolution mutation strategies (DE/rand/1/bin, DE/rand/2/bin, and DE/best/1/bin) to enhance the coefficients of the COCOMO and COCOMO II models. Their experiments, conducted using the NASA93 and NASA63 datasets, indicated that the proposed models yielded more accurate effort estimations compared to the original models.

The focus on differential evolution extends to additional studies. Singh et al. [17] introduced a novel variant of differential evolution named Homeostasis adaptation-based mutation operator differential evolution (HABDE), incorporating the Homeostasis adaptation-based mutation operator (HABMO) to increase solution diversity and minimize the likelihood of converging on local optima. Compared to GA, PSO, and DE, HABDE demonstrated superior performance. Singh and Kumar [18] utilized Homeostasis mutation-based differential evolution (HMBDE), which augments the mutation process in the original model through a Homeostasis mutation vector. In these studies, optimization targeted the basic COCOMO model, employing MMRE as the metric and focusing on the NASA63 dataset, segmented by development modes.

III. METHODOLOGY

A. METHODOLOGY OVERVIEW

Our methodology encompasses several key components: the adaptation and optimization of COCOMO models using SOMA, dataset preparation and preprocessing, evaluation metrics, and statistical analysis to validate the findings.

1) COST CONSTRUCTIVE MODELS

We base our study on the traditional COCOMO and COCOMO II models, which provide a structured framework for software effort estimation through algorithmic models that utilize project size, development environment, and personnel capabilities among other factors. Our focus extends to the basic, intermediate, and post-architecture models of COCOMO and COCOMO II, which are adapted and optimized through metaheuristic algorithms.

2) SELF-ORGANIZING MIGRATING ALGORITHM

At the core of our methodology is the SOMA, a metaheuristic optimization algorithm chosen for its efficiency in finding near-optimal solutions in complex, multidimensional spaces. SOMA's evolutionary approach to solution improvement is leveraged to optimize the parameters of the COCOMO models, thus enhancing their predictive accuracy.

3) DATASETS DESCRIPTION

Our analysis utilizes six publicly available datasets (NASA93, NASA63, NASA18, Kemerer, Miyazaki94, and Turkish) characterized by diverse project types and sizes. These datasets undergo rigorous preprocessing to ensure consistency and accuracy in the estimation process.

4) EVALUATION METRICS

To assess the performance of the SOMA-optimized COCOMO models, we employ several evaluation metrics including Mean Magnitude of Relative Error (MMRE), Prediction at level 0.25 (PRED(0.25)), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE). These metrics allow for a comprehensive analysis of estimation accuracy and model performance.

5) STATISTICAL ANALYSIS

A paired Wilcoxon signed-rank test is used to evaluate the statistical significance of the results, comparing the performance of the SOMA-optimized models against traditional COCOMO models. This non-parametric test provides a robust mechanism for validating the enhancements achieved through our proposed methodology.

6) EXPERIMENTAL SETUP

We detail the experimental setup for testing the efficacy of the SOMA-optimized COCOMO models, including the configuration of the SOMA parameters, the cross-validation process, and the comparative analysis against baseline models and other metaheuristic algorithms.

B. COST CONSTRUCTIVE MODELS

The COnstructive COst MOdel (COCOMO) was developed using 63 different sample projects and was published in 1981 by Barry W. Boehm. According to [3], Boehm suggested three types of COCOMO modes: basic, intermediate, and detailed. Various input parameters estimate the cost depending on the model type. These parameters include project size measured in thousands of lines of source code ($KLOC$), 15 cost driver attributes, and calibration constants (a, b, c, d), whose values depend on the project's mode (organic, semi-detached, or embedded) and are listed in Table 1. Organic mode requires extensive experience with similar projects, a thorough understanding of the project

TABLE 1. Software development modes [3].

Development mode	a		b	c	d
	Basic	Intermediate			
organic (< 50 KLOC)	2.4	3.2	1.05	2.5	0.38
semi-detached (< 300 KLOC)	3	3.0	1.12	2.5	0.35
embedded (all KLOC)	3.6	2.8	1.20	2.5	0.32

goals, and is open to changes in requirements and other technical specifications [3]. In contrast to organic mode, embedded mode requires moderate experience, a general understanding of the project objectives, and strict compliance with the requirements and other specifications. Semi-detached mode is in the middle of the previous modes.

The basic COCOMO model, using (1), calculates effort expressed in person-months (PM) by relying on the size of the project and calibration constants (a, b) [3]. The exponent (b) reflects that the effort needed to develop the software does not increase linearly with the project size. Time required for software development (T_{DEV}) in calendar months can be obtained using (2), where c and d are calibration constants.

$$EFFORT[PM] = a \cdot (KLOC)^b \tag{1}$$

$$T_{DEV}[MONTHS] = c \cdot (EFFORT)^d \tag{2}$$

The intermediate COCOMO model computes effort (3) more precisely than the basic COCOMO model since it considers a set of effort multipliers [3]. The product of effort multipliers (EM_i) consists of 15 cost driver attributes divided into four categories (Table 2): product attributes, computer attributes, personnel attributes, and project attributes. Each cost driver is assigned one of six rating levels (Table 3), and each rating level has a single value assigned to it, known as the effort multiplier (EM).

$$EFFORT[PM] = a \cdot (KLOC)^b \cdot \prod_{i=1}^{15} EM_i \tag{3}$$

The detailed COCOMO model updates the values of cost factors at each stage of the development life cycle [3].

C. COST CONSTRUCTIVE MODEL II

The COCOMO II model made some advancements, such as non-sequential development, reuse, and object-oriented approach [4]. This model defines three submodels: the Applications Composition model, the Early Design model, and the Post-Architecture model.

The Application Composition model estimates effort in projects created by assembling existing components and estimates are based on object points [20].

The Early Design model is used to determine a preliminary estimate in the initial phases of development when a detailed system design is unavailable [20]. This model employs a set of seven effort multipliers, five scale factors, and the software size characterized by unadjusted function points or thousands of lines of code. Function points can be converted into lines of source code (LOC) using conversion tables.

TABLE 2. Cost drivers: COCOMO (intermediate), COCOMO II (post-architecture) [3], [19].

Category	Cost Driver	Description	COCOMO	COCOMO II
Product Attributes	RELY	Required software reliability	Y	Y
	DATA	Database size	Y	Y
	CPLX	Product complexity	Y	Y
	RUSE	Developed for reusability	-	Y
	DOCU	Documentation match to life cycle needs	-	Y
Computer Attributes	TIME	Execution time constraint	Y	Y
	STOR	Main storage constraint	Y	Y
	PVOL	Platform volatility	-	Y
	VIRT	Virtual machine volatility	Y	-
	TURN	Computer turnaround time	Y	-
Personnel Attributes	ACAP	Analyst capability	Y	Y
	PCAP	Programmer capability	Y	Y
	PCON	Personnel continuity	-	Y
	APEX	Application experience	-	Y
	PLEX	Platform experience	-	Y
	LTEX	Language and tool experience	-	Y
	LEXP	Programming language experience	Y	-
	VEXP	Virtual machine experience	Y	-
	AEXP	Applications experience	Y	-
Project Attributes	TOOL	Use of software tools	Y	Y
	SITE	Multisite development	-	Y
	SCED	Required development schedule	Y	Y
	MODP	Use of modern programming practices	Y	-

TABLE 3. intermediate COCOMO: effort multipliers ratings [3].

Cost Driver	Very low	Low	Nominal	High	Very high	Extra high
ACAP	1.46	1.19	1.00	0.86	0.71	-
PCAP	1.42	1.17	1.00	0.86	0.70	-
AEXP	1.29	1.13	1.00	0.91	0.82	-
MODP	1.24	1.10	1.00	0.91	0.82	-
TOOL	1.24	1.10	1.00	0.91	0.83	-
VEXP	1.21	1.10	1.00	0.90	-	-
LEXP	1.14	1.07	1.00	0.95	-	-
SCED	1.23	1.08	1.00	1.04	1.10	-
STOR	-	-	1.00	1.06	1.21	1.56
DATA	-	0.94	1.00	1.08	1.16	-
TIME	-	-	1.00	1.11	1.30	1.66
TURN	-	0.87	1.00	1.07	1.15	-
VIRT	-	0.87	1.00	1.15	1.30	-
RELY	0.75	0.88	1.00	1.15	1.40	-
CPLX	0.70	0.85	1.00	1.15	1.30	1.65

The Post-Architecture model, based on a detailed system specification and suitable for use in the development or maintenance phase, provides more accurate estimations than the Early Design Model [19]. The estimated effort in person-months is calculated using (4), and exponent E is given by (5). The exponent E value ranges from 1.1 to 1.26 and depends on calibration constant b and the sum of five scaling factors SF_j (Table 4).

Each scale factor has 6 rating levels, and each level has assigned a weight (Table 5). In this model, 17 effort multipliers are considered (Table 2, Table 5), and the sizing parameter can be expressed in KLOC or the number of function points. Calibration constants $a = 2.94$ and

TABLE 4. Scale factors [19].

Scale Factor	Description
PREC	Precedentedness
FLEX	Development flexibility
RESL	Risk resolution
TEAM	Team cohesion
PMAT	Process maturity

TABLE 5. post-architecture COCOMO II: effort multipliers ratings, scale factors ratings [19].

Driver	Type	Very low	Low	Nominal	High	Very high	Extra high
PREC	SF	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	SF	5.07	4.05	3.04	2.03	1.01	0.00
RESL	SF	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	SF	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	SF	7.80	6.24	4.68	3.12	1.56	0.00
RELY	EM	0.82	0.92	1.00	1.10	1.26	-
DATA	EM	-	0.90	1.00	1.14	1.28	-
CPLX	EM	0.73	0.87	1.00	1.17	1.34	1.74
RUSE	EM	-	0.95	1.00	1.07	1.15	1.24
DOCU	EM	0.81	0.91	1.00	1.11	1.23	-
TIME	EM	-	-	1.00	1.11	1.29	1.63
STOR	EM	-	-	1.00	1.05	1.17	1.46
PVOL	EM	-	0.87	1.00	1.15	1.30	-
ACAP	EM	1.42	1.19	1.00	0.85	0.71	-
PCAP	EM	1.34	1.15	1.00	0.88	0.76	-
PCON	EM	1.29	1.12	1.00	0.90	0.81	-
APEX	EM	1.22	1.10	1.00	0.88	0.81	-
PLEX	EM	1.19	1.09	1.00	0.91	0.85	-
LTEX	EM	1.20	1.09	1.00	0.91	0.84	-
TOOL	EM	1.17	1.09	1.00	0.90	0.78	-
SITE	EM	1.22	1.09	1.00	0.93	0.86	0.80
SCED	EM	1.43	1.14	1.00	1.00	1.00	-

$b = 0.91$ if local data are unavailable. Time required for software development (T_{DEV}) in calendar months is calculated using (6) and (7), where c and d are calibration constants ($c = 3.67$, $d = 0.28$).

$$EFFORT[PM] = a \cdot (KLOC)^E \cdot \prod_{i=1}^{17} EM_i \tag{4}$$

$$E = b + 0.01 \cdot \sum_{j=1}^{5} SF_j \tag{5}$$

$$T_{DEV}[MONTHS] = c \cdot (EFFORT)^F \tag{6}$$

$$F = d + 0.2 \cdot 0.01 \cdot \sum_{j=1}^{5} SF_j$$
$$= d + 0.2 \cdot (E - b) \tag{7}$$

D. SELF-ORGANIZING MIGRATING ALGORITHM

The Self-Organizing Migrating Algorithm (SOMA) is a stochastic optimization algorithm classified as a swarm algorithm but can also be classified as an evolutionary algorithm [21], [22]. This algorithm is based on self-organizing behaviour, which occurs when a social group of individuals cooperate in solving a common problem (e.g., finding a food source). SOMA varies from standard evolutionary algorithms

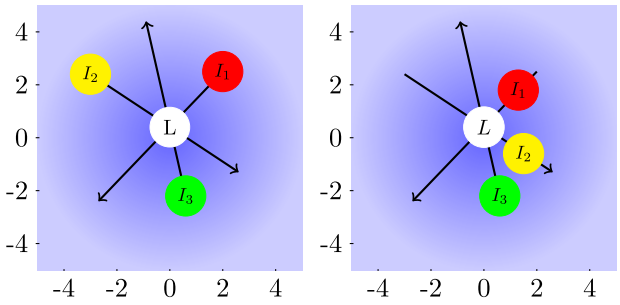

FIGURE 1. Strategy all to one: individuals before migration and after one migration (L-Leader, I_1 - Individual 1, I_2 - Individual 2, I_3 - Individual 3) [22].

TABLE 6. SOMA parameters [22].

Parameter	Recommended range
PathLength	[1.1, 5]
Step	[0.11, PathLength]
PRT	[0, 1]
Dim	given by problem
PopSize	[10, up to user]
Migrations	[10, up to user]
MinDiv	[arbitrary negative, up to user]

(genetic algorithm, differential evolution) in that no new individuals are formed during the optimization process, but the original individuals migrate in each generation. In SOMA, the name *generation* is replaced by a *migration loop*, and operations like mutation and crossover, which are already familiar from evolutionary algorithms, constitute a significant part of this algorithm. Moreover, the strategy choice and the definition of the parameters influence the optimization process, and in the algorithm, the initial population of candidate solutions is randomly generated based on $Specimen = \{\{Integer, \{Low, High\}\}, \{Real, \{Low, High\}\}, \dots\}$.

The strategies allow individuals to cooperate in searching for an optimal solution. In each migration loop, the Leader must be identified, and the choice of the Leader depends on the strategy (All To One, All To All, All To Rand, and All To All Adaptive) [22]. In All to One strategy (Figure 1), the individual with the best fitness will be chosen as a Leader, and all other individuals (except the Leader) will migrate toward the Leader.

The SOMA is controlled by controlling parameters (PathLength, Step, PRT, PopSize) and stopping parameters (Migrations, MinDiv) shown in Table 6 [22]. Parameter *PathLength* specifies the length of the individual's path. If the *PathLength* value is 1, the individual will reach the Leader's position. If the value of *PathLength* is less than 1, the position of the Leader will not be reached, and the individual may become stuck in the local optimum. The *Step* parameter expresses the length of the step. The recommended value of 0.11 makes it impossible for an individual to migrate to the Leader's position. The *PRT* parameter (perturbation) is a mutation parameter representing an individual's travel direction. The higher the *PRT* value, the faster the convergence. The *Dim* parameter specifies the

dimensionality of the solved problem. The *PopSize* parameter represents the population size. The *Migration* parameter sets the maximum number of iterations, and the *MinDiv* parameter expresses the maximum permissible difference between the best and worst individual.

Mutation ensures diversity between individuals by generating random perturbations. SOMA does this by utilizing the *PRT* parameter and generating a perturbation vector (*PRTvector*) through it [21]. A perturbation vector is generated in each migration loop for each individual before his migration. For each individual's parameter, a random number $rnd_j \in [0, 1]$ is generated, where $j = 1, 2, \ldots, Dim$, and depending on the value of the *PRT* parameter, the value 0 or 1 is stored in the perturbation vector for the given parameter j according (8).

$$\begin{aligned} &if \quad rnd_j < PRT \quad then \quad PRTvector_j = 1 \\ &\qquad\qquad\quad else \quad PRTvector_j = 0 \end{aligned} \tag{8}$$

The crossover operation ensures the creation of new individuals. New positions in N-dimensional space are generated for each individual as a sequence of possible solutions, and only the best fittest survives to the next migration loop. An individual's discrete movement is given by (9), where *ML* is the current migration loop, $x_{i,j}^{ML+1}$ is the new position of the current individual i, $x_{i,j}^{ML}$ is the current position of the actual individual i, $x_{L,j}^{ML}$ is the Leader position in current migration loop, and $t \in [0, by \quad Step \quad to, PathLenght]$ is a step generated from the position of the actual individual i to the Leader's position [22]. $PRTvector_j$ defines the movement direction. The value 0 for element j of individual i indicates that the parameter for the relevant dimension can not be changed.

$$x_{i,j}^{ML+1} = x_{i,j}^{ML} + (x_{L,j}^{ML} - x_{i,j}^{ML}) \cdot t \cdot PRTvector_j \tag{9}$$

In the original version of SOMA [21], the *PRTvector* was generated for each individual once; however, with changes in [22], the *PRTvector* is formed for each new t jump. This modification ensures more dynamic searching in the space; a stepwise line is used instead of a straight line.

E. DATASETS DESCRIPTION

A total of six publicly available datasets were selected for training and testing, namely NASA93, NASA63, and the Turkish dataset from the PROMISE data repository [23], Kemerer [24] and Miyazaki94 dataset [25] from the SEACRAFT repository [26], and dataset NASA18 [27]. NASA93, referred to as COCOMO NASA 2, and NASA63 dataset, known as COCOMO81, have a typical attribute structure for the intermediate COCOMO model (15 effort multipliers, project size and one attribute for actual effort). NASA93 contains 93 projects from different centers developed between 1971 and 1987, and in addition to the above, each project has seven additional attributes (e.g., development mode). NASA63 consists of 63 projects, and the division of the dataset into development modes is reported by Singh et al. in study [17]. NASA18, Kemerer, and Miyazaki94 datasets provide attributes only for the basic COCOMO model (project size and actual effort). NASA18 contains 18 projects collected from the SW Engineering Laboratory at the NASA/Goddard Space Flight Center, and ten attributes describe each project. Kemerer dataset has 15 projects with eight attributes, and the project size is measured in KLOC and function points. Miyazaki94 dataset consists of 48 software projects described by nine attributes. Turkish dataset (COCOMO SDR) contains 12 projects from 5 different software companies in various domains. Each project has 25 attributes in the standard format of the post-architecture COCOMO II model, including actual effort and project ID.

This study uses project size, effort multipliers, and scale factors as independent variables to calculate the predicted effort values of the dependent variable (effort). To assess the performance of models, the attribute with the actual effort values of the dependent variable will used. Attribute for development mode will also considered in this study. Each dataset expresses effort in PM and project size in KLOC, except for the Turkish dataset, which states project size in LOC. This value will be converted to KLOC.

Actual effort and project size characteristics are processed by descriptive statistics (minimum and maximum value, mean value, median, standard deviation, and range) in Table 7. The table also includes the number of projects (n) in the dataset. The characteristics show significant dissimilarities, especially for the NASA93 and NASA63 datasets. Therefore, the relationship between project size and actual effort is investigated: how many thousands of lines of code can be implemented with one effort. The distribution of ratios into quartiles is shown in Figures 2 and 3, where graphical data visualization using a box plot and inter-quartile range (IQR) method helps easily identify unusual values. In the IQR method, we applied a rule based on the whisker value equal to 3.0 to identify only extreme values marked as far out outliers [28], [29]. These values are located outside the outer fences, i.e., out of range $[Q_1 - 3 \cdot IQR, Q_3 + 3 \cdot IQR]$.

F. EVALUATION METRICS

Evaluation criteria are an essential part of evaluating the accuracy of prediction models; they measure how closely the predicted effort P_i of project i from a dataset of size n matches its actual effort A_i. In this study, the accuracy of the proposed model is evaluated using several metrics [30], [31], [32]: MMRE, PRED(0.25), MAE, and RMSE. For all these metrics, with the exception of PRED(0.25), a lower value obtained from a specific metric indicates a higher accuracy of prediction. In the software effort estimation area, the MMRE metric is the widely adopted [2], [33], [34], [35], [36] to validate models, and therefore we also used it as a fitness function in this study. M. Shepperd and S. MacDonell [32] warn that the MMRE is biased to underestimated projects.

TABLE 7. Descriptive statistics.

Dataset		n	KLOC						Effort					
			min	max	mean	median	st deviation	range	min	max	mean	median	st deviation	range
NASA93	organic	3	40	240	123.33	90	84.98	200	150	192	168	162	17.66	42
	semi-detached	69	0.9	980	88.86	32.6	141.52	979.1	8.4	4560	421.52	155	723.55	4551.6
	embedded	21	3	350	106.8	65	104.81	347	12	8211	1356.24	648	1797.81	8199
NASA63	organic	25	3	132	26.88	16	28.62	129	6	243	76.52	57	65.74	237
	semi-detached	11	2.14	1150	193.94	28	333.22	1147.86	7.3	6600	1035.3	98	1843.66	6592.7
	embedded	27	1.98	390	76.25	30	106.35	388.02	5.9	11400	1101.77	321	2379.95	11394.1
NASA18		18	2.1	100.8	35.26	17.15	34.11	98.7	5	138.3	49.47	26.2	44.44	133.3
Kemerer		15	39	450	186.57	164.8	132.18	411	23.2	1107.31	219.25	130.3	254.14	1084.11
Miyazaki94		48	6.9	417.6	70.79	45.4	86.65	410.7	5.6	1586	87.47	38.1	226.36	1580.4
Turkish		12	1.37	114.28	20.92	7.12	30.79	112.91	1	22	5.73	3.5	6.55	21

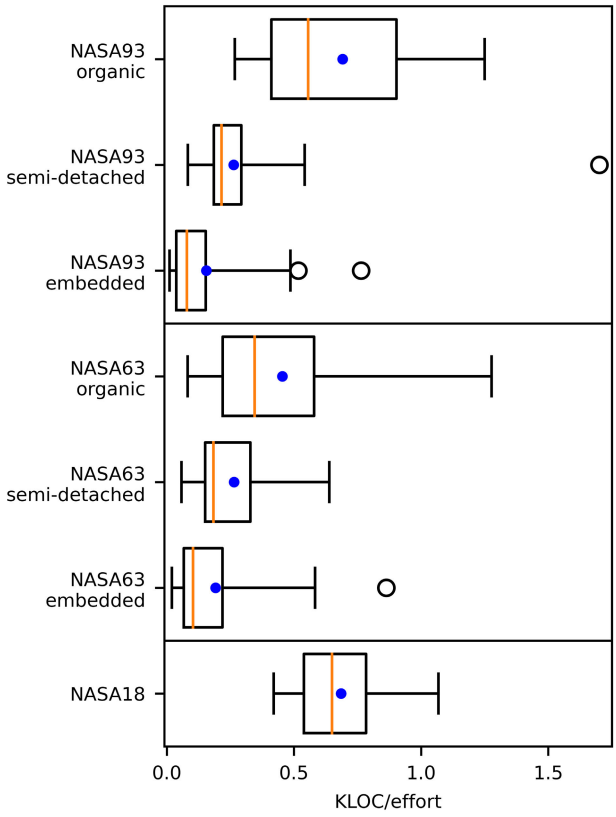

FIGURE 2. NASA datasets: the distribution of ratios of project size and actual effort (blue dot = mean).

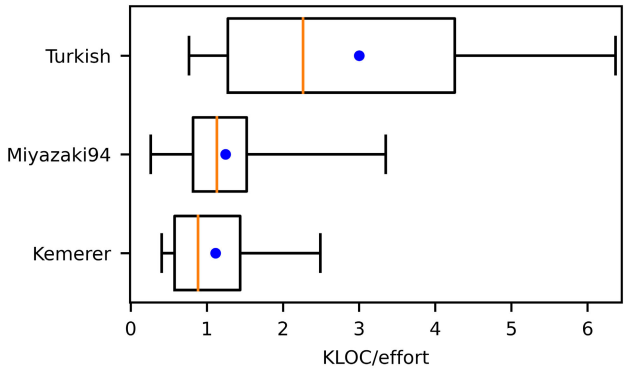

FIGURE 3. Turkish, Miyazaki94 and Kemerer dataset: the distribution of ratios of project size and actual effort (blue dot = mean).

Magnitude of Relative Error (MRE) (10) is the ratio of the absolute error and the actual effort value, where the absolute error is the difference between the actual effort value and the predicted effort value.

$$MRE = \frac{|A_i - P_i|}{A_i} \tag{10}$$

Mean Magnitude of Relative Error (MMRE) (11) is defined as the mean of the MRE values over n projects in the dataset.

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE_i \tag{11}$$

PRED(x) (12), represents prediction accuracy at level x. Where x is defined as the ratio of k projects to all projects in the dataset. The value of k indicates the number of projects in which the MRE value is less than or equal to the value of x.

$$PRED(x) = \frac{k}{n} \tag{12}$$

Mean Absolute Error (MAE) (13) measures the mean of absolute errors.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |A_i - P_i| \tag{13}$$

Root Mean Squared Error (RMSE) (14) measures how data are concentrated around the best fit.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(A_i - P_i)^2} \tag{14}$$

G. STATISTICAL ANALYSIS

This study employs a paired Wilcoxon signed-rank test to evaluate statistical significance in effort prediction between eSOMCOCOMO and the original COCOMO and COCOMO II models. This non-parametric test is based on the ranks derived from the differences between two related paired samples [37]. At a significance level of 0.05, the statistical hypotheses in subsection I-C are tested. We reject the null hypothesis (H0) and accept the alternative hypothesis (H1)

if the obtained significance value (p-value) is less than or equal to 0.05. We do not reject the null hypothesis (H0) if the p-value exceeds 0.05.

IV. EXPERIMENTAL SETUP

This study investigates the impact of the optimization algorithm SOMA on the precision of estimates of the COCOMO and COCOMO II models, where SOMA is used to optimize calibration constants a and b. Subsection IV-A introduces the proposed approach, the enhanced Self-Organizing Migrating Constructive Cost Model (eSOMCOCOMO). Subsection, IV-B describes test experiments carried out on datasets NASA93, NASA63, NASA18, Kemerer, Miyazaki94, and Turkish. The last subsection, IV-C, contains chosen baseline models and metaheuristic algorithms for performance evaluation. MMRE, PRED(0.25), MAE, and RMSE were used to verify the results.

A. PROPOSED APPROACH eSOMCOCOMO

The enhanced Self-Organizing Migrating Constructive Cost Model (eSOMCOCOMO) flowchart shows Figure 4 and comprises two phases. The first phase is data preprocessing, and the second phase constitutes the foundation of eSOMCOCOMO, uniting constructive cost models with the SOMA optimization algorithm.

The first phase starts by loading the dataset and its preprocessing. In the case of the NASA93, NASA63, and Turkish datasets, the preprocessing steps were supplemented by converting the ordinal features into numerical values according to Tables 3 or 5, and projects in datasets NASA93 and NASA63 were also divided into software development modes, where each mode is treated separately. Subsequently, the correctness of the data was checked. There are no missing values in datasets, and examining the ratio between the project sizes and actual effort values showed minor discrepancies in some projects (Figures 2 and 3). Therefore, these projects were considered outliers and removed.

In the second phase, the optimization process is carried out on test experiments (Table 9). A 10-fold cross-validation technique is used for each experiment to mitigate the problem of overfitting. In 10-fold cross-validation, the dataset is randomly and equally partitioned into ten folds. In each iteration, one-fold is used for testing, and the remaining 9-folds are used for training. The SOMA optimization algorithm (Figure 5) is applied to the train data, and the best solution found is evaluated on the test data. The optimization's overall accuracy/solution is given by averaging the results of iterations (accuracies from testing/the best solutions found during training).

1) SOMA OPTIMIZATION ALGORITHM

The parameters are set at the beginning of the SOMA algorithm (Table 8). The values for control parameters were picked from recommended ranges (Table 6), and the number of migrations was chosen as the stopping parameter. The lower range limit was chosen when setting the *Dim* parameter since the suggested rule [22], $[0.5, 0.7] \cdot DIM$, is not reasonable to apply in a 2-dimensional problem. The value of the *PRT* parameter for a low-dimensional problem with a large population can be set from the higher range $[0.7, 1]$ despite the recommended value being 0.1 [22]. A larger value of *PRT* eliminates the occurrence of the stochastic component and increases the convergence rate, so we decided to set $PRT = 0.6$. The explore domain of optimized constants a and b are subject to lower and upper boundary constraints, where $a \in [0, 10]$ and $b \in [0.2, 3]$.

TABLE 8. SOMA parameter settings.

Parameter	Value
PathLength	3
Step	0.11
PRT	0.6
Dim	2
PopSize	10
Migrations	50
Strategy	All To One
Domain a [Lo, Hi]	[0, 10]
Domain b [Lo, Hi]	[0.2, 3]

The initial population is created randomly based on the specimen definition, $Specimen = \{\{Real, \{0, 10\}\}, \{Real, \{0.2, 3\}\}\}$. Each individual in the population represents one candidate solution in a 2-dimensional space. The objective of the optimization is to minimize or maximize the value of the objective function by optimizing the parameters of individuals. In this study, the MMRE was chosen as a fitness function. Therefore, the individual that reaches the lowest MMRE value after the maximum number of migrations will represent a new solution, i.e., optimized constants a and b.

Before the first migration begins, it is necessary to know the fitness of the individuals from the initial population. An individual's fitness is given by MMRE (11), and for each project in the dataset, MRE is calculated by (10). The actual effort value is known from the dataset, and the estimated effort is for the basic COCOMO model obtained by (1), for the intermediate COCOMO model given by (3), and for the post-architecture COCOMO II model calculated by (4) and (5). Scale factors, effort multipliers, and values expressing project size are part of dataset. In each migration loop, the Leader must be identified. When applying the All To One strategy, the individual with the best fitness becomes the Leader, and the remaining individuals migrate toward the Leader. Individuals migrate one after the other. The *PRT* parameter influences the direction of migration, and according to (8), a new *PRTvector* is constructed for each t jump of the individual, and new positions are then given by (9). After the jump positions of the individual are known, the domains are checked. Each position out of bounds will be replaced by a new randomly generated one. In the next step, the fitness of the jump positions is calculated. The individual remembers the best position found during his migration and returns to its starting position. Finally, once all migrations of individuals are completed, each individual moves to his

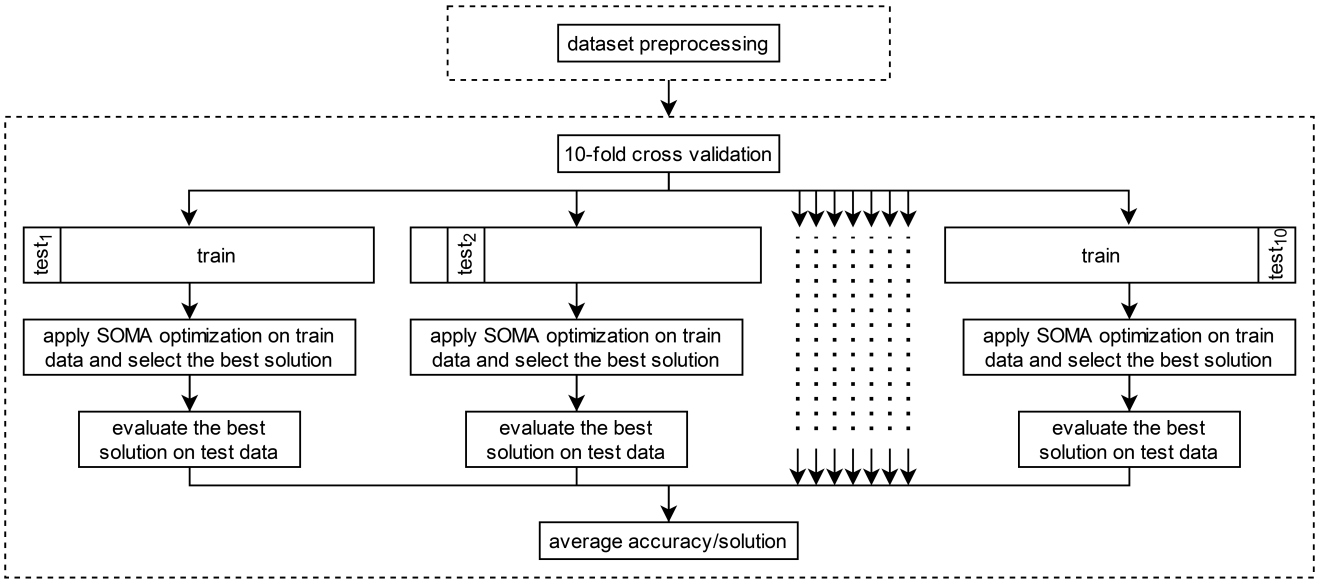

FIGURE 4. Flowchart of the proposed approach eSOMCOCOMO.

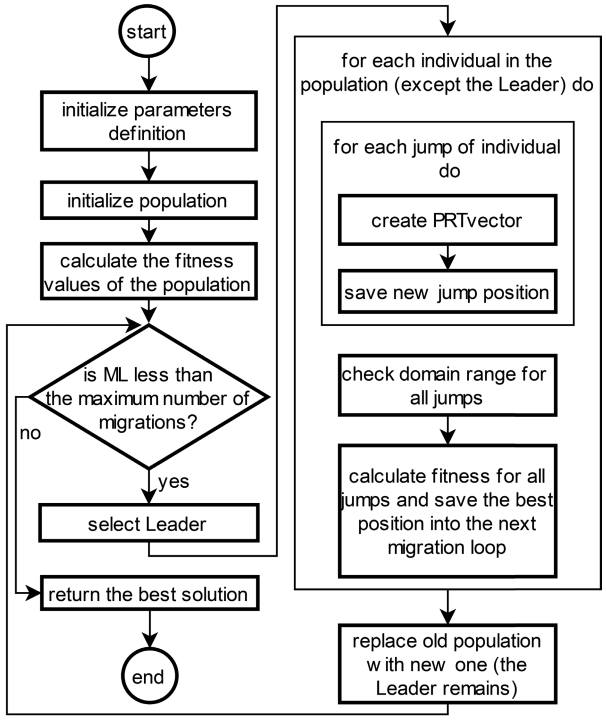

FIGURE 5. SOMA optimization algorithm [21], [22].

best-found position. The algorithm terminates by reaching the maximum number of migrations and returning the best solution.

B. EXPERIMENTS DESCRIPTION

The eSOMCOCOMO was tested in eight experiments (Table 9). The optimization of the basic COCOMO model

TABLE 9. Test experiments.

Model	Experiment	Dataset	Note
basic COCOMO	EXP A1	NASA18	-
	EXP A2	Kemerer	-
	EXP A3	Miyazaki94	-
	EXP A4	NASA93	datasets divided into software development modes
	EXP A5	NASA63	
intermediate COCOMO	EXP B1	NASA93	
	EXP B2	NASA63	
post-architecture COCOMO II	EXP C1	Turkish	-

was performed on datasets NASA18, Kemerer, Miyazaki94, NASA93, and NASA63. The intermediate COCOMO model was tested on NASA datasets NASA93 and NASA63, and the last optimization of the post-architecture COCOMO II model was performed on the Turkish dataset. Software developmental modes were also considered for the NASA93 and NASA63 datasets, and the numbers of projects in each mode are shown in Table 7. NASA93 dataset has only three projects in organic mode; therefore, this optimization was not performed.

C. BENCHMARK MODELS

The performance of the eSOMCOCOMO against traditional models and other metaheuristic algorithms was assessed by MMRE, PRED(0.25), MAE, and RMSE. The proposed experiments (Table 10) were compared with the original COCOMO or COCOMO II model and metaheuristic algorithms (Particle Swarm Optimization (PSO) [9] and Genetic Algorithm (GA) [38]). Moreover, the experiment (EXP C1) was compared with Local Calibration [39], and the experiments (EXP A1 - EXP A5) were compared with the baseline models (the Walston-Felix model [40], the

TABLE 10. Baseline models.

Walston-Felix model	$EFFORT[PM] = 0.7 \cdot (KLOC)^{0.91}$
Bailey-Basili model	$EFFORT[PM] = 3.5 + 0.73 \cdot (KLOC)^{1.16}$
Halstead model	$EFFORT[PM] = 5.2 \cdot (KLOC)^{1.50}$

TABLE 11. EXP A1: comparison of MMRE, PRED(0.25), MAE and RMSE.

Model		MMRE	PRED(0.25)	MAE	RMSE
basic COCOMO	organic	0.92162	0.16667	54.0705	82.4695
	semi-detached	2.0023	0.0	122.633	184.43
	embedded	3.68606	0.0	237.434	360.1
Walston-Felix		0.63396	0.0	32.2469	43.8222
Bailey-Basili		0.20098	0.72222	10.8643	17.4023
Halstead		18.4494	0.0	1406.28	2255.41
PSO		0.19769	0.83333	8.03563	12.4692
GA		0.19997	0.72222	8.78451	13.5174
eSOMCOCOMO		0.19692	0.83333	7.91715	12.3742

Bailey-Basili model [27], and the Halstead model [41]). The definition of the baseline models is presented in Table 10.

V. RESULTS AND DISCUSSION

The efficiency of the proposed eSOMCOCOMO approach with other models (IV-C) was compared on preprocessed datasets and evaluated by MMRE, PRED(0.25), MAE, and RMSE. The results are presented according to the type of optimized model in the following subsections. In the case of eSOMCOCOMO, we used the solution found during the optimization process (Figure 4) instead of the original constants (a and b) to predict effort. Moreover, statistical testing, Taylor diagrams, and residual analysis were employed to assess the validity of the eSOMCOCOMO.

A. BASIC COCOMO (EXP A1-A5)

EXP A1 was conducted on the NASA18 dataset, and the results presented in Table 11 show that eSOMCOCOMO achieved the best results across all evaluation metrics. Compared to the original COCOMO models, there was a significant reduction in error, and the prediction at level 0.25 increased enormously from 0.16667 to 0.83333. Particle Swarm Optimization is the second-best prediction model, with nearly comparable performance, and the Genetic Algorithm is the third-best. Also worth mentioning is the Bailey-Basili model, which predicts almost as well as metaheuristic algorithms. On the other hand, the Halstead model achieves the highest error rate.

Table 12 shows the outcomes for EXP A2. If MMRE is considered, GA and PSO algorithms perform better than eSOMCOCOMO. The approach eSOMCOCOMO seemed stuck in a local optimum while searching for an optimal solution. However, eSOMCOCOMO outperforms GA and PSO regarding PRED(0.25), MAE, and RMSE. The Halstead and the basic COCOMO models are the least strong prediction models.

According to Table 13, in experiment EXP A3, the most finest results were achieved by the eSOMCOCOMO model, except for the RMSE metric. The lowest RMSE value (152.093) reached the Bailey-Basili model. The Walston-Felix model also achieves promising results, but only 16.667% of the samples had a value of MRE less than or equal to 25%.

TABLE 12. EXP A2: comparison of MMRE, PRED(0.25), MAE and RMSE.

Model		MMRE	PRED(0.25)	MAE	RMSE
basic COCOMO	organic	2.44324	0.0	370.004	479.3
	semi-detached	5.15422	0.0	864.256	1098.54
	embedded	10.1513	0.0	1806.12	2338.82
Walston-Felix		0.51941	0.2	140.862	260.308
Bailey-Basili		1.00806	0.33333	154.408	219.492
Halstead		76.8	0.0	15481	21700.3
PSO		0.45803	0.33333	104.5	220.272
GA		0.45075	0.2	113.662	235.975
eSOMCOCOMO		0.47567	0.53333	98.4948	212.325

TABLE 13. EXP A3: comparison of MMRE, PRED(0.25), MAE and RMSE.

Model		MMRE	PRED(0.25)	MAE	RMSE
basic COCOMO	organic	2.63363	0.04167	137.808	229.063
	semi-detached	4.9721	0.0	290.527	491.016
	embedded	8.86714	0.0	583.856	1055.98
Walston-Felix		0.45786	0.16667	56.637	209.682
Bailey-Basili		0.90107	0.29167	63.3831	152.093
Halstead		48.6347	0.0	4325.43	9850.29
PSO		0.34737	0.47917	47.331	200.319
GA		0.35651	0.4375	48.4841	203.399
eSOMCOCOMO		0.34647	0.47917	47.2857	199.628

TABLE 14. EXP A4: comparison of MMRE, PRED(0.25), MAE and RMSE.

Mode	Model	MMRE	PRED(0.25)	MAE	RMSE
semi-detached	basic COCOMO	0.38252	0.48529	158.017	356.33
	Walston-Felix	0.87705	0	387.777	778.879
	Bailey-Basili	0.62511	0.02941	272.277	533.921
	Halstead	8.60861	0	6814.97	21020.1
	PSO	0.32029	0.5	136.775	292.136
	GA	0.32026	0.5	135.113	286.602
	eSOMCOCOMO	0.32029	0.5	136.665	291.522
embedded	basic COCOMO	0.58827	0.16667	945.937	1618.15
	Walston-Felix	0.96031	0	1477.81	2383.6
	Bailey-Basili	0.86925	0	1377.17	2268.37
	Halstead	3.23627	0.05556	3852.57	6473.98
	PSO	0.50785	0.33333	979.131	1840.67
	GA	0.52325	0.27778	990.585	1851.58
	eSOMCOCOMO	0.50775	0.33333	974.028	1830.58

EXP A4 utilizes the NASA93 dataset divided into software development modes (Table 14). In semi-detached mode, the best prediction characteristics were attained by GA. In embedded mode, eSOMCOCOMO has the lowest MMRE value; PSO and eSOMCOCOMO succeeded equally well in PRED(0.25), and the original basic COCOMO model leads in MAE and RMSE metrics. Upon closer inspection of the results, it is evident that predictions made with metaheuristic algorithms are nearly identical in semi-detached mode.

In EXP A5 (Table 15), the development modes of the NASA63 dataset are also considered. Regarding the MMRE measure, the Bailey-Basili model acts best in organic mode, GA in semi-detached mode, and PSO in embedded mode. The basic COCOMO model has the most outstanding PRED(0.25) outcomes in all development modes. It also

TABLE 15. EXP A5: comparison of MMRE, PRED(0.25), MAE and RMSE.

Mode	Model	MMRE	PRED(0.25)	MAE	RMSE
organic	basic COCOMO	0.69121	0.28	41.74	61.754
	Walston-Felix	0.7557	0	63.0279	85.0967
	Bailey-Basili	0.53771	0.2	42.8769	59.7146
	Halstead	10.928	0.04	908.625	1836.18
	PSO	0.54431	0.16	40.0943	56.0797
	GA	0.54572	0.12	44.4605	61.1576
	eSOMCOCOMO	0.54406	0.16	39.9012	55.9264
semi-detached	basic COCOMO	0.63229	0.36364	394.75	698.768
	Walston-Felix	0.86534	0	957.984	1972.6
	Bailey-Basili	0.56488	0.18182	638.206	1294.55
	Halstead	11.6183	0	25225.3	61550.4
	PSO	0.5246	0.09091	369.554	608.74
	GA	0.51153	0.18182	500.86	1063.9
	eSOMCOCOMO	0.52748	0.09091	334.724	515.088
embedded	basic COCOMO	0.83399	0.15385	782.737	1840.88
	Walston-Felix	0.91363	0	1108.75	2630.39
	Bailey-Basili	0.72627	0.11539	1011.9	2495.3
	Halstead	5.67378	0.03846	4693.69	10256.4
	PSO	0.63342	0.11539	946.961	2375.76
	GA	0.63349	0.07692	951.736	2388.61
	eSOMCOCOMO	0.63398	0.11539	942.272	2365.22

TABLE 16. EXP B1: comparison of MMRE, PRED(0.25), MAE and RMSE.

Mode	Model	MMRE	PRED(0.25)	MAE	RMSE
semi-detached	intermediate COCOMO	0.37249	0.52941	417.379	2462.24
	PSO	0.30636	0.52941	252.159	1046.88
	GA	0.31265	0.52941	255.164	1036.63
	eSOMCOCOMO	0.30629	0.54412	249.708	1018.47
embedded	intermediate COCOMO	0.29106	0.44444	429.148	765.741
	PSO	0.28796	0.44444	412.186	742.936
	GA	0.29639	0.38889	422.147	784.424
	eSOMCOCOMO	0.28743	0.44444	407.701	731.269

performed well concerning MAE and RMSE in embedded mode. The approach eSOMCOCOMO obtains the finest MAE and RMS values in organic and semi-detached modes.

B. INTERMEDIATE COCOMO (EXP B1-B2)

The results of experiments EXP B1 (NASA93 dataset) and EXP B2 (NASA63 dataset) were processed for each software development mode separately.

In EXP B1 (Table 16), eSOMCOCOMO produces the best outcomes in semi-detached and embedded modes. The PSO algorithm achieves nearly equally satisfactory results. In addition, in embedded mode, it can be seen that there is only a negligible improvement in prediction compared to the intermediate COCOMO model.

Following Table 17, the intermediate COCOMO model has the most outstanding outcomes for all development modes regarding MAE and RMSE. Moreover, it also achieves the highest value in PRED(0.25), except in semi-detached mode. In this mode, the highest value, 0.72727, was achieved by eSOMCOOCMO and metaheuristic algorithm PSO. Further, PSO has the lowest MMRE value in organic mode, whereas eSOMCOCOMO performs better in other modes.

C. POST-ARCHITECTURE COCOMO II (EXP C1)

The EXP C1 results are given in Table 18. The approach eSOMCOCOMO, adopting the SOMA optimization

TABLE 17. EXP B2: comparison of MMRE, PRED(0.25), MAE and RMSE.

Mode	Model	MMRE	PRED(0.25)	MAE	RMSE
organic	intermediate COCOMO	0.34687	0.52	22.9251	33.0368
	PSO	0.29277	0.44	24.4255	41.9924
	GA	0.29306	0.44	24.649	41.788
	eSOMCOCOMO	0.29324	0.48	24.2726	41.1277
semi-detached	intermediate COCOMO	0.23435	0.63636	201.42	367.354
	PSO	0.20804	0.72727	235.793	419.833
	GA	0.23002	0.45455	366.914	811.738
	eSOMCOCOMO	0.20764	0.72727	215.564	378.122
embedded	intermediate COCOMO	0.37411	0.53846	317.523	662.501
	PSO	0.36189	0.46154	336.459	750.862
	GA	0.36593	0.38462	354.868	871.7
	eSOMCOCOMO	0.36155	0.46154	335.502	739.839

TABLE 18. EXP C1: comparison of MMRE, PRED(0.25), MAE, and RMSE.

Model	MMRE	PRED(0.25)	MAE	RMSE
post-architecture COCOMO II	7.36273	0.25	48.9213	94.2338
Local Calibration	0.67334	0.25	4.38678	7.22315
PSO	0.49702	0.33333	3.72467	7.24501
GA	0.50969	0.25	3.75285	7.34343
eSOMCOCOMO	0.49694	0.33333	3.72461	7.23761

algorithm, improved the estimation accuracy of the COCOMO II model and outperformed benchmarked models in all evaluation metrics. Compared to the original COCOMO II model, there was a significant improvement in MMRE, MAE, and RMSE; e.g., the value of MMRE decreased from 7.36273 to 0.49694.

D. TAYLOR DIAGRAMS

The predictive skills of the eSOMCOCOMO against the original COCOMO models were also assessed using the Taylor diagram [42], [43]. Taylor's diagram demonstrates the similarity between selected benchmark models using three metrics: the Standard Deviation, the Correlation Coefficient, and the centered Root Mean Square Error (CRMSE).

Taylor diagrams for experiments EXP A1-A5 are shown in Figure 6. In EXP A1 (Figure 6a), the Correlation Coefficient is about 0.97 for all models, but its values slowly decrease in the following order: eSOMCOCOMO, M2, M3, and M4. The standard Deviation of eSOMCOCOMO is about 3, for M2 is about 5.1, for M3 is about 6.9, and for M4 is about 9.3. CRMSE is about 0.7 for eSOMCOCOMO, 1.9 for M2, 3.8 for M3, and 6.15 for M4. From the diagram, it can be concluded that in the case of EXP A1, eSOMCOCOMO achieves the best results because it lies closest to the "observed point or its time series line" and reaches a high correlation and the lowest value of centered Root Mean Square Error. M4 can be described as the worst predicting model. This model is farthest from the "observed" and has the highest value of CRMSE. Figures 6e and 6h show that the original model intermediate COCOMO demonstrates better results than the proposed approach eSOMCOCOMO. However, in the remaining diagrams, eSOMCOCOMO indicates superiority.

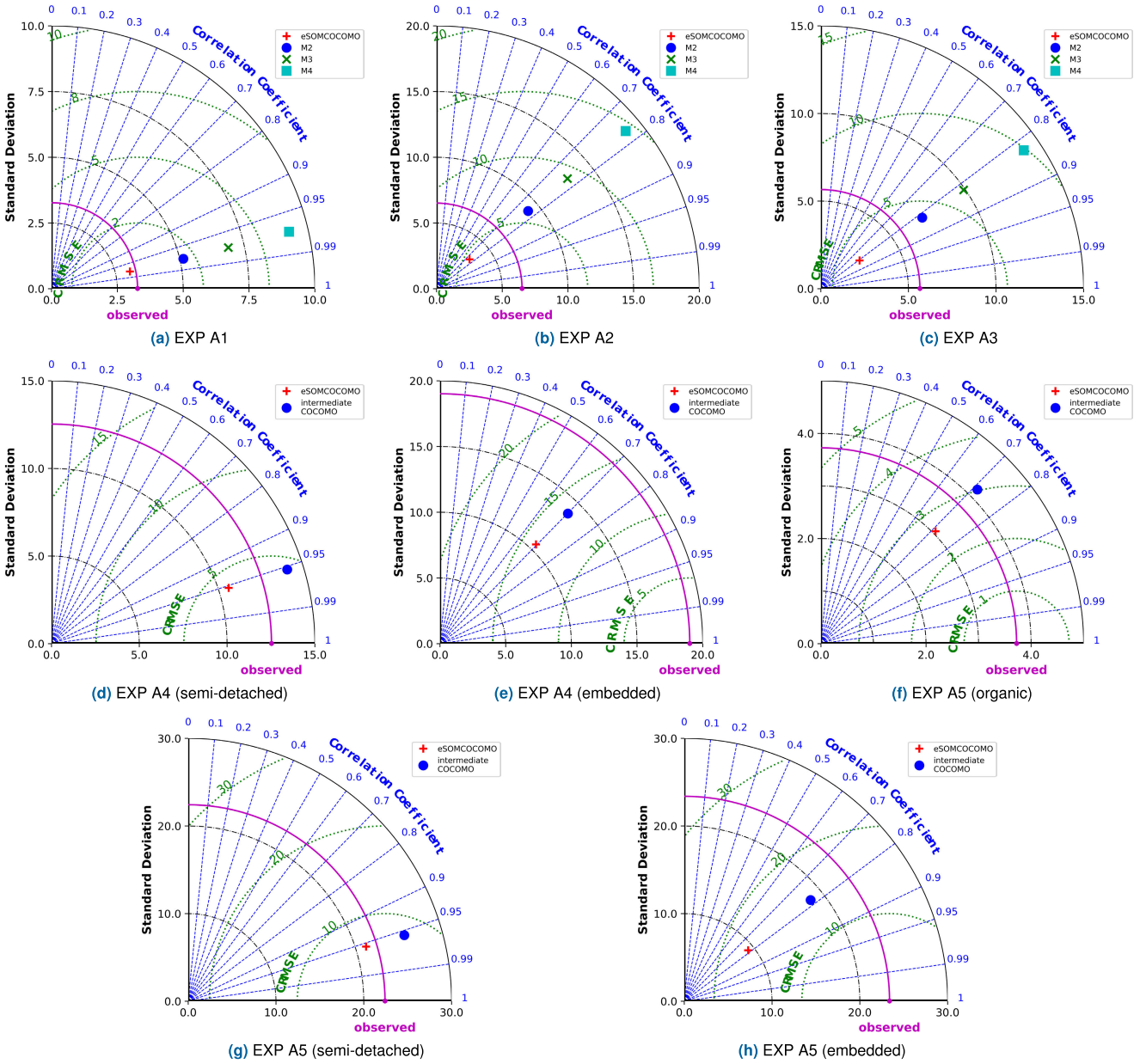

FIGURE 6. Taylor diagrams for EXP A1-A5 (M2-basic COCOMO (organic), M3-basic COCOMO (semi-detached), M4-basic COCOMO (embedded)) under non-standardized square root transformation.

Taylor diagrams for experiments EXP B1-B2 and EXP C1 are shown in Figure 7. The eSOMCOCOMO approach shows the best results in EXP B1 (semi-detached) (Figure 7a). In EXP C1 (Figure 7f), the eSOMCOCOMO has significantly lower CRMSE and is closer to the ''observed point'' than the original post-architecture model. However, its Correlation Coefficient is lower. Almost identical results can be seen in experiments in Figures 7b and 7d. In Figures 7c and 7e, the intermediate COCOMO model has a Standard Deviation similar to the ''observed point,'' CRMSE is insignificantly lower than in eSOMCOCOMO. Nevertheless, eSOMCOCOMO has a lower Standard Deviation and a greater value for the Correlation Coefficient.

E. RESIDUAL ANALYSIS

The residual plots (Figures 8 and 9) show the relationship between independent variable project size (KLOC) on the x-axis and residuals on the y-axis, where residuals express the differences between actual and predicted effort values. In plots, residuals were transformed by non-standardized transformation log 10 [44]; specifically, a symmetric log 10 was applied, and the residuals were increased by one. Each subplot plots the residuals of the eSOMCOCOMO and the original COCOMO model. In EXP A1-A3, only the organic mode of the basic COCOMO model was plotted.

Residuals near the zero residual line indicate correct prediction, residuals above the zero line mean underestimating,

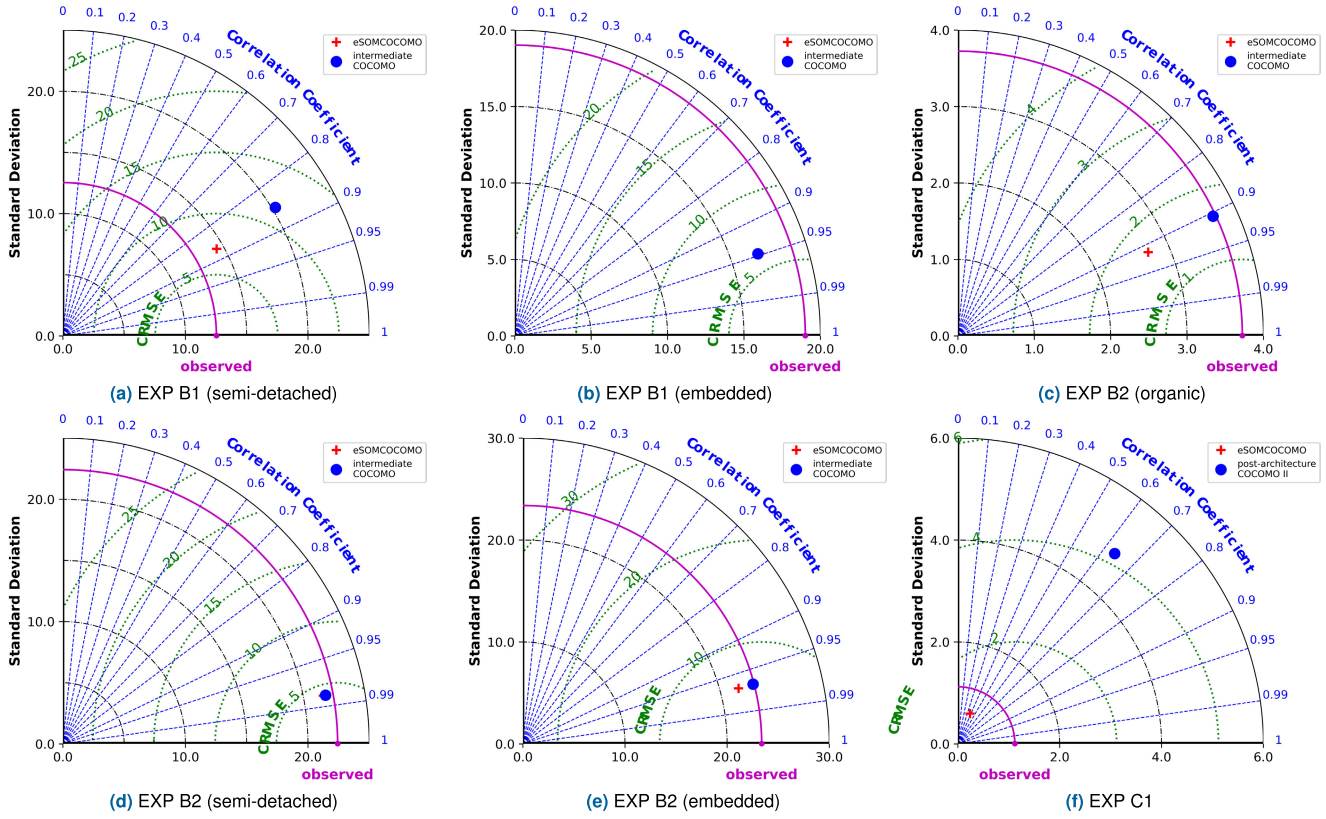

FIGURE 7. Taylor diagrams for EXP B1-B2 and EXP C1 under non-standardized square root transformation.

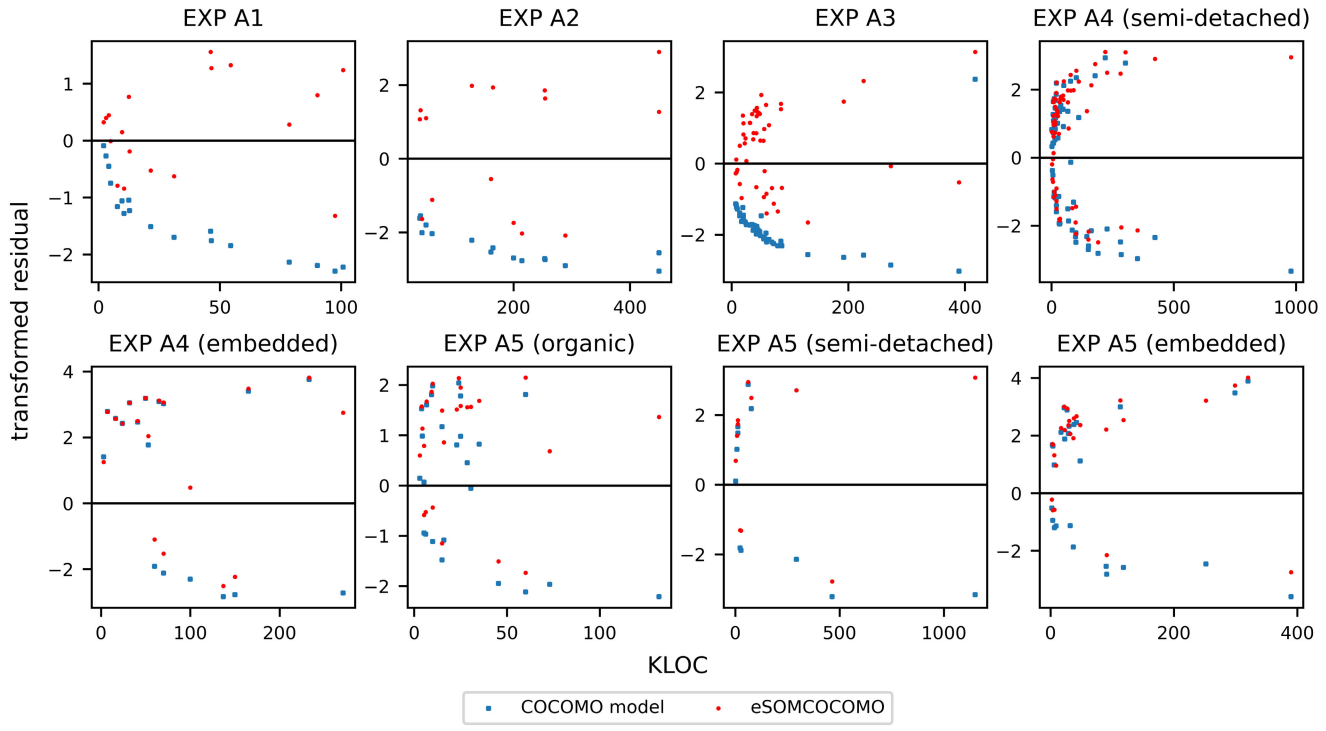

FIGURE 8. Residual plots for EXP A1-A5.

and residuals under the zero residual line mean overestimating. The Figures 8 and 9 indicate that the predictions made by eSOMCOCOMO in comparison to the original COCOMO model tend to be underestimated (EXP A1, EXP A2, and

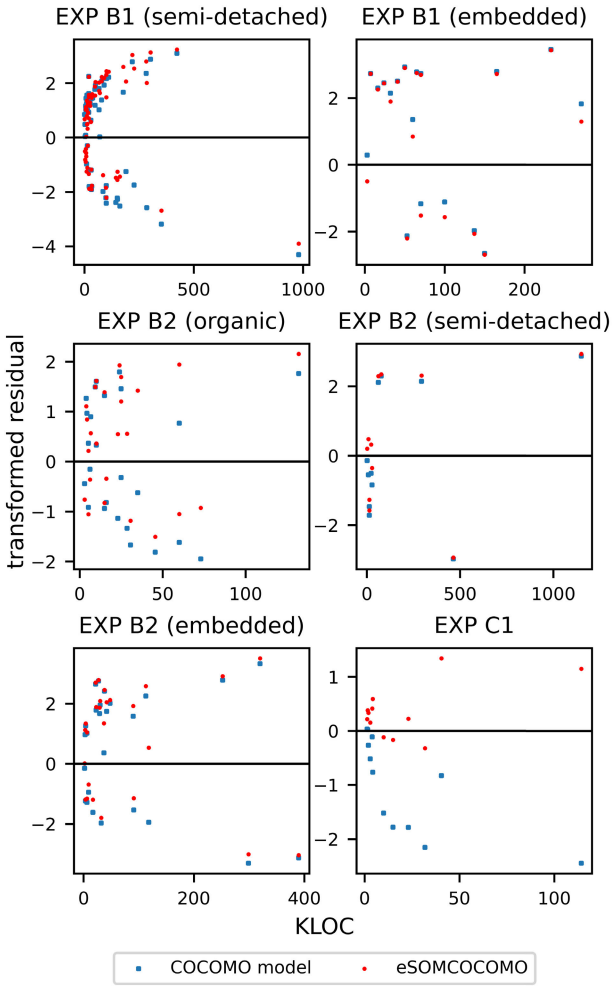

FIGURE 9. Residual plots for EXP B1-B2 and EXP C1.

EXP C1). However, the estimates get closer to the zero residual line; they show some refinement. Moreover, a pattern can be recognized in, for instance, EXP A3, EXP A4 (semi-detached), and EXP A5 (embedded). With the project size increasing, the estimate gets more inaccurate.

F. STATISTICAL EVALUATION OF HYPOTHESES

The Wilcoxon signed-rank test was used to determine if eSOMCOCOMO predicts more accurately than the original COCOMO and COCOMO II models. Specifically, a one-sided Wilcoxon signed-rank test was performed to check if the median of differences is significantly lower (or greater) than zero. The findings are displayed in Tables 19-24, where the superiority of the eSOMCOCOMO model was assessed using the evaluation metrics MMRE, PRED(0.25), MAE, and RMSE. The bolded p-value indicates that the value is less than or equal to the significance level of 0.05. In addition to the original models, the tables also include the remaining benchmark models.

Based on the analysis of the results, there was a significant improvement in the prediction of eSOMCOCOMO compared to the original COCOMO models in all evaluation metrics only in experiments EXP A1-A3. We reject the null hypothesis (H0) and accept the alternative hypothesis (H1) in these experiments. In EXP C1 and EXP B1 (embedded), a significant improvement can be observed only in the MAE and RMSE metrics.

From the perspective of the other benchmark models, eSOMCOCOMO has superiority in the performance in all evaluation metrics when compared to experiment EXP A1 (Walston-Felix and Halstead models), experiment EXP A2 (Halstead model), and experiment EXP A3–A4 (Bailey-Basili, Walston-Felix, and Halstead models). Statistical significance of the superiority of eSOMCOCOMO was also confirmed in EXP A4, namely in PSO (RMSE) and in GA (MMRE, MAE, RMSE). Moreover, for the genetic algorithm, the p-value ≤ 0.05 was in EXP B1 (semi-detached mode: MAE, RMSE), in EXP B2 (organic and semi-detached modes: RMSE), and in EXP C1 (RMSE). The superiority of the eSOMCOCOMO model can also be observed in EXP A5 for baseline models and metaheuristic algorithms.

G. THREATS TO VALIDITY

We encountered several challenges in enhancing software effort estimation with the Self-Organizing Migration Algorithm (SOMA)-optimized Constructive Cost Model (COCOMO). We identified potential threats to the validity of our study. These considerations are crucial for interpreting the results and understanding the practical limitations of applying the proposed model in various software development contexts.

1) *Dataset Diversity and Size:* The effectiveness of the SOMA-optimized COCOMO models was tested across various datasets, including NASA93, NASA63, and Turkish datasets. While these datasets provide a varied basis for analysis, the generalizability of our findings may be limited by the specific characteristics of these datasets, such as project type, size, and domain. Additionally, the relatively small number of projects in certain datasets, particularly when divided by development mode, may not fully capture the breadth of software development projects in the industry.
2) *Model Complexity and Parameterization:* The COCOMO models, both traditional and SOMA-optimized, rely on accurately estimating several parameters, including size (KLOC), effort multipliers, and scale factors. The complexity of these models and the potential for inaccuracies in parameter estimation challenge the reliability of effort predictions. This complexity is further compounded in the SOMA-optimized models due to the additional layer of algorithmic optimization.
3) *Human Factors:* The role of human judgment in the estimation process, even when using algorithmic models, cannot be understated. Biases, varying levels of expertise, and differences in interpreting project requirements can significantly influence the accuracy

of input parameters, thus affecting the overall effectiveness of the COCOMO models.

H. PRIMARY CHALLENGES IN PRACTICAL APPLICATIONS

1) *Parameter Accuracy:* One of the fundamental challenges encountered in applying the SOMA-optimized COCOMO models is the accurate determination of input parameters. Misestimations can lead to significant errors in effort predictions, highlighting the need for extensive domain knowledge and careful project analysis.
2) *Algorithmic Complexity:* While the SOMA optimization algorithm enhances the predictive accuracy of COCOMO models, it also introduces additional complexity to the estimation process. The selection of appropriate SOMA parameters and the interpretation of optimization results require a deep understanding of both the algorithm and the underlying estimation model.
3) *Scalability to Large Projects:* Our study observed performance variances in specific scenarios, particularly in estimating efforts for large-scale projects. This limitation points to further refinement of the SOMA-optimized COCOMO models to improve their scalability and accuracy in estimating efforts for larger and more complex projects.
4) *Adaptability to Rapidly Changing Technologies:* The rapid evolution of software development technologies and methodologies poses a challenge to the long-term applicability of any effort estimation model. Ensuring that the SOMA-optimized COCOMO models remain relevant and accurate in the face of technological advancements requires ongoing research and model updates.

VI. CONCLUSION AND FUTURE WORK

This study analyses software effort estimation accuracy enhancement using a Self-Organizing Migration Algorithm (SOMA)-Optimized Constructive Cost Model (COCOMO). By conducting a comparative analysis of traditional COCOMO models and SOMA-optimized variants across preprocessed datasets (NASA93, NASA63, NASA18, Kemerer, Miyazaki94, and Turkish), our research focuses on evaluation metrics including Mean Magnitude of Relative Error (MMRE), Prediction at 0.25 (PRED(0.25)), Mean Absolute Error (MAE), and Root Mean Square Error (RMSE). The analysis encompasses various configurations of COCOMO models—basic, intermediate, and post-architecture COCOMO II, supplemented with additional statistical testing and residual analysis for in-depth insights. Two research questions have been addressed.

Response to RQ1: The application of SOMA has been shown to enhance the accuracy of effort estimations in the COCOMO models. The eSOMCOCOMo approach was tested in eight experiments (Table 9), and their results are shown in Tables 11-18. Based on the results, EXP A1, EXP A2, EXP A3, and EXP C1 provide the best evidence of the efficacy of the eSOMCOCOMO. In these experiments, there was a significant reduction in MMRE, MAE, and RMSE and an increase in PRED(0.25) compared to the original COCOMO models. These outstanding results can also be seen in the Taylor diagrams (Figures 6 and 7). In experiments considering development modes (EXP A4, EXP A5, EXP B1, EXP B2), a reduction in error can also be seen, especially in MMRE. However, these experiments demonstrate a less significant improvement, and in some cases, the results of MAE, RMSE, and PRED(0.25) are almost comparable. In the study, the results were also assessed using residual analysis, where, e.g., in experiments EXP A3, EXP A4 (semi-detached), and EXP A5 (embedded), the prediction inaccuracy increases with project size.

The findings show that the SOMA-optimized COCOMO models outperform conventional models in terms of predictive accuracy. This is particularly evident in metrics like MMRE, where improvements of up to 12

Response to RQ2: The optimization of the constants (a, b) using SOMA not only improved the estimation precision but did so with statistical significance, affirming hypothesis H1 in some experiments and indicating a substantial reduction in estimation errors as measured by our selected metrics. The results in some experiments lead to the rejection of the null hypothesis (H0), which stated that SOMA would not improve estimation accuracy, and support the acceptance of the alternative hypothesis (H1). Tables 19-24 provide statistical testing results between eSOMCOCOMO and other benchmark models.

Despite the progress made, the study revealed underestimation and overestimation issues, particularly in larger projects, pointing to the need for further research.

For our future research a key area of focus lies in the expansive exploration of metaheuristic algorithms beyond SOMA. Comparative analyses of algorithms such as Genetic Algorithms, Ant Colony Optimization, and Firefly Algorithms are encouraged. The objectives are twofold: to evaluate their effectiveness in software effort estimation and to understand their unique strengths and weaknesses in this context. This exploration is expected to illuminate algorithm-specific applications capable of addressing the deficiencies identified in current practices.

Moreover, the role of fitness functions in the optimization process warrants further investigation. There is a compelling need to experiment with various fitness functions to identify those that most effectively bridge the gap between estimated and actual efforts. The development of bespoke fitness functions, tailored to the nuances of software effort estimation, could markedly enhance model accuracy.

Additionally, integrating multi-objective optimization techniques and examining hybrid metaheuristic approaches present promising avenues for advancing the optimization process. These strategies aim to balance multiple criteria reflective of the complex nature of software projects,

TABLE 19. EXP A1-A3: Wilcoxon signed-rank test.

Stats eSOMCOCOMO &	EXP A1: p-value				EXP A2: p-value				EXP A3: p-value			
	MMRE	PRED (0.25)	MAE	RMSE	MMRE	PRED (0.25)	MAE	RMSE	MMRE	PRED (0.25)	MAE	RMSE
basic COCOMO (organic)	**7.25e-05**	**3.39e-04**	**3.81e-05**	**3.81e-05**	**1.53e-04**	**3.58e-03**	**1.01e-03**	**2.69e-03**	**4.87e-13**	**3.16e-05**	**1.30e-09**	**2.15e-09**
basic COCOMO (semi-detached)	**3.81e-06**	**5.66e-05**	**3.81e-06**	**3.81e-06**	**3.05e-05**	**3.58e-03**	**3.05e-05**	**3.05e-05**	**7.11e-15**	**3.35e-06**	**4.04e-11**	**5.99e-11**
basic COCOMO (organic)	**3.81e-06**	**5.66e-05**	**3.81e-06**	**3.81e-06**	**3.05e-05**	**3.58e-03**	**3.05e-05**	**3.05e-05**	**3.55e-15**	**3.35e-06**	**3.55e-15**	**3.55e-15**
Walston-Felix	**1.91e-05**	**5.66e-05**	**1.91e-05**	**3.81e-05**	1.80e-01	6.80e-02	**3.19e-02**	**1.28e-02**	**6.34e-05**	**1.26e-03**	**2.13e-06**	**3.85e-07**
Bailey-Basili	5.84e-01	2.03e-01	4.33e-01	4.83e-01	**3.19e-02**	1.67e-01	**4.73e-02**	6.77e-02	**3.88e-05**	**7.07e-02**	**2.57e-02**	**3.26e-02**
Halstead	**3.81e-06**	**5.66e-05**	**3.81e-06**	**3.81e-06**	**3.05e-05**	**3.58e-03**	**3.05e-05**	**3.05e-05**	**3.55e-15**	**3.35e-06**	**3.55e-15**	**3.55e-15**
PSO	3.51e-01	5.00e-01	5.94e-02	9.82e-02	6.19e-01	1.02e-01	1.38e-01	1.51e-01	7.88e-01	5.00e-01	9.79e-01	9.82e-01
GA	3.67e-01	2.03e-01	2.48e-01	2.34e-01	5.33e-01	**3.97e-02**	1.26e-01	1.15e-01	6.29e-01	2.98e-01	7.73e-01	9.20e-01

TABLE 20. EXP A4: Wilcoxon signed-rank test.

Mode	Stats eSOMCOCOMO &	EXP A4: p-value			
		MMRE	PRED (0.25)	MAE	RMSE
semi-detached	basic COCOMO	2.49e-01	4.25e-01	6.07e-01	5.85e-01
	Walston-Felix	**1.20e-12**	**1.69e-08**	**1.63e-12**	**5.20e-12**
	Bailey-Basili	**1.79e-07**	**1.73e-07**	**5.54e-09**	**4.98e-09**
	Halstead	**4.37e-13**	**1.69e-08**	**4.36e-13**	**4.77e-13**
	PSO	8.45e-01	5.00e-01	9.21e-01	8.40e-01
	GA	5.70e-01	5.00e-01	9.90e-01	9.93e-01
embedded	basic COCOMO	1.96e-01	1.16e-01	5.84e-01	7.52e-01
	Walston-Felix	**3.81e-06**	**1.76e-02**	**3.81e-06**	**3.81e-06**
	Bailey-Basili	**3.81e-06**	**1.76e-02**	**3.81e-06**	**3.81e-06**
	Halstead	**1.92e-02**	**4.56e-02**	**1.18e-02**	**1.34e-02**
	PSO	1.73e-01	5.00e-01	5.94e-02	**5.20e-03**
	GA	**4.49e-02**	3.31e-01	**1.92e-02**	**2.37e-03**

TABLE 21. EXP A5: Wilcoxon signed-rank test.

Mode	Stats eSOMCOCOMO &	EXP A5: p-value			
		MMRE	PRED (0.25)	MAE	RMSE
organic	basic COCOMO	3.56e-01	7.98e-01	7.95e-01	8.79e-01
	Walston-Felix	**1.71e-03**	8.63e-02	**2.78e-04**	**9.39e-05**
	Bailey-Basili	2.29e-01	6.38e-01	**1.71e-02**	**7.36e-03**
	Halstead	**1.64e-06**	1.61e-01	**5.04e-06**	**1.33e-05**
	PSO	9.09e-02	5.00e-01	**3.55e-02**	**1.97e-02**
	GA	1.05e-01	3.62e-01	**2.30e-03**	**8.66e-03**
semi-detached	basic COCOMO	5.84e-01	9.19e-01	5.84e-01	5.51e-01
	Walston-Felix	**4.88e-04**	2.96e-01	**4.88e-04**	**4.88e-04**
	Bailey-Basili	3.82e-01	6.79e-01	1.20e-01	8.74e-02
	Halstead	**4.88e-04**	**4.88e-04**	**4.88e-04**	2.96e-01
	PSO	4.49e-01	5.00e-01	2.07e-01	1.60e-01
	GA	6.50e-01	6.79e-01	4.49e-01	6.50e-01
embedded	basic COCOMO	8.54e-01	6.21e-01	9.85e-01	9.92e-01
	Walston-Felix	**4.92e-07**	1.49e-01	**1.64e-06**	**3.09e-06**
	Bailey-Basili	**9.86e-04**	5.00e-01	**2.95e-04**	**7.24e-05**
	Halstead	**7.99e-06**	2.51e-01	**1.82e-04**	**3.32e-04**
	PSO	8.25e-02	5.00e-01	**8.02e-04**	**2.06e-04**
	GA	6.79e-02	3.55e-01	**8.02e-04**	**2.06e-04**

TABLE 22. EXP B1: Wilcoxon signed-rank test.

Mode	Stats eSOMCOCOMO &	EXP B1: p-value			
		MMRE	PRED (0.25)	MAE	RMSE
semi-detached	intermediate COCOMO	1.93e-01	4.31e-01	5.32e-01	5.44e-01
	PSO	4.95e-01	4.06e-01	9.92e-01	9.96e-01
	GA	6.88e-02	4.13e-01	**1.96e-03**	**1.64e-04**
embedded	intermediate COCOMO	4.49e-01	5.00e-01	**4.49e-02**	**1.71e-02**
	PSO	6.95e-01	5.00e-01	6.80e-01	6.33e-01
	GA	6.65e-01	3.31e-01	7.25e-01	8.15e-01

TABLE 23. EXP B2: Wilcoxon signed-rank test.

Mode	Stats eSOMCOCOMO &	EXP B2: p-value			
		MMRE	PRED (0.25)	MAE	RMSE
organic	intermediate COCOMO	3.46e-01	6.22e-01	5.11e-01	4.37e-01
	PSO	4.58e-01	3.53e-01	3.46e-01	3.08e-01
	GA	7.02e-01	3.53e-01	5.37e-02	**3.13e-02**
semi-detached	intermediate COCOMO	2.32e-01	2.96e-01	6.18e-01	6.18e-01
	PSO	3.50e-01	5.00e-01	1.60e-01	1.03e-01
	GA	1.03e-01	8.14e-02	8.74e-02	7.37e-02
embedded	intermediate COCOMO	2.42e-01	7.49e-01	9.09e-01	9.45e-01
	PSO	1.64e-01	5.00e-01	9.07e-02	8.65e-02
	GA	3.09e-01	2.63e-01	**2.32e-02**	**1.56e-02**

TABLE 24. EXP C1: Wilcoxon signed-rank test.

Stats eSOMCOCOMO &	EXP C1: p-value			
	MMRE	PRED(0.25)	MAE	RMSE
post-architecture COCOMO II	6.47e-02	3.58e-01	**4.61e-02**	**4.61e-02**
Local Calibration	3.39e-01	3.58e-01	3.96e-01	6.04e-01
PSO	3.39e-01	5.00e-01	2.85e-01	8.81e-02
GA	2.85e-01	3.03e-01	2.35e-01	**3.86e-02**

fostering the development of more sophisticated and effective estimation models.

The application of clustering techniques to develop more personalized and context-aware estimation models also holds significant potential. By catering to the unique attributes of software projects, these approaches could revolutionize software effort estimation, yielding more accurate and reliable predictions. Furthermore, exploring metaheuristic algorithms in other areas of software effort estimation promises to unveil novel methodologies and techniques, contributing fresh insights and approaches to the field. In summary a future research can be described as:

1) Explore and compare the efficacy of various metaheuristic algorithms beyond SOMA in software effort estimation.

2) Investigate and develop tailored fitness functions to enhance the accuracy of effort estimation models.
3) Integrate multi-objective optimization and hybrid metaheuristic approaches to refine the estimation process.
4) Apply clustering techniques for personalized models and explore metaheuristic applications in novel estimation areas.

In conclusion, while this research has significantly applied the SOMA algorithm to improve estimation accuracy, the field remains ripe for innovation. The insights gained lay the groundwork for a new wave of research aimed at overcoming the current limitations and unlocking even more precise and reliable estimation methods.

REFERENCES

[1] S. McConnell, *Software Estimation: Demystifying the Black Art*, 1st ed. Redmond, WA, USA: Microsoft Press, Mar. 2006.
[2] R. Silhavy, P. Silhavy, and Z. Prokopova, "Using actors and use cases for software size estimation," *Electronics*, vol. 10, no. 5, p. 592, Mar. 2021.
[3] B. W. Boehm, "Software engineering economics," *IEEE Trans. Softw. Eng.*, vol. SE-10, no. 1, pp. 4–21, Jan. 1984.
[4] B. Boehm, B. Clark, E. Horowitz, C. Westland, R. Madachy, and R. Selby, "Cost models for future software life cycle processes: COCOMO 2.0," *Ann. Softw. Eng.*, vol. 1, no. 1, pp. 57–94, Dec. 1995.
[5] Y. Mahmood, N. Kama, A. Azmi, A. S. Khan, and M. Ali, "Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation," *Softw., Pract. Exper.*, vol. 52, no. 1, pp. 39–65, Jan. 2022, doi: 10.1002/spe.3009.
[6] M. Birattari, L. Paquete, and T. Stützle, "Classification of metaheuristics and design of experiments for the analysis of components," Intellektik, Darmstadt, Germany, Tech. Rep. AIDA-01-05, 2003.
[7] A. A. Fadhil, R. G. H. Alsarraj, and A. M. Altaie, "Software cost estimation based on dolphin algorithm," *IEEE Access*, vol. 8, pp. 75279–75287, 2020.
[8] I. M. M. Parwita, R. Sarno, and A. Puspaningrum, "Optimization of COCOMO II coefficients using cuckoo optimization algorithm to improve the accuracy of effort estimation," in *Proc. 11th Int. Conf. Inf. Commun. Technol. Syst. (ICTS)*, Oct. 2017, pp. 99–104.
[9] K. Langsari and R. Sarno, "Optimizing COCOMO II parameters using particle swarm method," in *Proc. 3rd Int. Conf. Sci. Inf. Technol. (ICSITech)*, Oct. 2017, pp. 29–34.
[10] R. K. Sachan, A. Nigam, A. Singh, S. Singh, M. Choudhary, A. Tiwari, and D. S. Kushwaha, "Optimizing basic COCOMO model using simplified genetic algorithm," *Proc. Comput. Sci.*, vol. 89, pp. 492–498, Jan. 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050916311723
[11] N. M. Alsheikh and N. M. Munassar, "Improving software effort estimation models using grey wolf optimization algorithm," *IEEE Access*, vol. 11, pp. 143549–143579, 2023.
[12] A. F. Sheta, "Estimation of the COCOMO model parameters using genetic algorithms for NASA software projects," *J. Comput. Sci.*, vol. 2, no. 2, pp. 118–123, Feb. 2006.
[13] T. J. Gandomani, M. Dashti, and M. Z. Nafchi, "Hybrid genetic-environmental adaptation algorithm to improve parameters of COCOMO for software cost estimation," in *Proc. 2nd Int. Conf. Distrib. Comput. High Perform. Comput. (DCHPC)*, Mar. 2022, pp. 82–85.
[14] C. Anwar ul Hassan and M. Sufyan Khan, "An effective nature inspired approach for the estimation of software development cost," in *Proc. 16th Int. Conf. Emerg. Technol. (ICET)*, Dec. 2021, pp. 1–6.
[15] S. Kumari and S. Pushkar, "Software cost estimation using cuckoo search," in *Advances in Computational Intelligence*, vol. 509, S. K. Sahana and S. K. Saha, Eds. Singapore: Springer, 2017, pp. 167–175.
[16] P. Singal, A. C. Kumari, and P. Sharma, "Estimation of software development effort: A differential evolution approach," *Proc. Comput. Sci.*, vol. 167, pp. 2643–2652, Jan. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050920308097
[17] S. P. Singh, V. P. Singh, and A. K. Mehta, "Differential evolution using homeostasis adaption based mutation operator and its application for software cost estimation," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 33, no. 6, pp. 740–752, Jul. 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157818300910
[18] S. P. Singh and A. Kumar, "Software cost estimation using homeostasis mutation based differential evolution," in *Proc. 11th Int. Conf. Intell. Syst. Control (ISCO)*, Jan. 2017, pp. 173–181.
[19] B. Boehm, C. Abts, B. Clark, S. Devnani-Chulani, E. Horowitz, R. J. Madachy, and B. Steece, "COCOMO II model definition manual, version 1.4," TR Center Softw. Eng., Univ. Southern California, Los Angeles, CA, USA, 2000.
[20] B. Boehm, C. Abts, and S. Chulani, "Software development cost estimation approaches—A survey," *Ann. Softw. Eng.*, vol. 10, no. 1/4, pp. 177–205, Nov. 2000, doi: 10.1023/a:1018991717352.
[21] I. Zelinka, *SOMA—Self-Organizing Migrating Algorithm*. Berlin, Heidelberg: Springer, 2004, pp. 167–217, doi: 10.1007/978-3-540-39930-8_7.
[22] D. Davendra and I. Zelinka, *Self-Organizing Migrating Algorithm* (Studies in Computational Intelligence), vol. 626. Cham, Switzerland: Springer, 2016. [Online]. Available: http://link.springer.com/10.1007/978-3-319-28161-2
[23] J. S. Shirabad and T. J. Menzies, "The PROMISE repository of software engineering databases," School Inf. Technol. Eng., Univ. Ottawa, Ottawa, Canada, 2005. [Online]. Available: http://promise.site.uottawa.ca/SERepository
[24] J. W. Keung, "Kemerer [Data Set]," Zenodo, 2010. [Online]. Available: https://doi.org/10.5281/zenodo.268464
[25] S. Amasaki, "Miyazaki94 [Data set]," Zenodo, 2016. [Online]. Available: https://doi.org/10.5281/zenodo.268473
[26] T. Menzies, R. Krishna, and D. Pryor. (2017). *The Seacraft Repository of Empirical Software Engineering Data*. [Online]. Available: https://zenodo.org/communities/seacraft
[27] J. W. Bailey and V. R. Basili, "A meta-model for software development resource expenditures," in *Proc. 5th Int. Conf. Softw. Eng.*, 1981, pp. 107–116.
[28] J. W. Tukey, *Exploratory Data Analysis* (Addison-Wesley Series in Behavioral Science), 1st ed. Upper Saddle River, NJ, USA: Pearson, Jan. 1977.
[29] D. C. Hoaglin, B. Iglewicz, and J. W. Tukey, "Performance of some resistant rules for outlier labeling," *J. Amer. Stat. Assoc.*, vol. 81, no. 396, p. 991, Dec. 1986. [Online]. Available: http://www.jstor.org/stable/2289073
[30] B. A. Kitchenham, L. M. Pickard, S. G. MacDonell, and M. J. Shepperd, "What accuracy statistics really measure [software estimation]," *IEE Proc.-Softw.*, vol. 148, no. 3, pp. 81–85, Jun. 2001.
[31] I. Myrtveit, E. Stensrud, and M. Shepperd, "Reliability and validity in comparative studies of software prediction models," *IEEE Trans. Softw. Eng.*, vol. 31, no. 5, pp. 380–391, May 2005.
[32] M. Shepperd and S. MacDonell, "Evaluating prediction systems in software project estimation," *Inf. Softw. Technol.*, vol. 54, no. 8, pp. 820–827, Aug. 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095058491200002X
[33] P. Silhavy, R. Silhavy, and Z. Prokopova, "Evaluation of data clustering for stepwise linear regression on use case points estimation," in *Software Engineering Trends and Techniques in Intelligent Systems* (Advances in Intelligent Systems and Computing), vol. 575, R. Silhavy, P. Silhavy, Z. Prokopova, R. Senkerik, and Z. K. Oplatkova, Eds. Cham, Switzerland: Springer, 2017. [Online]. Available: https://doi.org/10.1007/978-3-319-57141-6_52
[34] R. Silhavy, P. Silhavy, and Z. Prokopova, "Improving algorithmic optimisation method by spectral clustering," in *Software Engineering Trends and Techniques in Intelligent Systems* (Advances in Intelligent Systems and Computing), vol. 575, R. Silhavy, P. Silhavy, Z. Prokopova, R. Senkerik, and Z. K. Oplatkova, Eds. Cham, Switzerland: Springer, 2017. [Online]. Available: https://doi.org/10.1007/978-3-319-57141-6_1
[35] V. V. Hai, H. L. T. K. Nhung, Z. Prokopova, R. Silhavy, and P. Silhavy, "A new approach to calibrating functional complexity weight in software development effort estimation," *Computers*, vol. 11, no. 2, p. 15, Jan. 2022.
[36] H. L. T. K. Nhung, V. Van Hai, R. Silhavy, Z. Prokopova, and P. Silhavy, "Parametric software effort estimation based on optimizing correction factors and multiple linear regression," *IEEE Access*, vol. 10, pp. 2963–2986, 2022.
[37] F. Wilcoxon, *Individual Comparisons By Ranking Methods*. New York, NY, USA: Springer, 1992, pp. 196–202, doi: 10.1007/978-1-4612-4380-9_16.

[38] B. KumarSingh and A. K. Misra, "Software effort estimation by genetic algorithm tuned parameters of modified constructive cost model for NASA software projects," *Int. J. Comput. Appl.*, vol. 59, no. 9, pp. 22–26, Dec. 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:7832996
[39] M. Baiquni and R. Sarno, "Improving the accuracy of COCOMO II using fuzzy logic and local calibration method," in *Proc. 3rd Int. Conf. Sci. Inf. Technol. (ICSITech)*, Oct. 2017, pp. 284–289.
[40] C. E. Walston and C. P. Felix, "A method of programming measurement and estimation," *IBM Syst. J.*, vol. 16, no. 1, pp. 54–73, 1977.
[41] A. Sheta, D. Rine, and A. Ayesh, "Development of software effort and schedule estimation models using soft computing techniques," in *Proc. IEEE Congr. Evol. Comput., IEEE World Congr. Comput. Intell.*, Jun. 2008, pp. 1283–1289.
[42] P. A. Rochford. (2016). *Skillmetrics: A Python Package for Calculating the Skill of Model Predictions Against Observations*. [Online]. Available: http://github.com/PeterRochford/SkillMetrics
[43] K. E. Taylor, "Summarizing multiple aspects of model performance in a single diagram," *J. Geophys. Res., Atmos.*, vol. 106, no. D7, pp. 7183–7192, Apr. 2001.
[44] G. E. Gignac, *How2statsbook*, G. E. Gignac, Ed. Perth, WA, Australia: G. E. Gignac, 2023, ch. 2. [Online]. Available: http://www.how2statsbook.com/p/chapters.html

DARINA BAJUSOVA was born in Vranov nad Topl'ou, Slovak Republic, in 1996. She received the M.Sc. degree in engineering informatics from the Faculty of Applied Informatics, Tomas Bata University in Zlín, in 2021, where she is currently pursuing the Ph.D. degree. Since 2021, she has been an Assistant Researcher with the Faculty of Applied Informatics, Tomas Bata University in Zlín. Her research interests include effort estimation in software engineering and database systems.

PETR SILHAVY received the Ph.D. degree in engineering informatics from the Faculty of Applied Informatics, Tomas Bata University in Zlín, Zlín, Czech Republic. He is currently an Associate Professor with the Faculty of Applied Informatics, Tomas Bata University in Zlín. He is also a Senior Researcher and an Associate Professor of system engineering and informatics with a demonstrated history of working in research and higher education. He has expertise as a CTO and a Software Developer in database programming, database design, data management, and data science. His research interests include prediction and empirical methods for software engineering.

RADEK SILHAVY received the Ph.D. degree in engineering informatics from the Faculty of Applied Informatics, Tomas Bata University in Zlín, Zlín, Czech Republic, in 2009. He is currently an Associate Professor and a Senior Researcher with the Faculty of Applied Informatics, Tomas Bata University in Zlín. He is also an Associate Professor of system engineering and informatics with a demonstrated history of working in research, higher education, project management, and software analysis. He is involved in academic publishing as the Editor-in-Chief, an editor, or a reviewer. His research interests include predictive analytics for software engineering, empirical methods in software engineering, or prediction models focused on cost, size, and effort estimations in system/software engineering.