Computer Systems Science and Engineering

Tech Science Press

DOI: 10.32604/csse.2024.055612

ARTICLE

Software Cost Estimation Using Social Group Optimization

Sagiraju Srinadhraju*, **Samaresh Mishra and Suresh Chandra Satapathy**

School of Computer Engineering, KIIT Deemed to be University, Bhubaneswar, 751 024, India

*Corresponding Author: Sagiraju Srinadhraju. Email: 2181073@kiit.ac.in

Received: 02 July 2024 Accepted: 09 September 2024 Published: 22 November 2024

ABSTRACT

This paper introduces the integration of the Social Group Optimization (SGO) algorithm to enhance the accuracy of software cost estimation using the Constructive Cost Model (COCOMO). COCOMO's fixed coefficients often limit its adaptability, as they don't account for variations across organizations. By fine-tuning these parameters with SGO, we aim to improve estimation accuracy. We train and validate our SGO-enhanced model using historical project data, evaluating its performance with metrics like the mean magnitude of relative error (MMRE) and Manhattan distance (MD). Experimental results show that SGO optimization significantly improves the predictive accuracy of software cost models, offering valuable insights for project managers and practitioners in the field. However, the approach's effectiveness may vary depending on the quality and quantity of available historical data, and its scalability across diverse project types and sizes remains a key consideration for future research.

KEYWORDS

Manhattan distance; mean magnitude of relative error; nature-inspired algorithms; project management; SGO

1 Introduction

The ability to accurately estimate the cost of developing software has become a paramount concern for project managers, stakeholders, and organizations [1–3]. Software cost estimation serves as the foundation for project planning, resource allocation, and decision-making, and its accuracy can significantly impact project success and profitability. Accurate estimation significantly enhances the efficiency and precision of software projects, encompassing aspects like resource allocation, minimizing project failures, determining the required level of reliability, tool selection, programmer proficiency, and more [4,5]. There are two types of software cost estimation techniques: algorithmic and non-algorithmic. It is standard practice to use non-algorithmic techniques like analogy-based estimating, top-down and bottom-up estimation, Parkinson's Law, and price-to-win. Additionally, algorithmic methods include Software Engineering Laboratory (SEL), Constructive Cost Model-II (COCOMO-II), Halstead, Doty, Putnam, Baily Basil, Walston Filex Model, COCOMO, and Halstead, Doty, Putnam. One of the most used algorithms for algorithmic estimate is COCOMO-II. However, its accuracy remains an ongoing concern. To enhance the precision of this model, various algorithms have undergone rigorous investigation. The field of computational intelligence has made substantial contributions to improving software cost estimation. Particle Swarm Optimization [6], Genetic algorithm [7], Firefly algorithm [8], Differential Evolution [9] and many others have shown their capability while to estimate software cost.

One such technique that has gained prominence in recent years is the Social Group Optimization (SGO) algorithm [10]. SGO is a nature-inspired optimization algorithm inspired by the social behavior of human for solving complex problems. It leverages the collective intelligence of a group of persons to solve complex optimization problems.

Table 1 presents a summary of research contributions and published papers that have employed the SGO method, based on information gathered from the literature.

Table 1: Overview of application of SGO method

Study	Algorithm	Application	Outcome
Das et al. [11]	Modified Social Group Optimization (MSGO)	Civil engineering structure damage identification	Improved accuracy in identifying damage
Naik et al. [12]	Social Group Optimization (SGO)	Global optimization of multimodal functions and data clustering	Significant results due to SGO's excellent optimization capabilities
Rani et al. [13]	SGO-based test generation method (SGO-MT)	Software fault detection	Achieved expected results in detecting software faults
Verma et al. [14]	Discrete Social Group Optimization (DSGO)	Traveling salesman problem	Effectively solved the traveling salesman problem
Naik et al. [15]	Binary Social Group Optimization (BSGO)	Binary 0–1 knapsack problem	Achieved high-quality solutions and demonstrated superiority over other binary algorithms
Naik et al. [16]	Modified SGO with new probability factor (MSGO)	Short-term hydrothermal scheduling	Provided better solutions compared to other algorithms
Monisha et al. [17]	SGO with Shannon function	Thresholding for benchmark RGB images	Verified that SGO combined with Shannon functions works best
Reddy et al. [18]	Multi-Strategy Ensemble SGO (ME-SGO)	Engineering problems and electric vehicle optimization	Provided a reliable solution by overcoming SGO's drawbacks
Manic et al. [19]	SGO with Tsallis entropy	Brain MRI abnormal region segmentation detection	Enhanced detection efficiency in brain abnormality detection
Parwekar et al. [20]	SGO for energy consumption optimization	Wireless sensor networks	Reduced transmission distance and energy consumption at the node

(Continued)

Table 1 (continued)

Study	Algorithm	Application	Outcome
Dey et al. [21]	SGO-assisted Kapur's entropy and morphological segmentation	COVID-19 infection detection from CT images	Enhanced segmentation accuracy for COVID-19 detection
Singh et al. [22]	Hybrid SGO and support vector classifier	COVID-19 infection detection from chest X-ray images	Improved detection accuracy using the hybrid method
Dey et al. [23]	SGO-assisted segmentation	Skin melanoma image analysis	Enhanced segmentation and evaluation of skin melanoma images
Akhtar et al. [24]	Hybrid differential evolution social group optimization	Non-instantaneous deteriorating inventory problem	Optimized inventory management with time and price-dependent demand
Kalananda et al. [25]	Social group whale optimization algorithm	Numerical and engineering optimization problems	Improved optimization performance for complex engineering problems
Reddy et al. [26]	SGO-assisted differential evolution	Photovoltaic (PV) parameter extraction	Enhanced PV parameter extraction for standard and modified diode models
Praveen et al. [27]	SGO	Resource allocation and task scheduling in cloud environment	Improved efficiency in resource allocation and task scheduling
Kraiem et al. [28]	Modified SGO	Photovoltaic cell and module models parameter identification	Achieved accurate parameter identification for PV models
Secui et al. [29]	Modified SGO	Economic emission dispatch with wind power	Solved the economic emission dispatch problem with wind power integration
Tran [30]	Multiple-objective SGO and multi-criteria decision-making	Time–cost optimization in construction projects	Optimized time-cost in construction projects using multi-objective SGO
Ullah et al. [31]	Hybrid SGO-Support Vector Machine (SVM)	Transformer fault diagnosis	Developed an optimal diagnosis model using improved SGO-SVM
Huynh et al. [32]	Multiple objective SGO	Construction project optimization	Optimized time–cost–quality–carbon dioxide in construction projects

(Continued)

Table 1 (continued)

Study	Algorithm	Application	Outcome
Naik [33]	Marine predators social group optimization	Marine predators and social group optimization hybrid	Applied hybrid approach for marine predators optimization
Vadivel et al. [34]	SGO-assisted MPPT	Maximum power point tracking in PV arrays	Improved MPPT scheme for partial shaded PV arrays
Wang et al. [35]	Dual-population SGO	Optimization based on human social group behavior	Enhanced performance with dual-population strategy
Naik [36]	Multi-objective SGO	Machining process optimization	Improved machining process through multi-objective optimization
Tran et al. [37]	SGO and smeared stiffener method	Optimization of stiffeners for laminated cylindrical panels	Achieved maximum fundamental frequency for stiffened panels
Garg et al. [38]	SGO-assisted image watermarking	Adaptive image watermarking	Developed a robust and secured watermarking method
Rahaman et al. [39]	SGO for charger-UAV scheduling	Wireless rechargeable sensor networks	Efficiently scheduled charger-UAVs in wireless sensor networks

From the literature, we found that no work has been done in the area of software cost estimation within the field of software engineering. This gap motivated us to address the software cost estimation problem using the SGO algorithm. Furthermore, the justification for Choosing SGO:

- **Efficiency in Complex Search Spaces:** SGO's social learning mechanism makes it highly effective in navigating complex, multi-dimensional search spaces, which are typical in software cost estimation models.
- **Robust Convergence:** SGO has shown strong convergence properties, particularly in avoiding local optima, which is crucial for achieving accurate and reliable estimates in software projects.
- **Flexibility and Adaptability:** SGO is adaptable to various types of optimization problems, including those with varying levels of complexity, which makes it particularly well-suited for fine-tuning parameters in diverse software cost estimation scenarios.
- **Balanced Exploration and Exploitation:** The inherent balance between exploration and exploitation in SGO ensures that the search process remains both thorough and efficient, leading to better overall performance compared to algorithms that might lean too heavily toward one aspect.

By choosing SGO, the paper leverages an algorithm that not only aligns well with the specific challenges of software cost estimation but also provides a robust, efficient, and adaptable approach that can outperform other metaheuristic algorithms in this context.

The objective of this research is to explore the potential of SGO in optimizing software cost estimation models, aiming to improve their accuracy and adaptability. Numerous well-established derivative-free methods have been employed to facilitate a fair comparison between the proposed approach and various other software effort estimation models, such as Doty, Interactive Voice Response (IVR), Halstead, Bailey Basil, and SEL, etc. Our experimentation leveraged two historical datasets, COCOMO81 and Turkish Industry software projects. The following are the main contributions of the paper:

- The SGO algorithm is used for the first time to determine the optimal parameter choices for COCOMO and COCOMO-II for software cost estimation.
- The experimental results of the proposed methodology are compared with a number of derivative free methods, such as the Genetic Algorithm (GA), Differential Evolution (DE), variants of DE, Biogeography-Based Optimisation (BBO), Tabu Search, Flower Pollination Algorithm (FPA), and Particle Swarm Optimisation (PSO), as well as other models of the same nature, such as Dosty, IVR, Halstead, Bailey Basil, and SEL.

The remainder of the paper is structured as follows: Introduction is presented in Section 1, related research about the estimation of software projects is covered in Section 2. In Section 3 the proposed approach is covered. Finally, the Sections 4 and 5 describe the results analysis, conclusion, and future directions, respectively.

2 Related Work

Several estimation techniques have been applied to enhance the optimization of coefficients in the COCOMO model [40,41]. These techniques encompass PSO, GA, Tabu Search, Intelligent Water Drop (IWD) algorithms, and more. In a separate investigation [42], the cuckoo search algorithm was employed to fine-tune COCOMO-II coefficients. Experiments were conducted on 18 datasets extracted from NASA-93 software projects to validate these approaches. The results unequivocally demonstrated that the cuckoo search algorithm outperformed Baily-basil, Doty, Hallstead, and the original COCOMO-II models. Building on this research, Kumari et al. [43] proposed an innovative hybrid approach that combined cuckoo search with artificial neural networks to optimize existing COCOMO-II parameters. Additionally, a hybrid technique was introduced in the paper [44], incorporating Tabu Search and GA to optimize COCOMO-II parameters. To improve the performance of the differential evolution technique, Urbanek et al. [45] combined analytical programming and differential evolution techniques for software cost estimation. In order to improve the existing coefficients of COCOMO-II, Dalal et al. [46] presented a generalized reduced gradient nonlinear optimization approach along with best-fit analysis, reporting better results than the original COCOMO-II model.

In a different context, Santos et al. [47] introduced COCOMO-II for effort estimation, employing an organizational case study within the aeronautical industry. Their research indicated that COCOMO-II provided a more accurate estimation technique for real software projects compared to alternative approaches. Additionally, Hughes [48] reviewed expert judgment as an estimating method to calculate software project costs. Effendi et al. [49] introduced bat algorithm to optimize COCOMO-II, while Ahmad et al. [50] devised a hybrid whale-crow optimized-based optimal regression method for estimating software project costs, relying on data from four software industries. Their results indicated that the proposed model outperformed other estimation models. Sheta et al. [51] introduced a soft-computing method for software project cost estimation, utilizing PSO to optimize COCOMO coefficients and incorporating fuzzy logic to create a set of linear methods over the domain of possible software lines of code (LOC). Their algorithms were compared to baseline methods such as HS, WF, BB, and Doty models, although COCOMO-II still exhibited limitations in terms of accuracy.

Despite various works done in the past to estimate the cost using varied algorithms, methods and hybrid approach, still there remains a continuing challenge to achieve better accuracy by tuning the co-efficient of COCOMO model. In this study, we present a novel Stochastic Global Optimization (SGO) approach for optimizing COCOMO-II coefficients. To validate its effectiveness, we compare the proposed algorithm with the conventional COCOMO-II model and several other baseline methods. The simulation results underscore the superior performance of the proposed algorithm in terms of both Mean Magnitude of Relative Error (MMRE) and Mean Deviation (MD) when compared to the original COCOMO-II, PSO, GA, and other baseline models. This highlights the potentials of SGO as a promising avenue for further improving software cost estimation accuracy.

2.1 COCOMO Model

The COCOMO model, initially developed by Boehm in 1981 [52], serves as an algorithmic approach employed for estimating project costs and effort. This model comprises three layers: basic, intermediate, and detail. In 1995, Boehm et al. introduced COCOMO-II [53], a refined version of COCOMO that offers superior software project cost estimation capabilities when compared to its predecessor. Below Fig. 1 illustrates the comprehensive process of COCOMO-II.

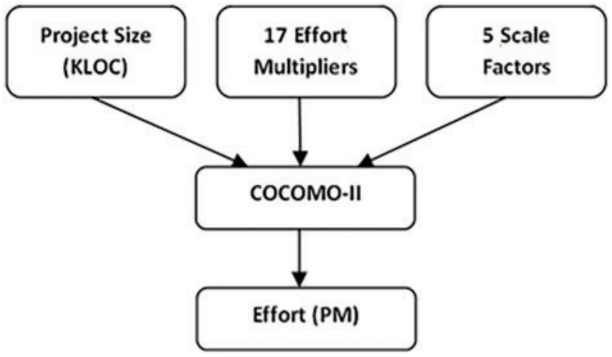

Figure 1: COCOMO-II model

COCOMO is an algorithmic approach employed for the estimation of project costs and effort. It calculates the development effort by considering the software's size, measured in KDSI (Thousands of Delivered Source Instructions), and software cost factors. This model's calibration was based on data gathered from approximately 63 projects undertaken by NASA. The estimated development effort is determined according to the project's development mode, categorized as Organic, Semi-detached, or Embedded. COCOMO Model establishes a nonlinear relationship between the project size and the estimated effort.

The formula for estimating effort (in Man Months, MM) is expressed as:

$$\text{Estimated effort (MM)} = a * (KDSI)^{b} \tag{1}$$

In the Eq. (1), "*a*" and "*b*" represent constants, the values of which depend on the project's development mode. KDSI denotes the size of the project in thousands of delivered source instructions. Table 2 provides the specific values of "*a*" and "*b*" for Organic, Semi-detached, and Embedded projects.

Table 2: COCOMO models

Model	a	b
Organic	3.2	1.05
Semi-detached	3	1.12
Embedded	2.8	1.2

The COCOMO Model recognized 15 cost drivers that have the potential to influence the estimated effort required for a software project. Each of these cost drivers was allocated specific weights that could be applied to the estimated effort, determined by the rating assigned to the cost driver (ranging from very low to extra high).

2.2 *COCOMO-II*

COCOMO-II, introduced by Barry Boehm in 2000, represents a model that incorporates more precise enhancements in certain cost drivers. It encompasses various software attributes, including 17 Effort Multipliers (EM), 5 Scale Factors (SF), Software Size (measured in Kilo Source Lines of Code, KSLOC), and the estimated effort, all of which are utilized in the COCOMO-II Architecture Post Model. The Effort Multipliers are organized into four categories, and there are 5 Factor Scales (SF).

The formula for estimating effort (in Person-Months, PM) within the COCOMO-II model is represented by Eq. (2):

$$\text{Estimated effort (PM)} = a * (SIZE)^{E} * \prod_{I=1}^{17} EM_i \tag{2}$$

Here, 'a' is a constant multiplier with a value of 2.94, which adjusts effort based on specific project conditions. '$SIZE$' denotes the estimated software size in Kilo Source Lines of Code (KSLOC), and 'E' represents the effort's scale expansion factor. E accounts for the exponential factor that considers the relative economies or diseconomies of scale when dealing with adjustments for the increasing size of software projects. 'EM_i' signifies the Effort Multiplier, where i can take values from 1 to 17.

To calculate the Scale Factor, the coefficient 'E' is determined using Eq. (3):

$$E = b + 0.01 * \sum_{j=1}^{5} SF_j \tag{3}$$

In this equation, 'b' is a constant exponential factor with a value of 0.91, and SF_j denotes a Scale Factor, where j can take values from 1 to 5.

3 Proposed Methodology

Software project cost estimation using both COCOMO-I and COCOMO-II is involved with numerous uncertainties. In both the COCOMO models, the multiplicative constants 'a' and 'b' require optimization to enhance estimation accuracy. This study aims to optimize these constants to elevate the performance of both COCOMO-I and COCOMO-II, employing the SGO method. To assess the effectiveness of SGO-COCOMO, a comprehensive evaluation is conducted by comparing it with other optimization methods such as PSO, GA, DE, BBO, FPA, Hybrid, and various variants of DE algorithms, as well as various cost estimation models including IVR, SEL, Bailey-Basil, Doty, and Halstead [54–58]. The experiments utilize datasets from COCOMO81 and Turkish Industry software projects as input, including project size in terms of KLOC, measured effort, 17 effort multipliers, and 5 scale factors. The output of this optimization process yields new, refined values for '*a*' and '*b*' for the COCOMO-I and COCOMO-II models, enhancing the accuracy of cost estimation.

3.1 Fitness Function

When it comes to estimating the costs of various projects, successful completion is indicated when the predicted effort closely aligns with the actual effort. Achieving higher accuracy entails aiming for lower values of MMRE (Mean Magnitude of Relative Error) and MD (Mean Deviation). As a result, our objective is to minimize MD and MMRE, thereby reducing the disparities between the actual and predicted efforts. The fitness functions employed in our experiments are defined in Eqs. (5) and (6).

The primary objective of this estimation method is to validate the precision of predictions. It is essential to minimize the gap between the predicted effort, denoted as $Estimate\_effort_i$, and the actual effort observed in real-world conditions, represented as $Actual\_effort_i$. A significant difference between $Actual\_effort_i$ and $Estimate\_effort_i$ can diminish prediction accuracy and have adverse implications for software system development efforts.

To evaluate the accuracy of estimated effort, this study employs the Magnitude of Relative Error (MRE) [59], a commonly used criterion in software cost estimation. MRE is calculated for each data point according to the formula defined in Eq. (4):

$$MRE_i = \frac{|Actual\_effort_i - Estimate\_effort_i|}{Actual\_effort_i} \times 100 \tag{4}$$

Eq. (5) illustrates how MMRE [59] is utilized to calculate the average value of the outcomes from each unique accuracy prediction value that was measured in the MRE criteria:

$$\text{MMRE} = \frac{1}{N}\sum_{i=1}^{N} MRE_i \tag{5}$$

- **Reason for Selection:** MMRE is widely recognized in the software engineering community due to its simplicity and effectiveness in providing a straightforward measure of estimation accuracy. It directly reflects the percentage of deviation from actual values, making it intuitive and easy to interpret.
- **Effectiveness:** MMRE is particularly effective in comparing the overall accuracy of different cost estimation models, as it aggregates the relative error across all projects, giving a clear picture of the model's general performance. However, it does have limitations, such as being sensitive to outliers and providing equal weight to overestimation and underestimation.

The Manhattan distance in Eq. (6) is used to determine the absolute gap between actual effort and estimated effort.

$$\text{MD} = \sum_{i=1}^{N} |Actual\_effort_i - Estimate\_effort_i| \tag{6}$$

- **Reason for Selection:** MD is chosen because it provides a measure of the total error across all projects without normalizing by the actual cost. This metric is particularly useful in understanding the cumulative error, which can be important when the scale of the projects varies significantly.

- **Effectiveness:** MD is effective in scenarios where the absolute error is critical, such as when budgeting for large-scale projects. It highlights the overall deviation, making it easier to assess how far off the estimates are from the actual costs in absolute terms.

3.2 SGO Algorithm

Satapathy et al. [10,55] proposed SGO in 2016. It is a stochastic-based computational algorithm known. The following is the explanation of this optimization algorithm. The SGO algorithm draws inspiration from human social dynamics to tackle complex problems. In this approach, a social group's members represent candidate solutions, each equipped with unique attributes related to their problem-solving abilities. These individual traits correspond to the various dimensions of the problem's design variables. The optimization process comprises two key phases: the Improving Phase and the Acquiring Phase.

Let's denote the individuals in the social group as P_i, with i ranging from 1 to pop_size. Here, pop_size represents the number of individuals in the group, and each person, P_i, is characterized by a set of traits $(p_{i1}, p_{i2}, p_{i3}, \ldots, p_{iD})$ that define their dimensions. Additionally, they each have a fitness value, f_i, where i ranges from 1 to pop_size.

3.2.1 Phase 1: Improving Phase

In the Improving Phase, the best-performing individual within the social group, referred to as 'gbest,' which actively shares knowledge with all other members, aiding in their knowledge enhancement. During this phase, each individual acquires knowledge from 'gbest.' The update for each individual is calculated as follows:

$$Pnew_i = c * P_i + r * (\text{gbest} - P_i) \tag{7}$$

Accept the new solution, $Pnew_i$, if it leads to an improvement in fitness compared to the current solution. Here, 'r' is a random number sampled from a uniform distribution U (0, 1), and 'c' is the self-introspection parameter, constrained within the range (0, 1), with a specified value of $c = 0.2$.

3.2.2 Phase 2: Acquiring Phase

In the Acquiring Phase, an individual within the social group interacts with the best individual ($best_P$) in the group and engages in random interactions with other group members to acquire knowledge. The Acquiring Phase is expressed as follows:

```
For i = 1: pop_size
    Randomly select one person $P_r$, where $i \neq r$
        If f ($P_i$) < f ($P_r$)
            $Pnew_i = P_i + r_1 * (P_i - P_r) + r_2 * (best_P - P_i)$
        Else
            $Pnew_i = P_i + r_1 * (P_r - P_i) + r_2 * (best_P - P_i)$        (8)
        End If
End for
```

Accept $Pnew_i$ if it results in improved fitness compared to the current solution.

Here, r_1, r_2 are two independently sampled random values from U (0, 1). These random sequences introduce stochasticity into the algorithm.

Pseudo code for SGO-COCOMO-II algorithm is given is given below:

Algorithm 1: Pseudo code of SGO-COCOMO-II algorithm

```
1. Define algorithm parameters: $c$, population size = pop_size
2. Objective Function $f(X) = MMRE$ or $MD$, where these are defined in terms of Eqs. (5) and (6)
3. Initialize/generate a random search agent (person) of populations
       $P = (P_1, P_2, \ldots, P_{pop\_size})$
4. Evaluate $f(P_i)$, $i = 1, 2, 3, \ldots,$ pop_size
5. Select the best solution as gbest
6. while $iter < Max\_iter$
7. for i = 1: pop_size
8. Update persons using Eq. (7)
9. end for
10. Select the best solution as $best_P$
11. for i = 1: pop_size
12. Update persons using Eq. (8)
13. end for
14. Update the current best solution
15. end while
```

4 Simulation and Experimental Results

4.1 Experiment 1

In this study, we employed the SGO algorithm for software cost estimation to predict the parameters of the COCOMO model. These estimated parameters will greatly enhance the accuracy of effort estimation for a wide range of projects, including organic, semidetached, and embedded ones. We applied Algorithm 1 to compute the COCOMO model parameters as outlined in Eq. (1), and for guiding the evolutionary process of the SGO algorithm, we utilized the parameter set provided in Table 3. To evaluate the effectiveness of our model, we conducted performance tests using a dataset comprising 63 projects (organic (25) projects, semidetached (11) projects, and embedded (27) projects) from the COCOMO81 software project dataset. You can find specific dataset details in Tables A1–A3.

Table 3: Parameters of software cost estimation-based SGO algorithm

Sl. no.	Parameter name	Description	Value
1	Pop	Population size	50
2	‘a’	Domain for a	−5 to 5
3	‘b’	Domain for b	−5 to 5
4	Max_iter	Maximum number of iterations	100
5	Dim	Dimension	2

(Continued)

Table 3 (continued)

Sl. no.	Parameter name	Description	Value
6	‘c’	Self-introspection parameter	0.2
7	S	Search space for given application	a, b

4.1.1 Evaluation Criteria

The evaluation criteria used to assess the accuracy of the software cost estimates generated by the proposed model are designed to align with the actual costs incurred during project development in real-world environments. Detailed information regarding the actual effort and KDSI (Thousands of Delivered Source Instructions) for all types of projects (organic, semidetached, and embedded) can be found in the A, B, C, D. Our evaluation process relies on the dataset from previously executed software projects. Specifically, we focus on the evaluation metric known as MMRE, which is defined in Eq. (5).

4.1.2 Experimental Results and Analysis

To assess the enhanced capabilities of the SGO-based model we developed, experiments were conducted using the COCOMO81 dataset. These experiments yielded parametric values of $a = 2.0764$ and $b = 1.0706$. We then compared this proposed model with existing COCOMO algorithms, including GA [6], PSO [7], Hybrid Algorithm (Hybrid Algo) [56], Differential Evolution-Based Model (DEBM) [57], DE [9], and Homeostasis Adaption Based Differential Evolution (HABDE) [58]. The discussion of results for the COCOMO81 software projects (including organic, semidetached, and embedded types) is presented below.

4.1.3 Comparison Effort for COCOMO Based Algorithms

Initially, we calculated the effort estimates obtained from Eq. (1) with the actual COCOMO81 dataset. The comparative analysis of effort levels, as presented in Tables 4–6, clearly demonstrates that our proposed model outperforms other soft computing models. In conclusion, we provide the following remarks:

- In Table 4, we can find the results of effort estimates produced by seven algorithms for organic projects. SGO consistently outperforms other algorithms such as GA, PSO, DE, DEBM, Hybrid Algo, and HABDE in the majority of organic projects, yielding superior effort values. Likewise, as illustrated in Table 5, SGO delivers superior effort estimates for a significant portion of semi-detached projects. Additionally, Table 6 reveals that SGO excels in generating effort estimates for most embedded projects. This indicates that the effort values (expressed in person-months) derived from SGO exhibit greater diversity and convergence rates in accordance with the proposed model.

Table 4: Comparison effort for COCOMO based algorithms on organic projects

Sl. no.	GA	PSO	DE	DEBM	Hybrid algo	HABDE	SGO
1	0.419895045	0.372687909	0.298150327	0.303098376	0.292245865	0.229036056	0.237566543891928
2	0.173631032	0.154110384	0.123288307	0.129350203	0.126701085	0.100082937	0.102137504907155
3	0.039510482	0.035068475	0.02805478	0.031032256	0.031210965	0.024982	0.024809813029841
4	0.008363829	0.007423517	0.005938813	0.006943662	0.007179986	0.005827258	0.005624112476726
5	0.065131451	0.057808981	0.046247184	0.050250336	0.050090621	0.039915193	0.040006312272578
6	0.021777155	0.019328836	0.015463069	0.017471969	0.050090621	0.039915193	0.014038533981783
7	0.036755316	0.032623061	0.026098449	0.028942919	0.029147193	0.023345171	0.023153452023160
8	0.173631032	0.154110384	0.123288307	0.129350203	0.126701085	0.100082937	0.102137504907155
9	0.036755316	0.032623061	0.026098449	0.028942919	0.029147193	0.023345171	0.023153452023160
10	0.013663962	0.012127777	0.009702221	0.011146712	0.0114255	0.009232355	0.008991339563468
11	0.006060016	0.005378712	0.00430297	0.005089261	0.005292837	0.004308028	0.004133269678976
12	0.0114627	0.010173994	0.010173994	0.009409828	0.009675478	0.007830525	0.007601497245152
13	0.127371045	0.113051223	0.090440978	0.095943614	0.094500061	0.074853687	0.075956108359601
14	0.075723016	0.067209778	0.053767822	0.058108422	0.057768089	0.045971171	0.046204025728732
15	0.08167816	0.072495408	0.057996326	0.062509051	0.062058993	0.049352451	0.049671571397799
16	0.094941077	0.084267228	0.067413783	0.072269866	0.071557002	0.056829348	0.057355382802727
17	0.216281588	0.191965907	0.153572725	0.159864654	0.155977602	0.122967468	0.126000000000000
18	0.062220649	0.055225428	0.044180343	0.048083039	0.04796936	0.038240453	0.038295521045999
19	0.023339844	0.020715838	0.01657267	0.018679436	0.018964439	0.015251102	0.015000000000000
20	0.0114627	0.010173994	0.008139195	0.009409828	0.009675478	0.007830525	0.007601497245152
21	0.009306041	0.008259799	0.006607839	0.007696489	0.00794328	0.006440603	0.006228296597521
22	0.065131451	0.057808981	0.046247184	0.050250336	0.050090621	0.039915193	0.040006312272578
23	0.059324368	0.052654765	0.042123812	0.045922955	0.045853397	0.0365692	0.036589756883968
24	0.014903962	0.013228368	0.010582695	0.012120612	0.012404501	0.010015665	0.009769802129187
25	0.023339844	0.020715838	0.01657267	0.018679436	0.018964439	0.015251102	0.015000000000000

Table 5: Comparison effort for COCOMO based algorithms on semi-detached projects

Sl. no.	GA	PSO	DE	DEBM	Hybrid algo	HABDE	SGO
1	1.28539338	1.137900396	1.251690436	1.065845111	0.986180343	0.758600264	0.748783979870568
2	5.784279909	5.262524067	5.788776473	5.499084096	4.560854191	3.508349378	3.621142318472880
3	0.295543542	0.254731191	0.28020431	0.214408413	0.220767032	0.169820794	0.160462927237439
4	0.041765661	0.03473994	0.038213934	0.025362086	0.030107948	0.02315996	0.020648291869038
5	0.00574033	0.004605491	0.00506604	0.002910379	0.003991426	0.003070327	0.002580683312884
6	0.232869459	0.199844042	0.219828447	0.16531908	0.17319817	0.133229362	0.124999930054934
7	0.041765661	0.03473994	0.038213934	0.025362086	0.030107948	0.02315996	0.020648291869038
8	0.078231615	0.065818456	0.072400302	0.050295179	0.057042662	0.043878971	0.039856334771454
9	2.131333716	1.9042004	2.09462044	1.850437962	1.650307013	1.269466933	1.272000000000003
10	0.025158046	0.020734045	0.022807449	0.014589136	0.017969505	0.013822696	0.012139513273215
11	0.097130475	0.082040729	0.090244802	0.063685797	0.071101965	0.054693819	0.050000000000000

Table 6: Comparison effort for COCOMO based algorithms on embedded projects

Sl. no.	GA	PSO	DE	DEBM	Hybrid algo	HABDE	SGO
1	0.78907428	0.762771804	0.73383908	0.32878095	0.39453714	0.289327236	0.264133812219125
2	0.027544993	0.026626826	0.025616843	0.01147708	0.013772496	0.010099831	0.019076007583622
3	0.110746745	0.107055187	0.102994473	0.046144477	0.055373373	0.04060714	0.056732308286247
4	0.160682752	0.15532666	0.149434959	0.066951147	0.080341376	0.058917009	0.075935136605108
5	0.15427706	0.149134491	0.143477665	0.064282108	0.07713853	0.056568255	0.073553508095873
6	0.17362159	0.167834204	0.161468079	0.072342329	0.086810795	0.075236022	0.080684270493033
7	0.206664473	0.199775657	0.19219796	0.086110197	0.103332236	0.089554605	0.092481632726686
8	0.010138396	0.00980045	0.009428709	0.004224332	0.005069198	0.004393305	0.008719166326261
9	0.013889969	0.01342697	0.012917671	0.005787487	0.006944985	0.006018987	0.011157786481149
10	0.02375853	0.022966579	0.022095432	0.009899387	0.011879265	0.010295363	0.016989519977945
11	0.012617891	0.012197294	0.011734638	0.005257454	0.006308945	0.005467753	0.010349104809746
12	2.751716763	2.659992871	2.559096589	1.146548651	1.375858381	1.192410597	0.702678544314391
13	1.841731577	1.780340525	1.712810367	0.767388157	0.920865789	0.798083683	0.513057148970856
14	2.065876733	1.997014175	1.921265362	0.860781972	1.032938367	0.826350693	0.561352370176723
15	0.831155273	0.803450097	0.772974404	0.346314697	0.415577636	0.332462109	0.275105165112695
16	0.600502427	0.580485679	0.558467257	0.250209344	0.300251213	0.240200971	0.213265891484713
17	0.213385091	0.206272254	0.198448134	0.088910454	0.106692545	0.085354036	0.094829197624214
18	0.28243157	0.273017184	0.26266136	0.117679821	0.141215785	0.112972628	0.118116071192240
19	0.006157749	0.005952491	0.005726707	0.002565729	0.003078875	0.0024631	0.005900000785723
20	3.489002526	3.372702442	3.24477235	1.453751053	1.744501263	1.395601011	0.846279223423543
21	0.240615114	0.23259461	0.223772056	0.100256298	0.120307557	0.096246046	0.104183376666385
22	0.116814609	0.112920788	0.108637586	0.048672754	0.058407304	0.046725843	0.059152986870379
23	0.608517996	0.588234063	0.565921736	0.253549165	0.304258998	0.263691132	0.215492519022750
24	0.02469633	0.023873119	0.022967587	0.010290137	0.012348165	0.010701743	0.017512604070445
25	0.141599028	0.13687906	0.131687096	0.058999595	0.070799514	0.061359579	0.068775255122874
26	0.081276026	0.078566825	0.075586704	0.033865011	0.040638013	0.035219611	0.044522416088239
27	0.03839489	0.03711506	0.035707248	0.015997871	0.019197445	0.016637786	0.024743685195371

4.1.4 Comparison MMRE for COCOMO Based Algorithms

For the project under consideration, another popular error computation technique called MMRE has been utilized, and the results are shown in Table 7. The MMRE value for other compared algorithms is imported from the paper [58]. From the table it is cleared with decreased MMRE for SGO algorithm as compared with other COCOMO based algorithms and it is clearly visible through Figs. 2–4.

Table 7: Comparison of MMRE for proposed algorithm on the COCOMO81 dataset

Dataset	SGO	GA	PSO	DE	DEBM	HYBRID	HABDE
MMRE (Organic)	0.5371	0.6721	0.6280	0.5704	0.5817	0.5777	0.5388
MMRE (Semi-detached)	0.5057	0.5462	0.5347	0.5428	0.5228	0.5241	0.5084
MMRE (Embedded)	0.6639	0.7630	0.7484	0.7484	0.7306	0.7211	0.7253

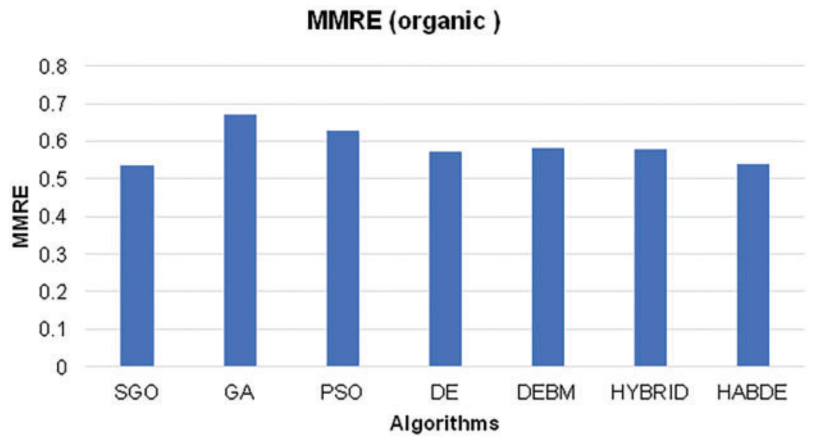

Figure 2: MMRE of proposed SGO, GA, PSO, DE, DEBM, HYBRID, HABDE using organic datasets

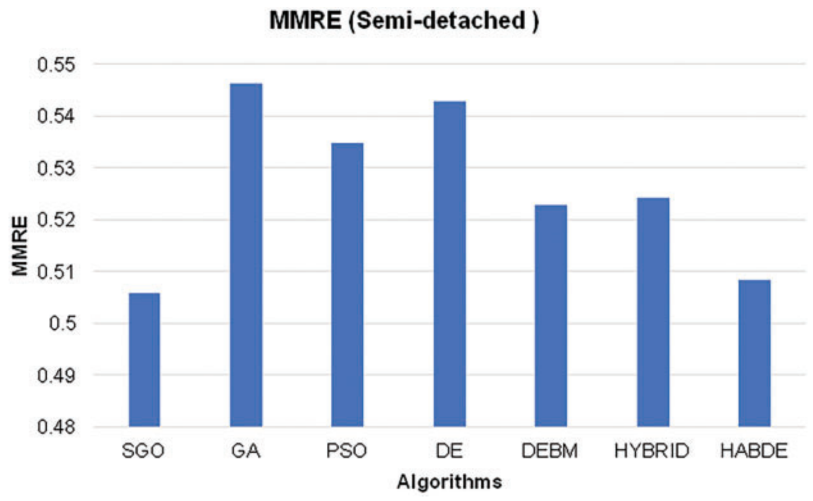

Figure 3: MMRE of proposed SGO, GA, PSO, DE, DEBM, HYBRID, HABDE using semi-detached datasets

4.1.5 Parameter of COCOMO Based SGO Algorithm

According to Eq. (1), the rate of convergence parameters (*a* and *b*) shows how quickly an estimating model approaches the intended value. The findings shown in Fig. 5 demonstrate that the proposed COCOMO-based SGO model has a higher rate of convergence since the SGO algorithm is capable of identifying the ideal value for parameters "*a*" and "*b*" after only 25 iterations.

4.1.6 Discussion

We can see that the SGO algorithm improves performance more than GA, PSO, DE, Hybrid Algorithm, DEBM, and HABDE. Once more, we can observe that the SGO method can calculate a parameter's value with less iterations. The proposed technique decreases the number of iterations and error rate. In general, existing methods' complexity rises since they generate less diversity.

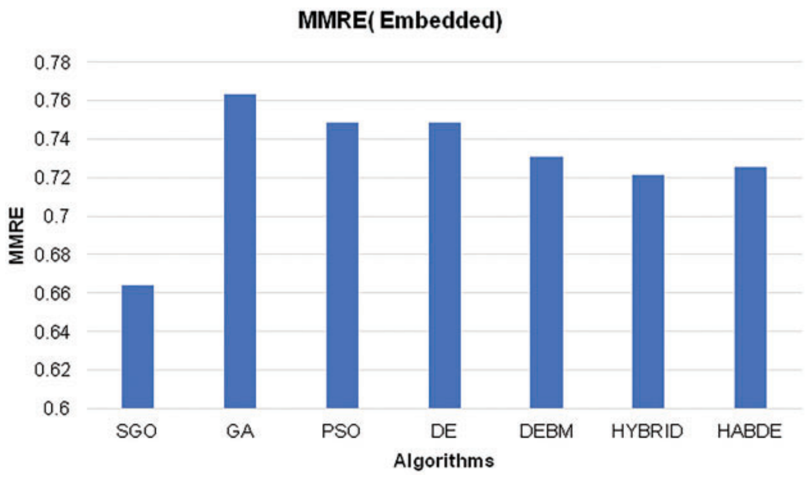

Figure 4: MMRE of proposed SGO, GA, PSO, DE, DEBM, HYBRID, HABDE using embedded datasets

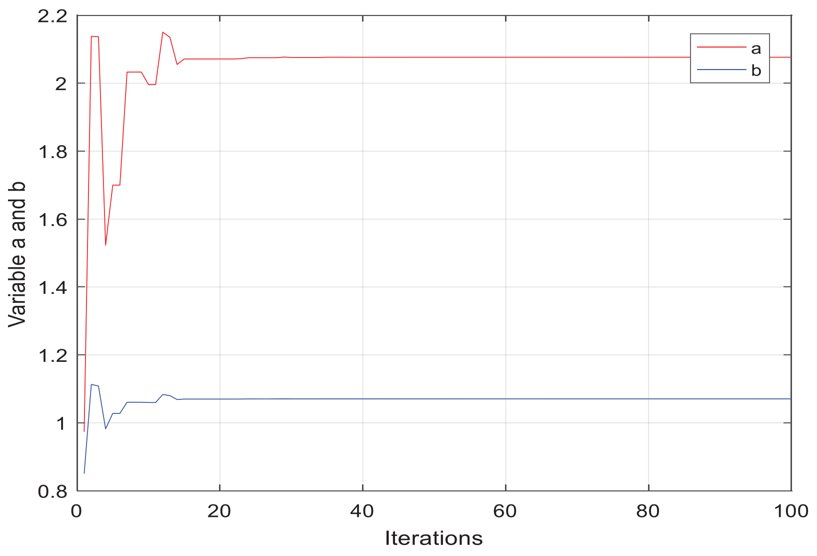

Figure 5: Convergence graph for parameter ‘*a*’ and ‘*b*’

4.2 *Experiment 2*

In this experiment, we employed the SGO algorithm for software cost estimation to fine-tune the parameters of the COCOMO-II model. Algorithm 1 was executed to estimate the COCOMO-II model parameters, as detailed in Eq. (2). We maintained consistency by utilizing the same set of parameters as in Experiment 1 to guide the evolutionary process of the SGO algorithm. To evaluate the performance of our developed model, we conducted tests using a latest dataset from the Turkish Software Industry. This dataset comprises information from five distinct software companies across various domains, encompassing data from a total of 12 projects. Each project is characterized by 25 attributes, including Project ID, 5 Scale Factors, 17 Effort Multipliers ranging from Very Low to Extra High, Measured Effort as the actual effort, and Project Size measured in KLOC. You can find specific dataset details in Table A4. All project data points were employed for calibration, and the results obtained from this calibration can serve as valuable insights for future projects within a similar category.

4.2.1 Evaluation Criteria

The evaluation criteria employed for assessing the accuracy of software cost estimates in the proposed model are designed to align with the actual costs incurred during real-world project development under live environmental conditions. For Experiment 2, two key evaluation criteria are used: MMRE (Mean Magnitude of Relative Error) and MD (Absolute Difference).

4.2.2 Experimental Results and Analysis

This section describes the experiment and the outcomes that were attained after applying the suggested method to the dataset. The primary goal of the optimization is to use the SGO approach to minimize uncertain parameters from a COCOMO-II model's coefficients ('*a*' and '*b*'), and then compare the new findings to those from the general Tabu Search [60] and PSO Method [2] coefficients.

The Fig. 6 illustrates the convergence process of SGO over multiple iterations with varying population sizes. Populations of 10, 20, 30, 40, and 50 were examined to evaluate the process's performance. Remarkably, each experiment involving SGO exhibited the same minimum error during convergence. After several iterations, we successfully obtained newly optimized coefficient values: $a = 4.3950$ and $b = -0.1834$, which differ from the original COCOMO-II values of $a = 2.94$ and $b = 0.91$. The results, calculated using these newly optimized coefficient values for effort, are presented in Table 8. Additionally, we imported results for COCOMO-II, Tabu Search, and PSO from paper [2] for comparison. The Fig. 7 displays the effort graph, demonstrating that SGO yields smoother and more accurate effort estimates when compared to efforts estimated using the basic COCOMO-II coefficients, Tabu Search, and the PSO Method.

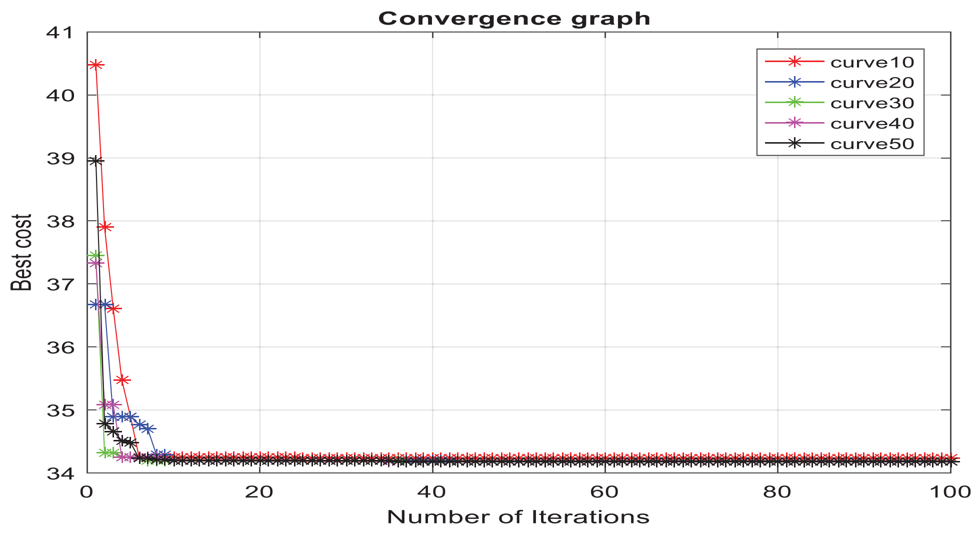

Figure 6: Best cost of MMRE with various population sizes, population size 10, 20, 30, 40, and 50

The results plotted in Fig. 8 which show that the proposed COCOMO based SGO model has better convergence rate as within 20 iteration SGO algorithm able to determine the best value for parameter '*a*' and '*b*'.

Table 8: Computed mean MRE (MMRE) and Manhattan distance (MD)

Model input	Model output	MMRE	MD
COCOMO-II model	Effort	733.1400	585.9266
Tabu Search	Effort	139.0699	90.5797
PSO	Effort	34.1939	43.2477
Proposed SGO	Effort	34.1922	43.2508

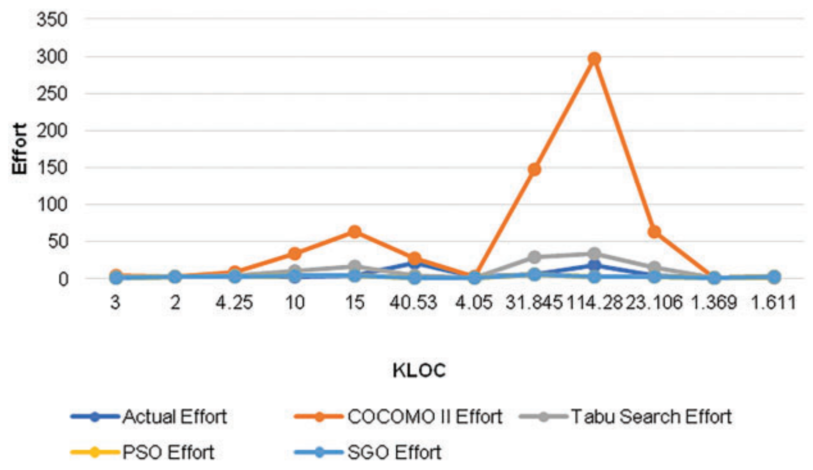

Figure 7: Effort graph for actual effort, COCOMO-II, Tabu, PSO and SGO

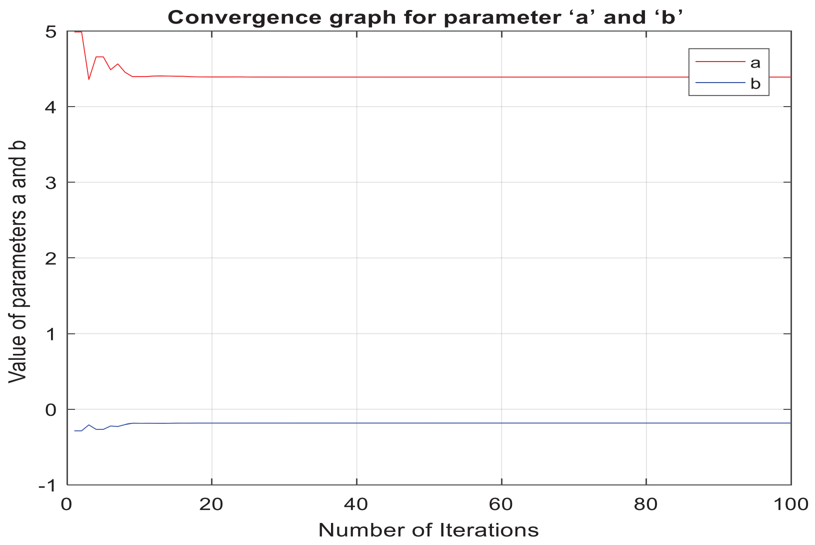

Figure 8: Convergence graph for parameters '*a*' and '*b*' in case of COCOMO-II based SGO model

The accuracy of the experiment is evaluated using fitness functions outlined in Eqs. (5) and (6). The experimental results are presented in Table 9, providing estimated effort values for all the compared algorithms. Notably, for Project Nos. 2, 8, and 11 with actual efforts of 2, 5, and 1, respectively, the proposed SGO method yielded highly accurate estimates of 2, 5, and 0.9789. This indicates a substantial reduction in error, with improvements of 0.00%, 0.00%, and 2.11%, respectively.

Table 9: Experimental result of estimated effort

Project no.	Size (KLOC)	Actual effort	COCOMO-II effort	Tabu Search effort	PSO effort	SGO effort
1	3.0000	1.2000	3.4881	1.9316	1.5679	1.5688
2	2.0000	2.0000	2.8568	1.8909	2.0001	2.0012
3	4.2500	4.5000	9.3041	4.4202	2.8581	2.8589
4	10.0000	3.0000	33.9773	11.0776	4.0969	4.0962
5	15.0000	4.0000	63.1555	17.2261	4.8891	4.8875
6	40.5300	22.0000	27.7316	4.8844	0.7244	0.7239
7	4.0500	2.0000	2.2887	1.1106	0.7411	0.7413
8	31.8450	5.0000	147.0897	28.8072	5.0012	4.9976
9	114.2800	18.0000	297.6050	33.2193	2.5041	2.5008
10	23.1060	4.0000	63.9962	14.4335	3.0896	3.0881
11	1.3690	1.0000	0.9239	0.7226	0.9789	0.9797
12	1.6110	2.1000	2.0424	1.4868	1.8112	1.8125

MMRE for each methodology signifies its overall accuracy. The MMRE values for the COCOMO-II model, Tabu Search, PSO, and SGO stand at 733.1400, 139.0699, 34.1939, and 34.1922, respectively. These values signify that SGO reduces errors significantly, by 698.9494% compared to the COCOMO-II model, 104.8793% compared to Tabu Search, and 0.0033% compared to PSO. Similarly, when examining MD (Absolute Difference), COCOMO-II exhibits an MD of 585.9266%, Tabu Search has an MD of 90.5797%, PSO has an MD of 43.2477%, and SGO has an MD of 43.2508%. These figures demonstrate that SGO can reduce errors by the value 542.68% compared to COCOMO-II, 47.3310% compared to Tabu Search, and 0.0011% compared to PSO, as presented in Table 7 and Fig. 6. In summary, the results for MMRE and MD indicate that the effort estimation provided by the proposed SGO method offers a significantly improved solution when compared to the COCOMO-II model, Tabu Search, and the PSO method, as illustrated in Fig. 9.

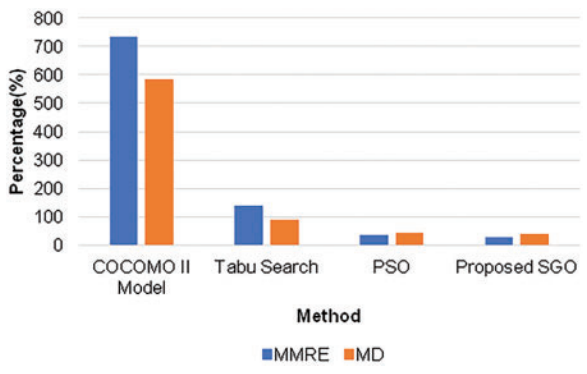

Figure 9: Comparison of MMRE and MD (in %) for COCOMO-II, Tabu Search, PSO, and SGO

4.2.3 Discussion

In this experiment, we investigated the effectiveness of employing the SGO method to optimize the parameters of the COCOMO-II model, aiming to enhance its accuracy. The SGO method was applied using the Turkish Software Industry dataset, and its performance was rigorously evaluated. The evaluation results clearly demonstrate the superiority of SGO, with a remarkable 698.9494% reduction in MMRE and a substantial 542.68% decrease in MD compared to the standard COCOMO-II model coefficient parameters. Furthermore, SGO outperforms Tabu Search with a 104.8793% lower MMRE and a 47.3310% lower MD, while it achieves an almost negligible 0.0033% MMRE and 0.0011% MD reduction compared to the PSO model. Overall, optimizing the parameters of the COCOMO-II model with the SGO method significantly enhances estimation accuracy when compared to the basic COCOMO-II model.

4.3 Experiment 3

This experiment is identical to Experiment 2, with the sole distinction being the comparison of the SGO algorithm's performance against BBO-COCOMO-II, PSO, GA, IVR, SEL, Bailey-Basil, Doty, Halstead and PPF (Past Present Future) algorithm [61,62].

4.3.1 Experimental Results and Analysis

The value of parameters '*a*' and '*b*' for our proposed SGO algorithm and some other methods is given in Table 10. The Table 11 presents the Magnitude of Relative Errors for estimates obtained using SGO, COCOMO-II, and other models applied to Turkish Industry software projects. Meanwhile, Table 12 provides a comparative analysis of MMRE (mean MMRE) and MD (Manhattan distance) for the proposed SGO, BBO, PSO, GA, PPF and several other models using datasets from the Turkish Industry. For our proposed SGO, experiment is carried out by using MATLAB, while for other models are reported from the paper [48].

Table 10: Value of parameters '*a*' and '*b*' for different model using Turkish Industry software projects

Models	'*a*'	'*b*'
Proposed SGO	4.3950	−0.1834
BBO-COCOMO-II	4.2826	−0.1757
GA	4.719	−0.858
PSO	4.2366	−0.1682
PPF	4.4625	−0.1885

Table 11: Magnitude of relative errors for estimations using SGO, PPF, BBO, COCOMO-II and others models using Turkish industry datasets

Project id	SGO	BBO	COCOMO-II	PSO	GA	FPA	SEL	Halstead	IVR	Doty	B-Basil	PPF
1	0.3073	0.2817	1.9070	0.2819	0.3310	0.1918	2.2409	2.0610	9.0228	12.9206	15.3924	0.3200
2	0.0006	0.0198	0.4282	0.0253	0.3269	0.0314	0.3337	0.0100	2.7725	4.4631	5.1451	0.0124
3	0.3647	0.3739	1.0676	0.3739	0.7429	0.3951	0.1949	0.3629	2.9895	4.3456	5.5475	0.3597
4	0.3654	0.3543	10.325	0.3630	0.6899	0.489	2.9719	6.3786	15.0090	18.6413	25.4997	0.3702
5	0.2219	0.2157	14.7888	0.2273	0.7889	0.2187	3.3434	9.1666	18.1392	21.5216	30.8103	0.2236

(Continued)

Table 11 (continued)

Project id	SGO	BBO	COCOMO-II	PSO	GA	FPA	SEL	Halstead	IVR	Doty	B-Basil	PPF
6	0.9671	0.967	0.26069	0.9664	0.9971	0.881	0.9904	7.2099	9.9144	10.593	17.3215	0.9672
7	0.6294	0.6349	0.14142	0.6350	0.8451	0.634	1.5705	1.8527	7.4923	10.4385	12.9309	0.6263
8	0.0005	0.0003	28.4174	0.0155	0.8960	0.0133	5.9982	24.158	35.3920	38.6284	59.9426	0.0029
9	0.8611	0.8596	15.5334	0.8561	0.9939	0.887	5.3792	46.5090	42.9412	40.9482	73.5306	0.8623
10	0.2280	0.2293	14.999	0.2194	0.9003	0.2095	5.4913	18.4368	30.4560	34.4040	51.5077	0.2286
11	0.0203	0.0430	0.0761	0.0511	0.1489	0.0522	0.849	0.1213	3.8791	6.3469	6.9175	0.0069
12	0.1369	0.1559	0.0275	0.1619	0.3282	0.1667	0.0387	0.3184	1.8017	3.1486	3.5538	0.1258

Table 12: MMRE (mean MMRE) and MD (Manhattan distance) comparison for proposed SGO, PSO, GA, PPF and other various models using Turkish industry datasets

Model	MMRE	MD
SGO	34.1922	43.2508
PPF	34.2148	43.2846
COCOMO-II	733.13	585.9424
PSO	34.800	43.3571
GA	66.57	60.0558
SEL	245.23	201.1912
Halstead	971.30	1254.72
IVR	1498.41	1459.898
Doty	1719.98	1520.708
Baily Basil	256.749	2504.08
FPA	34.54	43.3417
BBO-COCOMO-II	34.47	43.2952

Above Figs. 10 and 11 depict the comparisons of MMRE and MD between the proposed SGO method, PPF, BBO, and various other cost estimation techniques applied to Turkish Industry software projects, respectively.

4.3.2 Discussion

In this experiment, the SGO-based algorithm was employed to optimize the existing parameters of COCOMO-II. The proposed SGO method was rigorously assessed using datasets from the Turkish Industry's software projects. The simulation results clearly demonstrate the superior performance of the proposed SGO-based approach compared to conventional COCOMO-II, as well as other methods such as BBO, PSO, GA, PPF, and FPA, along with various other cost estimation techniques.

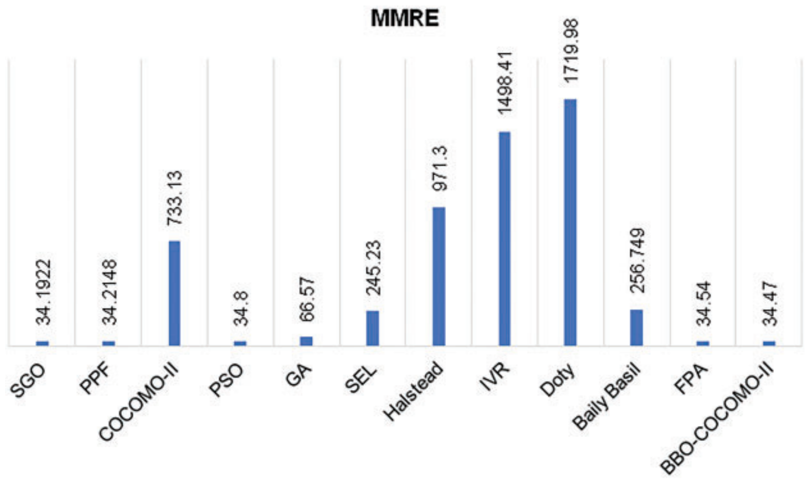

Figure 10: MMRE for proposed SGO, PPF, BBO, PSO, GA, COCOMO-II and others models using Turkish industry datasets

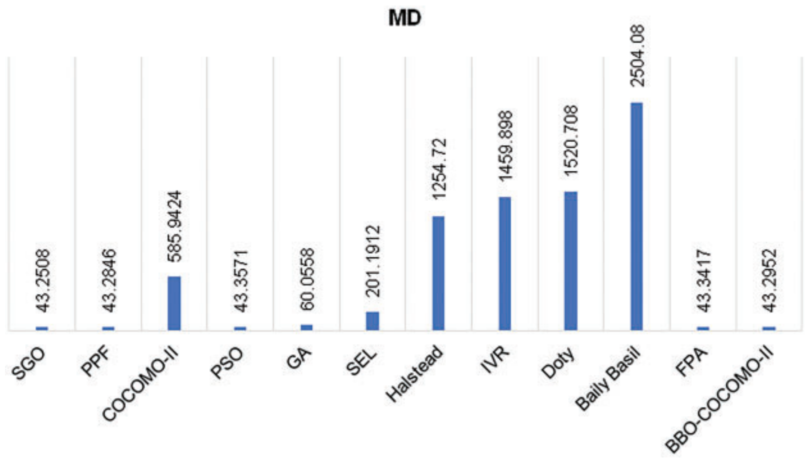

Figure 11: MD for proposed SGO, PPF, BBO, PSO, GA, COCOMO-II and others models using Turkish industry datasets

4.4 *Scalability, Applicability, Limitations and Challenges of SGO-Based Approach*

4.4.1 Scalability and Applicability

While the SGO-based approach has demonstrated significant improvements in software cost estimation accuracy, it is essential to consider its scalability across different types and sizes of software projects. Our experiments primarily focused on datasets like COCOMO81 and Turkish industry projects, which provided a strong foundation for validating the method. However, further investigation is needed to assess how well the SGO algorithm adapts to larger, more complex projects or those in diverse industry domains.

The adaptability of SGO to various project sizes is promising, but it may require adjustments or fine-tuning when applied to very large-scale or small-scale projects. The computational overhead associated with SGO could be a concern in scenarios where projects involve massive datasets or require real-time estimation. This is an area where balancing accuracy and computational efficiency will be crucial.

4.4.2 Limitations and Challenges

Generalizing the SGO-based approach presents several challenges. One potential limitation is the dependency on the quality and quantity of historical data used for training. In environments where historical data is scarce or of poor quality, the effectiveness of the SGO algorithm might be compromised. Additionally, while SGO has shown superiority over other optimization techniques, its performance may vary depending on the specific characteristics of the software project, such as the development methodology, team structure, and project complexity.

Furthermore, the need for fine-tuning the SGO parameters for different types of projects could limit its immediate applicability, requiring domain expertise and iterative experimentation to achieve optimal results. These challenges highlight the importance of ongoing research to refine and adapt the SGO-based method for broader use cases.

5 Conclusion

In this study, we introduced the Social Group Optimization (SGO) algorithm to optimize the parameters of the COCOMO and COCOMO-II models. Testing on COCOMO81 and Turkish Industry software projects showed that SGO outperformed conventional approaches, including COCOMO-II and other optimization algorithms like PSO, GA, BBO, and PPF. The results highlight SGO's effectiveness in improving software cost estimation accuracy.

Looking ahead, we plan to apply evolutionary algorithms to optimize not only COCOMO coefficients but also those of the Constructive Quality Estimation Model (CQEM). This will lead to a more comprehensive framework for both cost and quality estimation, enhancing project planning and resource allocation. Further, we explore the integration of SGO with other optimization techniques, such as hybrid algorithms or machine learning models, to further enhance its performance and adaptability in complex environments.

In conclusion, the success of SGO in this study underscores its potential as a valuable tool for software cost estimation, paving the way for more accurate and cost-effective software development.

Acknowledgement: The authors thank all the faculty and co-scholars of KIIT Deemed to be University, Bhubaneswar, India. Their goal is to provide a comfortable place to work. They have given us permission to utilize any and all resources at our service to fulfill our contribution's objective.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: Sagiraju Srinadhraju: Study conception and design, Data collection, Analysis and interpretation of results, Draft manuscript preparation; Samaresh Mishra, Sagiraju Srinadhraju: Data collection, Analysis and interpretation of results; Suresh Chandra Satapathy: Analysis and interpretation of results, Draft manuscript preparation. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Data available on request from the authors. The data that support the findings of this study are available from the corresponding author, Sagiraju Srinadhraju, upon reasonable request. Datasets used in this research are taken from the below URLs:

- http://promise.site.uottawa.ca/SERepository/datasets-page.html (accessed on 01 May 2024)
- https://openscience.us/repo/effort/cocomo/cocomosdr.html (accessed on 01 May 2024)

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] D. Nandal and O. Sangwan, "Software cost estimation by optimizing COCOMO model using hybrid BATGSA algorithm," *Int. J. Intell. Eng. Syst.*, vol. 11, no. 4, pp. 250–263, Aug. 2018. doi: 10.22266/ijies2018.0831.25.

[2] K. Langsari, R. Sarno, and S. Sholiq, "Optimizing effort parameter of COCOMO-II using particle swarm optimization method," *Telkomnika (Telecommun. Comput. Electron. Control)*, vol. 16, no. 5, pp. 2208–2216, Oct. 2018. doi: 10.12928/telkomnika.v16i5.9703.

[3] A. Ullah, B. Wang, J. Sheng, J. Long, M. Asim and F. Riaz, "A novel technique of software cost estimation using flower pollination algorithm," *in 2019 Int. Conf. Intell. Comput. Automat. Syst. (ICICAS)*, Chongqing, China, 2019, pp. 654–658. doi: 10.1109/icicas48597.2019.00142.

[4] A. Trendowicz and R. Jeffery, "Constructive cost model–COCOMO," in *Software Project Effort Estimation: Foundations and Best Practice Guidelines for Success*. India: Springer Nature, 2014, pp. 277–293.

[5] A. Salam, A. Khan, and S. Baseer, "A comparative study for software cost estimation using COCOMO-II & Walston-Felix models," in *1st Int. Conf. Innov. Comput. Sci. Softw. Eng. (ICONICS 2016)*, 2016, pp. 15–16.

[6] F. S. Gharehchopogh and A. Pourali, "A new approach based on continuous genetic algorithm in software cost estimation," *J. Sci. Res. Dev.*, vol. 2, no. 4, pp. 87–94, 2015.

[7] F. S. Gharehchopogh, I. Maleki, and S. R. Khaze, "A novel particle swarm optimization approach for software effort estimation," *Int. J. Acad. Res.*, vol. 6, no. 2, pp. 69–76, Mar. 2014. doi: 10.7813/2075-4124.2014/6-2/a.12.

[8] N. Ghatasheh, H. Faris, I. Aljarah, and R. M. H. Al-Sayyed, "Optimizing software effort estimation models using firefly algorithm," *J. Softw. Eng. Appl.*, vol. 8, no. 3, pp. 133–142, Mar. 2015. doi: 10.4236/jsea.2015.83014.

[9] P. Singal, A. C. Kumari, and P. Sharma, "Estimation of software development effort: A differential evolution approach," *Procedia Comput. Sci.*, vol. 167, pp. 2643–2652, Jan. 2020. doi: 10.1016/j.procs.2020.03.343.

[10] S. Satapathy and A. Naik, "Social Group Optimization (SGO): A new population evolutionary optimization technique," *Complex Intell. Syst.*, vol. 2, pp. 173–203, 2016. doi: 10.1007/s40747-016-0022-8.

[11] S. Das, P. Saha, S. C. Satapathy, and J. J. Jena, "Social group optimization algorithm for civil engineering structural health monitoring," *Eng. Optim.*, vol. 53, no. 10, pp. 1651–1670, 2021. doi: 10.1080/0305215X.2020.1808974.

[12] A. Naik, S. C. Satapathy, A. S. Ashour, and N. Dey, "Social group optimization for global optimization of multimodal functions & data clustering problems," *Neural Comput. Appl.*, vol. 30, no. 1, pp. 271–287, 2018. doi: 10.1007/s00521-016-2686-9.

[13] S. Rani and B. Suri, "Adopting social group optimization algorithm using mutation testing for test suite generation: SGO-MT," in *Computat. Sci. Apps.– ICCSA 2019*, Springer, Cham, 2019, pp. 520–528.

[14] S. Verma, J. J. Jena, S. C. Satapathy, and M. Rout, "Solving travelling salesman problem using discreet social group optimization," *J. Sci. Ind. Res.*, vol. 79, no. 10, pp. 928–930, 2020.

[15] A. Naik and P. K. Chokkalingam, "Binary social group optimization algorithm for solving 0–1 Knapsack problem," *Decis. Sci. Lett.*, vol. 11, no. 1, pp. 55–72, 2022. doi: 10.5267/j.dsl.2021.8.004.

[16] A. Naik, S. C. Satapathy, and A. Abraham, "Modified social group optimization: A meta-heuristic algorithm to solve short-term hydrothermal scheduling," *Appl. Soft Comput.*, vol. 95, 2020, Art. no. 106524. doi: 10.1016/j.asoc.2020.106524.

[17] R. Monisha, R. Mrinalini, M. N. Britto, R. Ramakrishnan, and V. Rajinikanth, "Social group optimization & Shannon's function-based RGB image multi-level thresholding," *Smart Intell. Comput. Appl.*, vol. 105, pp. 123–132, 2019. doi: 10.1007/978-981-13-1927-3.

[18] A. Reddy and K. V. L. Narayana, "Investigation of a multi-strategy ensemble social group optimization algorithm for the optimization of energy management in electric vehicles," *IEEE Access*, vol. 10, pp. 12084–12124, 2022. doi: 10.1109/ACCESS.2022.3144065.

[19] K. S. Manic, N. Al Shibli, and R. Al Sulaimi, "SGO & Tsallis entropy assisted segmentation of abnormal regions from brain MRI," *J. Eng. Sci. Tech.*, vol. 13, pp. 52–62, 2018.

[20] P. Parwekar, "SGO: A new approach for energy efficient clustering in WSN," *Int. J. Nat. Comput. Res.*, vol. 7, no. 3, pp. 54–72, 2018. doi: 10.4018/IJNCR.

[21] N. Dey, V. Rajinikanth, S. J. Fong, M. S. Kaiser, and M. Mahmud, "Social group optimization-assisted Kapur's entropy & morphological segmentation for automated detection of COVID-19 infection from computed tomography images," *Cogn. Comput.*, vol. 12, pp. 1011–1023, 2020. doi: 10.1007/s12559-020-09751-3.

[22] A. K. Singh, A. Kumar, M. Mahmud, M. S. Kaiser, and A. Kishore, "COVID-19 infection detection from chest X-ray images using hybrid social group optimization & support vector classifier," *Cogn. Comput.*, vol. 16, pp. 1765–1777, 2024. doi: 10.1007/s12559-021-09848-3.

[23] N. Dey, V. Rajinikanth, A. Ashour, and J. M. Tavares, "Social group optimization supported segmentation & evaluation of skin melanoma images," *Symmetry*, vol. 10, no. 2, 2018, Art. no. 51. doi: 10.3390/sym10020051.

[24] M. A. M. Akhtar, A. K. Manna, and A. K. Bhunia, "Optimization of a non-instantaneous deteriorating inventory problem with time & price dependent demand over finite time horizon via hybrid DESGO algorithm," *Exp. Sys. Apps.*, vol. 211, 2023, Art. no. 118676. doi: 10.1016/j.eswa.2022.118676.

[25] V. K. R. A. Kalananda and V. L. N. Komanapalli, "A combinatorial social group whale optimization algorithm for numerical & engineering optimization problems," *Appl. Soft Comput.*, vol. 99, 2021, Art. no. 106903. doi: 10.1016/j.asoc.2020.106903.

[26] V. K. R. A. Kalananda and V. L. N. Komanapalli, "Investigation of a social group assisted differential evolution for the optimal PV parameter extraction of standard & modified diode models," *Energy Conv. Manage.*, vol. 268, 2022, Art. no. 115955. doi: 10.1016/j.enconman.2022.115955.

[27] S. P. Praveen, K. T. Rao, and B. Janakiramaiah, "Effective allocation of resources & task scheduling in cloud environment using social group optimization," *Arab. J. Sci. Eng.*, vol. 43, pp. 4265–4272, 2018. doi: 10.1007/s13369-017-2926-z.

[28] H. Kraiem *et al.*, "Parameters identification of photovoltaic cell & module models using modified social group optimization algorithm," *Sustainability*, vol. 15, no. 13, 2023, Art. no. 10510. doi: 10.3390/su151310510.

[29] D. C. Secui, C. Hora, C. Bendea, M. L. Secui, G. Bendea and F. C. Dan, "Modified social group optimization to solve the problem of economic emission dispatch with the incorporation of wind power," *Sustainability*, vol. 16, no. 1, 2024, Art. no. 397. doi: 10.3390/su16010397.

[30] D. H. Tran, "Optimizing time-cost in generalized construction projects using multiple-objective social group optimization & multi-criteria decision-making methods," *Eng. Const. Arch. Manage.*, vol. 27, no. 9, pp. 2287–2313, 2020. doi: 10.1108/ECAM-08-2019-0412.

[31] K. Ullah, A. Basit, Z. Ullah, F. R. Albogamy, and G. Hafeez, "A transformer fault diagnosis method based on a hybrid improved social group optimization algorithm & elman neural network," *Energies*, vol. 15, no. 5, 2022, Art. no. 1771. doi: 10.3390/en15051771.

[32] V. -H. Huynh, T. -H. Nguyen, H. C. Pham, T. -M. -D. Huynh, T. -C. Nguyen and D. -H. Tran, "Multiple objective social group optimization for time-cost–quality-carbon dioxide in generalized construction projects," *Int. J. Civil Eng.*, vol. 19, pp. 805–822, 2021. doi: 10.1007/s40999-020-00581-w.

[33] A. Naik, "Marine predators social group optimization: A hybrid approach," *Evol. Intell.*, vol. 17, pp. 2355–2386, 2024. doi: 10.1007/s12065-023-00891-7.

[34] S. Vadivel, B. C. Sengodan, S. Ramasamy, M. Ahsan, J. Haider and E. M. G. Rodrigues, "Social grouping algorithm aided maximum power point tracking scheme for partial shaded photovoltaic array," *Energies*, vol. 15, no. 6, 2022, Art. no. 2105. doi: 10.3390/en15062105.

[35] C. Wang *et al.*, "Dual-population social group optimization algorithm based on human social group behavior law," *IEEE Trans. Comput. Soc. Syst.*, vol. 10, no. 1, pp. 166–177, Feb. 2023. doi: 10.1109/TCSS.2022.3141114.

[36] A. Naik, "Multi-objective social group optimization for machining process," *Evol. Intell.*, vol. 17, pp. 1655–1676, 2024. doi: 10.1007/s12065-023-00856-w.

[37] M. T. Tran, H. A. Pham, V. L. Nguyen, and A. T. Trinh, "Optimisation of stiffeners for maximum fundamental frequency of cross-ply laminated cylindrical panels using social group optimisation & smeared stiffener method," *Thin-Walled Struct.*, vol. 120, pp. 172–179, 2017. doi: 10.1016/j.tws.2017.08.033.

[38] P. Garg and R. K. Rama Kishore, "A robust & secured adaptive image watermarking using social group optimization," *Vis. Comput.*, vol. 39, pp. 4839–4854, 2023. doi: 10.1007/s00371-022-02631-x.

[39] S. M. A. Rahaman, M. Azharuddin, and P. Kuila, "Efficient scheduling of charger-UAV in wireless rechargeable sensor networks: Social group optimization based approach," *J. Netw. Syst. Manage.*, vol. 32, 2024, Art. no. 55. doi: 10.1007/s10922-024-09833-9.

[40] W. Zhang, Y. Yang, and Q. Wang, "A study on software effort prediction using machine learning techniques," *Comm. Comp. Info. Sci.*, vol. 275, pp. 1–15, 2013. doi: 10.1007/978-3-642-32341-6.

[41] M. Grover, P. K. Bhatia, and H. Mittal, "Estimating software test effort based on revised UCP model using fuzzy technique," *Smart. Innov. Syst. Tech.*, vol. 83, pp. 490–498, 2017. doi: 10.1007/978-3-319-63673-3_59.

[42] S. Kumari and S. Pushkar, "Software cost estimation using cuckoo search," *Adv. Intell. Syst. Comput.*, pp. 167–175, Nov. 2016. doi: 10.1007/978-981-10-2525-9_17.

[43] S. Kumari and S. Pushkar, "Cuckoo search based hybrid models for improving the accuracy of software effort estimation," *Microsys. Tech.*, vol. 24, no. 12, pp. 4767–4774, Apr. 2018. doi: 10.1007/s00542-018-3871-9.

[44] F. S. Gharehchopogh, R. Rezaii, and B. Arasteh, "A new approach by using Tabu search and genetic algorithms in software cost estimation," in *9th Int. Conf. App. Inf. Commun. Tech.*, 2015, pp. 113–117. doi: 10.1109/ICAict.2015.7338528.

[45] T. Urbanek, Z. Prokopova, R. Silhavy, and A. Kuncar, "Using analytical programming for software effort estimation," *Adv. Intell. Syst. Comput.*, pp. 261–272, 2016. doi: 10.1007/978-3-319-33622-0_24.

[46] S. Dalal, N. Dahiya, and V. Jaglan, "Efficient tuning of COCOMO model cost drivers through Generalized Reduced Gradient (GRG) nonlinear optimization with best-fit analysis," in *Adv. Comput. Intell. Eng.*, Springer, 2018, pp. 347–354.

[47] L. P. D. Santos and M. G. V. Ferreira, "Safety critical software effort estimation using COCOMO II: A case study in aeronautical industry," *Revista IEEE Am. Lat.*, vol. 16, no. 7, pp. 2069–2078, Jul. 2018. doi: 10.1109/tla.2018.8447378.

[48] R. T. Hughes, "Expert judgement as an estimating method," *Inf. Softw. Tech.*, vol. 38, no. 2, pp. 67–75, Jan. 1996. doi: 10.1016/0950-5849(95)01045-9.

[49] Y. A. Effendi, R. Sarno, and J. Prasetyo, "Implementation of bat algorithm for COCOMO II optimization," *Int. Sem. App. Tech. Inf. Commun.*, pp. 441–446, 2018. doi: 10.1109/isemantic.2018.8549699.

[50] S. W. Ahmad and G. R. Bamnote, "Whale-Crow Optimization (WCO)-based optimal regression model for software cost estimation," *Sadhana*, vol. 44, no. 4, Mar. 2019, Art. no. 94. doi: 10.1007/s12046-019-1085-1.

[51] A. Sheta, D. Rine, and A. Ayesh, "Development of software effort and schedule estimation models using soft computing techniques," in *2008 IEEE Cong. Evolutiona. Computati. (IEEE World Cong. Computation. Intelligen.)*, 2008, vol. 43, pp. 1283–1289. doi: 10.1109/cec.2008.4630961.

[52] B. W. Boehm, *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice-Hall, 1981.

[53] 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. SE*, vol. 1, no. 1, pp. 57–94, Dec. 1995. doi: 10.1007/bf02249046.

[54] Srivastava and D. Devesh, "VRS model: A model for estimation of efforts and time duration in development of IVR software system," *Int. J. SE*, vol. 5, no. 1, pp. 27–46, 2012.

[55] A. Naik, "Chaotic social group optimization for structural engineering design problems," *J. Bionic. Eng.*, vol. 20, no. 4, pp. 1852–1877, Feb. 2023. doi: 10.1007/s42235-023-00340-2.
[56] M. Hasanluo and F. S. Gharehchopogh, "Software cost estimation by a new hybrid model of particle swarm optimization and K-nearest neighbor algorithms," *J. Electr. Comput. Eng. Innov.*, vol. 4, no. 1, pp. 49–55, Jan. 2016. doi: 10.22061/jecei.2016.556.
[57] A. K. Bardsiri and S. M. Hashemi, "A differential evolution-based model to estimate the software services development effort," *J. Softw.*, vol. 28, no. 1, pp. 57–77, Dec. 2015. doi: 10.1002/smr.1765.
[58] S. P. Singh, V. P. Singh, and A. K. Mehta, "Differential evolution using homeostasis adaption based mutation operator & its application for software cost estimation," *J. King Saudi Univ.-Com. Info. Sci.*, vol. 33, no. 6, pp. 740–752, Jul. 2021. doi: 10.1016/j.jksuci.2018.05.009.
[59] K. Dejaeger, W. Verbeke, D. Martens, and B. Baesens, "Data mining techniques for software effort estimation: A comparative study," *IEEE Trans. SE*, vol. 38, no. 2, pp. 375–397, Mar. 2012. doi: 10.1109/TSE.2011.55.
[60] R. Chadha and S. Nagpal, "Optimization of COCOMOII model coefficients using Tabu search," *Int. J. CS IT*, vol. 3, no. 8, pp. 4463–4465, 2014.
[61] A. Ullah, B. Wang, J. Sheng, J. Long, M. Asim and Z. Sun, "Optimization of software cost estimation model based on biogeography-based optimization algorithm," *Int. Decis. Tech.*, vol. 14, no. 4, pp. 441–448, Jan. 2021. doi: 10.3233/IDT-200103.
[62] S. Srinadhraju, S. Mishra, and S. C. Satapathy, "Optimization of software cost estimation model based on present past future algorithm," in *8th Int. Conf. Inf. Syst. Dgn. Int. App.*, Dec. 2024, vol. 3.

Appendix A

Table A1: Dataset of COCOMO based algorithms on organic projects

Sl. no.	Actual _Effort	KLOC
1	0.243	0.132
2	0.24	0.06
3	0.033	0.016
4	0.043	0.004
5	0.079	0.025
6	0.088	0.0094
7	0.055	0.015
8	0.047	0.06
9	0.012	0.015
10	0.008	0.0062
11	0.008	0.003
12	0.006	0.0053
13	0.045	0.0455
14	0.083	0.0286
15	0.087	0.0306
16	0.106	0.035
17	0.126	0.073
18	0.176	0.024
19	0.122	0.01
20	0.014	0.0053

(Continued)

Table A1 (continued)

Sl. no.	Actual _Effort	KLOC
21	0.02	0.0044
22	0.13	0.025
23	0.07	0.023
24	0.057	0.0067
25	0.015	0.01

Table A2: Dataset of COCOMO based algorithms on semi-detached projects

Sl. no.	Actual _Effort	KLOC
1	1.6	0.293
2	6.6	1.15
3	0.539	0.077
4	0.098	0.013
5	0.0073	0.00214
6	1.063	0.062
7	0.082	0.013
8	0.036	0.023
9	1.272	0.464
10	0.041	0.0082
11	0.05	0.028

Table A3: Dataset of COCOMO based algorithms on embedded projects

Sl. no.	Actual _Effort	KLOC
1	2.04	0.113
2	0.008	0.0069
3	1.075	0.022
4	0.423	0.03
5	0.321	0.029
6	0.218	0.032
7	0.201	0.037
8	0.06	0.003
9	0.061	0.0039
10	0.04	0.0061
11	0.009	0.0036
12	11.4	0.32
13	6.4	0.229
14	2.455	0.252

(Continued)

Table A3 (continued)

Sl. no.	Actual _Effort	KLOC
15	0.724	0.118
16	0.453	0.09
17	0.523	0.038
18	0.387	0.048
19	0.0059	0.00198
20	0.702	0.39
21	0.605	0.042
22	0.23	0.023
23	0.156	0.091
24	0.018	0.0063
25	0.958	0.027
26	0.237	0.017
27	0.038	0.0091

Table A4: Dataset from Turkish software industry

Project no.	Size (KLOC)	Actual effort	Effort multiplier	Scale factor
1	3.0000	1.2000	0.3508	19.9200
2	2.0000	2.0000	0.4538	18.8300
3	4.2500	4.5000	0.6473	18.6800
4	10.0000	3.0000	1.1213	10.3100
5	15.0000	4.0000	1.0841	19.2800
6	40.5300	22.0000	0.2379	8.4100
7	4.0500	2.0000	0.1965	7.4200
8	31.8450	5.0000	1.0837	19.7300
9	114.2800	18.0000	0.3734	27.2300
10	23.1060	4.0000	0.6500	20.8200
11	1.3690	1.0000	0.2250	15.3600
12	1.6110	2.1000	0.4109	19.1100