AUT Journal of Modeling and Simulation

AUT J. Model. Simul., 56(2) (2024) 155-170
DOI: 10.22060/miscj.2024.23316.5366

Improving Software Effort Estimation through a Hybrid Approach of Metaheuristic Algorithms in Analogy-based Method

Ehsan Nasr, Keyvan Mohebbi *

Faculty of Engineering, Isfahan (Khorasgan) Branch, Islamic Azad University, Isfahan, Iran

ABSTRACT: Project management in software development is one of the most crucial activities as it encompasses the entire software development process from start to finish. Estimating the effort required for software projects is a significant challenge in project management. Managing software projects and consequently estimating their effort for more efficient and impactful management of such projects is necessary and unavoidable. Analogy-based estimation in software effort estimation involves comparing new projects to completed ones. However, this method can be ineffective due to variations in feature importance and dependencies. To address this, weights are assigned to features using optimization techniques like meta-heuristic algorithms. Yet, these algorithms may get stuck in local optima, yielding nonoptimal results. An approach to estimate software effort is proposed in this study. It aims to find global optimal feature weights by combining particle swarm and genetics metaheuristic algorithms. This hybrid approach leverages particle motion and composition to enhance solution generation, increasing the likelihood of finding the global optimum and overcoming local optima issues. The algorithm calculates feature weights for project estimation using analogy-based methods. The proposed approach was tested and assessed using two datasets, namely Maxwell and Desharnais. The experimental results indicated an enhancement in the evaluation criteria, including MMRE, MdMRE, and PRED, compared to similar research works.

Review History:

Received: Jul. 01, 2024
Revised: Sep. 16, 2024
Accepted: Sep. 23, 2024
Available Online: Nov. 23, 2024

Keywords:

Software Effort Estimation
Analogy-Based Estimation
Non-Algorithmic Model
Genetic Algorithm
Particle Swarm Optimization

1- Introduction

Effort estimation is the process of forecasting the resources, time, and personnel required to develop software. It is a critical aspect of project management, as incorrect cost estimates can lead to project failures due to inaccurate planning and scheduling. Unlike other types of projects such as construction and manufacturing, the unique characteristics of software services make effort estimation a more challenging task. The intangibility of software services and the high variability and inconsistency of datasets add to the complexity of the process. As a result, a one-size-fits-all model cannot be applied to different types of software and datasets. Poor estimation can result in budget and timeline overruns or overestimation, which may ultimately lead to project failure. Therefore, an accurate and optimal approach to predicting these costs is crucial [1].

Cost estimation models that can accurately predict the expenses involved in constructing a system during the initial stages of project development, even with limited project information, are highly valuable and necessary. It is essential to have such models that can provide precise cost estimations during the early stages of project construction, offering significant benefits and advantages. This is especially true when there is a lack of data related to the project. By utilizing cost estimation methods, project managers can effectively manage the time and construction expenses associated with building the system. This allows for better planning and decision-making throughout the project's lifecycle, ultimately leading to a more successful outcome. Additionally, accurate cost estimation can help prevent unexpected expenses and delays, which can have a significant impact on the overall success of the project. Therefore, it is important to invest in reliable cost estimation models to ensure that projects are completed on time and within budget [2]. According to Boehm, effort estimation during the early stages of system construction can range from 25% to 40% of the actual effort required [3]; This preliminary estimation is often inaccurate due to the limited knowledge of the project at that stage. Heemstra has also confirmed this view [4].

Several approaches have been proposed for software effort estimation, which can be categorized into algorithmic and non-algorithmic methods. Algorithmic approaches use mathematical models that define a cost function based on selected cost factors [5]. The selection of cost factors and the definition of the cost function vary across different methods, and they can be analytical or experimental. Popular

*Corresponding author's email: k.mohebbi@khuisf.ac.ir

algorithmic methods include COCOMO and SLIM. However, algorithmic approaches are not flexible enough to provide accurate estimates for large and complex projects. Non-algorithmic approaches, on the other hand, rely on analytical comparisons with similar previous projects, using existing databases to infer the cost of software. These approaches do not use equations or mathematical models.

Analogy-based estimation is a popular method for estimating the effort required for software development, which does not involve the use of algorithms. This approach involves comparing the features of a new project with those of previously completed projects to determine the level of similarity and predict the amount of effort needed for the new project. By using this method, software developers can make informed decisions about resource allocation and project planning, based on past experiences and knowledge. This approach is particularly useful when dealing with complex projects that require a high level of expertise and experience, as it allows developers to draw on their previous successes and failures to make more accurate predictions about future outcomes. Overall, analogy-based estimation is an effective tool for improving software development processes and ensuring successful project outcomes [6]. The analogy-based estimation approach comprises four major components: historical dataset, similarity function, retrieval rules, and solution function. The estimation process involves the following steps [7]:

To estimate the effort required for a new software development project using analogy-based estimation, the first step is to gather data from previous projects and create a database of relevant information. This information can include metrics such as function points and lines of code. Next, appropriate measurement parameters are selected based on the requirements of the target project. The historical dataset is then used to retrieve previous projects and assess the similarity between those projects and the target project. Finally, the effort required for the target project is estimated based on the similarities found in the previous projects.

A technique for estimating software effort is case-based reasoning (CBR) which is based on adapting previously successful solutions for similar software projects. The CBR can be enhanced by combining it with meta-heuristic algorithms [8]. This study introduces a new method for estimating software effort that utilizes a case-based approach and the particle swarm optimization (PSO) algorithm.

The use of PSO has increased due to its simplicity, low cost of calculations, and also its effect on a wide range of applications. The advantages of this algorithm compared to other optimization algorithms can be summarized as follows: The PSO algorithm contains memory so that all particles keep good solutions. In this algorithm, the position of each community member changes based on personal experiences and the experiences of the whole community. Compared to other optimization strategies, the PSO algorithm shows more flexibility by using swarming particles against the local optimal problem. Finally, this algorithm is simpler than the other population-based algorithms such as the ant colony optimization algorithm. In addition, the initial quantification of the population using this algorithm is simpler than other intelligent optimization algorithms.

However, to improve the efficiency of this approach, the study proposes incorporating additional optimization algorithms, such as genetic algorithms, to enhance the weighting process. While analogical models typically perform better with weighted features, the PSO algorithm may not always produce optimal weights. By combining PSO with genetic algorithms, the study aims to achieve more accurate weights and improve overall estimation accuracy. Hybrid algorithms are particularly useful in this context as they strive to attain the global optimal point.

The remainder of this paper is structured as follows: In the subsequent section, notable works related to the topic are examined. Afterward, a detailed explanation of the suggested approach is provided. This is followed by an evaluation of the effectiveness of the approach. Finally, the concluding section summarizes the paper and provides recommendations for future research.

2- Related Works

So far, different studies have been conducted in the field of software development effort estimation, some prominent works of which are mentioned below.

The first ideas related to estimating software effort were presented by Dalkey and Helmer (1963) [9] under the name of the Delphi model, which is a non-algorithmic classification method. Due to algorithmic methods' inability to handle the dynamic behavior of software projects in the early stages, non-algorithmic methods were introduced. In this approach, experienced individuals share their estimates about effort until reaching a final agreement on a common effort level among all experts. Then, Albrecht (1983) [10] made a significant impact by introducing Function Point (FP), allowing measurement in the early stages of projects and mitigating the negative impacts of the previous method, lines of code estimation. Previous models required an initial estimate of lines of code for software, which could not be accurate until software implementation was completed, leading to significant estimation errors that were reduced with feature additions or functional points.

The initial idea of algorithmic methods with the use of lines of code in law was introduced by Boehm (1984) [11]. He proposed a new model called the Constructive Cost Model (COCOMO) for estimating software development effort using empirical equations. COCOMO is an empirical model derived from collecting data from various software projects. These data are analyzed to derive formulas that best-fit observations, relating system and product size, team, and project factors to the required work for system development. Other models like SLIM and SEM-SEER continued the principles of the COCOMO method.

Shepperd et al. (1996) [12] introduced an approach called ANGEL, which is designed to automatically handle the collection, storage, and identification of similar past projects to estimate the effort needed for a new project. ANGEL

operates by minimizing the Euclidean distance in multi-dimensional space. This approach is adaptable to various datasets, regardless of the number of projects or collected variables. Their method of using analogies was tested with six diverse datasets from various backgrounds and was proven to be more effective than alternative methods. In addition, this research demonstrates that analogical estimation is a promising approach, especially when supported by an automated environment.

Walkerden et al. (1999) [13] attempted to predict effort using similar project efforts as a benchmark. They first removed features not matching the project from the dataset and selected projects only within the relevant domain. Their model compared linear regression on performance points, showing that comparative estimation might yield better results than using performance points and algorithmic models, but they did not consider weighting software project features in their research. Classified features are evaluated based on linguistic values like low, complex, and important. Linguistic values may be derived from numerical values. When derived from numerical data, they are often represented based on classic distances. Also, linguistic value representation does not mimic human interpretation methods, thus encountering errors and uncertainties. Idri et al. (2006) [14] proposed a fuzzy comparison method to address this issue, combining fuzzy logic with comparison-based reasoning. The fuzzy comparison uses fuzzy sets for linguistic values instead of classic distances. Furthermore, fuzzy scaling contrasts numerical values by transforming them into linguistic values.

Azzeh et al. (2008) [15] examined similarities between two software projects described with numerical and classified features using fuzzy clustering and logic-based approaches for similarity measurement. Two approaches regarding weighting and Euclidean distance were evaluated for assessing software similarity performance, showing validity similar to case-based reasoning methods.

Keung et al. (2008) [16] presented a solution named Analogy-X, which is based on utilizing the Mantel correlation randomization test. They leverage the correlation strength between the distance matrices of project characteristics and the known effort values of the dataset. The approach demonstrates (1) employing Mantel's correlation to determine the suitability of analogies, (2) a systematic feature selection process, and (3) using a leverage statistic for identifying outlier data points. Analogy-X establishes a solid statistical foundation for analogies, eliminates the necessity for heuristic searches, and significantly enhances its algorithmic efficiency.

Attarzadeh and Ow (2010) [17] aimed to accurately estimate software costs using machine learning and pattern recognition methods. Artificial neural networks can learn from previous data to find relationships between independent and dependent variables.

Azzeh et al. (2011) [18] combined a fuzzy number-based comparison method with all available initial data to enhance early-stage software effort estimation performance.

Amazel et al. (2014) [19] carried out a study to prove that using analogy-based techniques is a feasible approach for estimating software. To handle data that was not obtained from numerical values and was classified, they utilized a fuzzy analogy method along with a widely-used clustering technique known as the fuzzy algorithm, which is particularly suitable for dealing with extensive datasets containing batch values. They assessed the similarity of software projects by analyzing clustering outcomes and determined the effort required for a new project based on the nearest scales. The findings of this research indicated that this method led to enhanced precision in software estimation. Therefore, it can be concluded that utilizing analogy-based methods can be an effective strategy for estimating software accurately.

Khatibi Bardsiri and Khatibi (2015) [20] discovered that comparative methods were overlooked in previous studies regarding the nature of software projects in estimation processes. They improved the performance of the comparative method based on three main project classification features: development types, organization types, and development platforms on a dataset of 448 real software projects.

Kumari and Pushkar (2016) [21] developed a framework for project selection using a multi-objective genetic algorithm to enhance the analogy-based method. They believed that project selection was crucial in the analogy-based method, and the interaction effects between the projects could significantly influence the estimated software cost. The proposed framework was tested on COCOMO and NASA datasets, and it was able to achieve better results. However, the use of meta-heuristic algorithms may lead to the model getting stuck in a local optimum.

Idri and Abnane (2017) [22] proposed a fuzzy analogy-based model that utilizes fuzzy logic to implement the analogy method. They compared this method with six other techniques for estimating effort and tested it on various datasets. The fuzzy analogy-based model outperformed the other methods, but the model was based only on numerical features and could not use nominal ones.

Wu et al. (2018) [8] devised a novel strategy to enhance software cost estimation by integrating case-based reasoning with the particle swarm meta-heuristic algorithm. The effectiveness of their approach was evaluated using the Maxwell and Desharnais datasets, yielding promising outcomes.

Ezghari and Zahi (2018) [23] introduced a fuzzy analogy-based technique to improve the precision of software effort estimation. The effectiveness of their method was evaluated on 13 datasets, and the results showcased superior performance and accuracy compared to previous studies in the field.

Mustafa and Abdelwahed (2019) [24] presented a stochastic forest model that underwent experimental optimization, where key parameters were adjusted to enhance software project effort estimation. The performance of their optimized model was compared to that of the traditional regression tree, revealing that the optimized stochastic forest model surpassed the regression tree model in all evaluation criteria.

These studies demonstrate the importance of developing

new approaches to improve software estimation techniques, as well as the potential benefits of combining different methods to achieve better results. In recent years, there has been considerable research in the field of software development effort estimation. A review of recent studies reveals that many researchers are focusing on using meta-heuristic and optimization algorithms to enhance the efficiency of the estimation process.

Shah et al. (2020) [25] proposed a similarity-based estimation method using the artificial bee colony (ABC) algorithm, while Ranichandra (2020) [26] tried to enhance the non-orthogonal space distance, a metric indicating how similar software projects are by analyzing feature weights and redundancy through the ant colony optimization (ACO) algorithm.

Shahpar et al. (2021) [27] proposed an evolutionary ensemble analogy-based method that combines genetic algorithms with various analogy-based methods. Samavatian and Mohebbi (2021) [28] used the cuckoo search algorithm to select software features and then analyzed the results further using the particle swarm optimization algorithm. In another study, Shahpar et al. (2021) [29] utilized a combination of particle swarm optimization and simulated annealing to calculate project effort using a polynomial ensemble of different analogy-based estimation models.

A framework was introduced by Dashti et al. (2022) [30] to calculate the cost of software development. This framework utilizes the learnable evolution model, which is a technique that optimizes the weighting of different features. By implementing this model, software developers can more accurately estimate the cost of their projects and allocate resources accordingly. The authors suggest that this approach can lead to more efficient and effective software development processes. Additionally, they propose that this framework can be adapted for use in various industries beyond just software development.

The CBR technique is centered around adapting successful solutions from past software projects that are similar in nature. However, CBR encounters a challenge in the form of multiple parameters that are difficult to fine-tune. This underscores the significance of the adaptation and adjustment process as a fundamental aspect of CBR, aiming to generate precise and efficient results with minimal estimation error. In their study, Hameed et al. (2023) [31] employed the Genetic Algorithm (GA) to aid in identifying the optimal set of classical CBR parameters, thereby enhancing the accuracy of effort estimation for software projects. The proposed CBR-GA model effectively demonstrated the efficacy of utilizing the GA algorithm to explore the best combination of CBR parameters, leading to improved accuracy.

Moradbeiky (2023) [32] introduced a novel model, namely FEEM that incorporates data filtering and feature weighting techniques across three layers. The initial two layers employ tools and methods to select key features and assign weights using the Lightning Search Algorithm (LSA). The third layer combines LSA with an artificial neural network to create an estimator model, enhancing final estimation accuracy. This hierarchical structure enhances accuracy by filtering and analyzing data from lower layers, as demonstrated through evaluations of diverse datasets, showcasing improved software effort estimation precision.

Determining the right quantity of comparable past projects for reuse is a challenge in Analogy-based software effort estimation techniques, and the effectiveness and precision of these methods are significantly impacted by the quality of software datasets. Thus, Pal et al. (2024) [33] introduced a new method to determine the appropriate number of analogs from high-quality datasets in Analogy-based software effort estimation. Their approach involves using Spearman's rank-order correlation and Kruskal-Wallis test during data pre-processing to handle both numerical/ordinal and nominal attributes effectively. The method identifies reliable attributes that have a significant impact on effort estimation, leading to improved dataset quality, better attribute selection, reduced anomalies, and lower project development costs according to experimental results.

Overall, these studies demonstrate a growing interest in the use of meta-heuristic and optimization algorithms to improve the accuracy and efficiency of software development effort estimation.

3- Proposed Approach

Overall, these studies demonstrate a growing interest in the use of meta-heuristic and optimization algorithms to improve.

To enhance the precision of software cost estimation, it is crucial to accurately assign weights to the software features. To achieve this objective, we suggest a novel approach that combines Particle Swarm Optimization (PSO) and genetic algorithm. PSO is a widely recognized meta-heuristic algorithm that has demonstrated remarkable success in resolving diverse problems [34]. Similarly, the genetic algorithm is among the most successful nature-inspired meta-heuristic algorithms [35]. By integrating PSO and genetic algorithm, we aim to optimize the weights of the software features and improve the accuracy of cost estimation. This approach has the potential to provide a more efficient and effective solution for software development teams seeking to improve their cost estimation processes. In addition, we aim to reduce the probability of getting stuck in a local optimization point and achieve global optimization in the feature weights. The process of our proposed approach is depicted in Fig. 1.

The algorithm begins by creating a random population of particles, each with the following properties:

Position: Indicates the position of each particle in each iteration of the algorithm.

Best position: The best position that each particle had in the whole execution of the algorithm.

Velocity: Indicates the amount of displacement of each particle per iteration of the algorithm. This value is updated in each iteration.

Cost: The amount of cost that each particle has in its current position and its amount is calculated by the cost function.

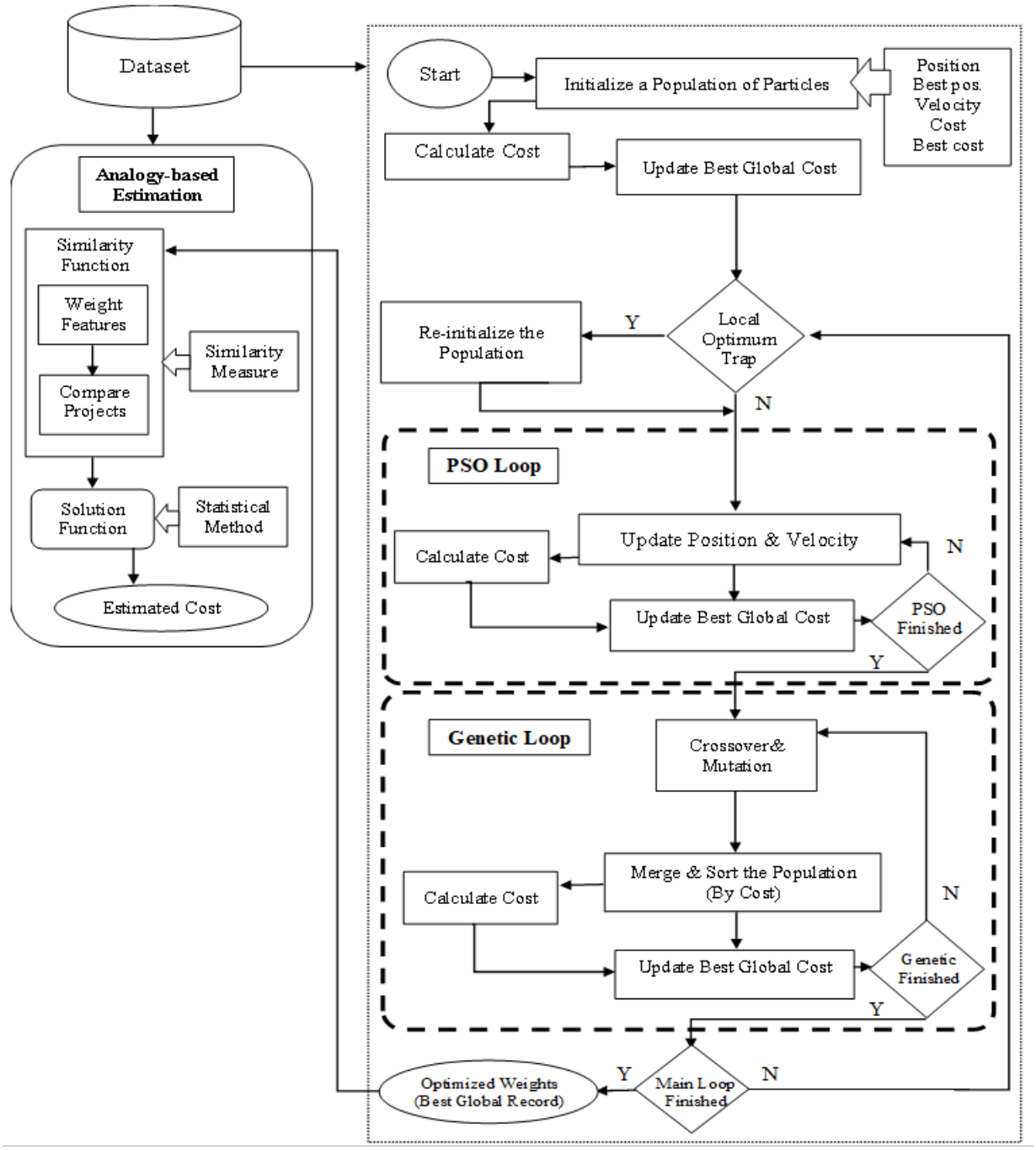

Fig. 1. Flowchart of the Proposed Approach

Best cost: The lowest cost that each particle had in the whole execution of the algorithm.

To improve the accuracy of weight generation, it is crucial to establish a suitable cost function that can effectively measure the performance of the generated weights. This study has utilized commonly used evaluation criteria for software cost estimates to create a function that incorporates *MMRE*, *MdMRE*, and *PRED*. The formula for this function is as follows:

$$Cost = MMRE + MdMRE - PRED \quad (1)$$

Where *MMRE* represents the mean magnitude of relative error, *MdMRE* denotes the median magnitude of relative error, and *PRED* reflects the percentage of relative error deviation [36]. As the objective is to minimize *MMRE* and *MdMRE* and bring *PRED* closer to zero, their sum is used to define the cost function. This function assigns higher priority to the weights that produce lower values.

To start running the algorithm, certain input parameters need to be defined, such as the number of features to be weighted ($Size_{feature}$), the initial population size ($Size_{population}$), the lower and upper limits for each feature ($Limit_{low}$ and $Limit_{up}$), and the maximum number of iterations ($Iter_{max}$). Once these inputs are specified, the algorithm proceeds to generate the initial population. This is done by assigning random positions to particles, which in this case represent the weights of the features. Afterward, the cost function is utilized to evaluate the cost of every particle. Subsequently, the particles are arranged in an ascending order based on their cost and position so that it becomes more convenient to access favorable genes within the genetic algorithm. The best global record of the population, including its position and cost, is then stored in a variable.

The proposed approach involves three loops as follows:

Main loop: The algorithm has a mechanism to prevent getting stuck in a local optimal point. If there is no improvement in the best cost after four iterations, a revolution occurs. In this revolution, the positions of all particles, except for one particle that serves as the superior generation, are randomly reinitialized for the next iteration. The primary loop carries out both the PSO and genetic loops simultaneously.

PSO loop: the velocity of each particle is adjusted, and it is verified whether the particle's movement falls within the predetermined range for its attributes. Subsequently, the particles are repositioned according to the calculated velocity, and their position and cost are recalibrated. If a particle encounters a more favorable experience compared to its previous best experience, the latter is replaced with the new value. Conversely, if the experience is inferior to the best cost observed by the entire population, the global record of the best outcome is updated accordingly.

Genetic loop: Here, particles are combined and mutated to produce a new population. During this process, it is checked whether each feature is within the specified range. Particle velocity is also considered in the genetic algorithm, and thus, it is genetically combined in the crossover and mutation of the population. The new population is the sum of the initial, crossed-over, and mutated populations. The best global record is updated if needed.

Once the feature weights are generated based on their importance in estimating the effort required, they are used in the similarity function of the analogy-based method. The similarity function employs distance measures to compare equivalent features, and based on the comparison of various features, the closest project to the current project is found.

4- Evaluation

This section outlines the methodology for assessing the proposed approach.

4- 1- Implementation Settings

The parameters of the proposed approach are set as depicted in Table 1.

4- 2- Datasets

Two commonly used datasets, namely Maxwell and Desharnais have been selected to validate the proposed approach. The Maxwell dataset contains 62 projects from one of the biggest commercial banks in Finland [37]. For each project, there are 25 features. The effort is determined by the number of working hours performed by the software developer, from the specifications to the time of delivery. The Desharnais dataset contains 81 projects from a Canadian Software house [38]. For each project, there exist 10 features. The effort is measured in person-hours.

4- 3- Data Preprocessing

In this research, normalization has been utilized as a crucial step for data preparation and preprocessing. Normalization is an essential technique that helps ensure that data is consistent and comparable across different scales and units of measurement. Among the various normalization methods available, the Min-Max method has been chosen. This method involves mapping each set of data to arbitrary intervals with known minimum and maximum values. By using a simple conversion formula, any desired interval can be mapped to a new one. For instance, if we want to map feature A, which falls between *min_A* and *max_A*, to a new range between *new_Min* and *new_Max*, we can use Eq. (2) to convert any initial value v in the initial interval to its corresponding value v' in the new interval.

$$v' = (v - minA)\frac{newMax - newMin}{newMax - newMin} + (newMin) \quad (2)$$

4- 4- Evaluation Criteria

The criteria used to evaluate the software development effort in this study are depicted in Table 2 [39].

Table 2. Evaluation Criteria

Criteria	Equation
Magnitude of Relative Error (MRE)	$\frac{\lvert ActualEffort_i - EstimatedEffort_i \rvert}{ActualEffort_i}$
Mean Magnitude of Relative Error (MMRE)	$\frac{\sum_{k=1}^{N} MRE}{N}$
Median Magnitude of Relative Error (MdMRE)	$Median(MREs)$
Percentage of the Prediction (PRED 0.25)	$\frac{100}{N}\sum_{i=1}^{N}\begin{cases}1 \; If \; MRE_i \leq \frac{25}{100} \\ 0 \; Else\end{cases}$

4- 5- Evaluation Method

Previous research works have highlighted that techniques like n-fold can result in highly variable software effort estimates [40]. To achieve more precise results, methods like Leave-One-Out Cross-Validation (LOOCV) or Hold-out should be utilized [41]. In this study, as the dataset is not extensive and all samples can be treated as test data simultaneously, the LOOCV approach was employed to validate the algorithm.

4- 6- Experimental Results

The first criterion to evaluate the proposed approach is the MRE, which shows the deviation of the predicted from the actual estimation. The best case is zero, and this value increases when the estimates get further from reality. The calculated MRE for the two datasets is shown in Fig. 2.

This figure highlights that a few projects from both datasets contain a significant number of errors, as they have very different effort levels compared to other projects. In some research areas, these cases may be identified as outliers and either normalized or removed from the dataset. However, since the Desharnais and Maxwell datasets are based on real projects with highly precise data, it is unlikely that errors occurred. Although these cases are rare, they can result in high error rates when implementing the analogy-based method, as it may not find similar projects to compare them with. Therefore, it is normal for the error rate of these cases to be high.

Afterward, we evaluated the proposed approach using the MMRE, MdMRE, and PRED(0.25) criteria. The result is shown in Fig. 3.

While the MRE calculates the difference in effort for a project, MMRE and MdMRE are cumulative measures that represent the relative error magnitude for all projects in the dataset. PRED(0.25) measures the percentage of predicted values within 25 percent of the actual value. We tried to keep MMRE and MdMRE values as close to zero as possible. In contrast, PRED(0.25) aims to achieve the highest possible value of one.

We also introduced the TotalCost criterion, which is the overall measure of the previous criteria, calculated using (1). TotalCost values for both datasets have improved over the similar work. The Desharnais dataset's TotalCost is negative, which is expected, as the best possible values for MMRE, MdMRE, and PRED are 0, 0, and 1, respectively. Hence, TotalCost is equal to -1 in the best case.

In the following, the performance of the proposed approach has been compared with previous methods. To do this, it is necessary for the compared methods to use the same evaluation criteria as well as the datasets. Therefore, we have decided to compare them based on each of the evaluation criteria MMRE, MdMRE, and PRED(0.25), selecting some of the most prominent works for each criterion.

Table III shows the evaluation result of MMRE in different methods for each dataset.

The comparison of MMRE in different methods for each

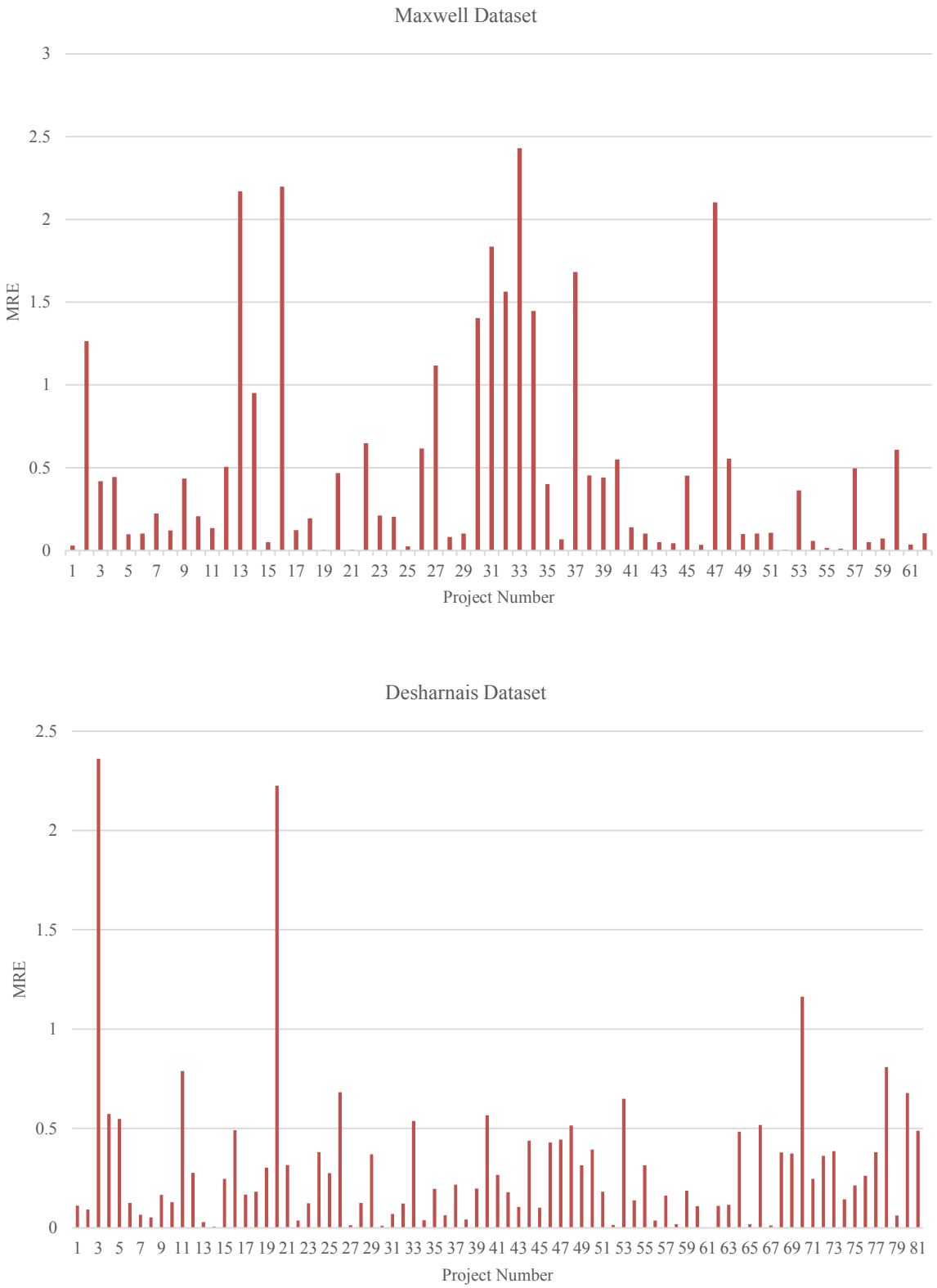

Fig. 2. MRE of the Proposed Approach

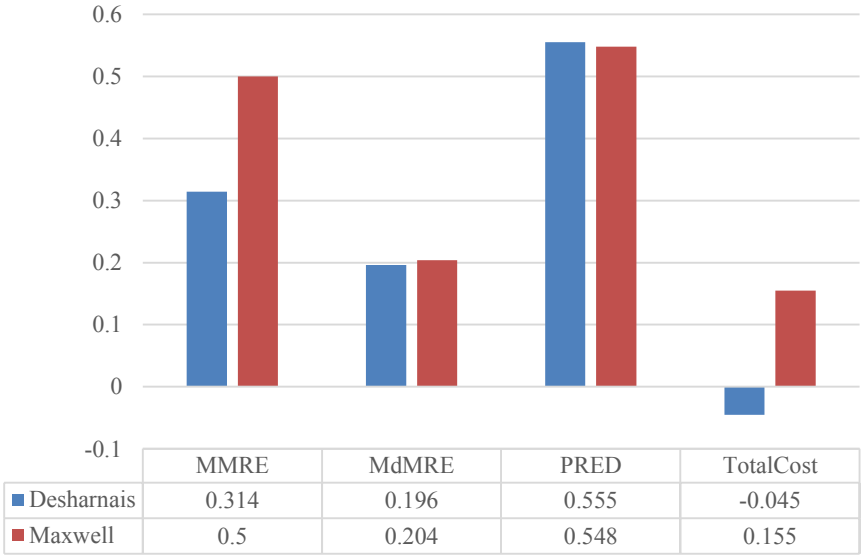

Fig. 3. Evaluation of the Proposed Approach

Table 3. Evaluation of MMRE in Different Methods

Method	Desharnais	Maxwell
PSO+CBR [8]	0.57	0.53
ANGEL [12]	0.38	0.61
Analogy-X [16]	0.38	0.91
Pal et al. [33]	0.31	0.46
Proposed Approach	0.31	0.5

dataset is depicted in Fig. 4.

Table 4 shows the evaluation result of MdMRE in different methods for each dataset.

The comparison of MdMRE in different methods for each dataset is depicted in Fig. 5.

Table 5 shows the evaluation result of PRED in different methods for each dataset.

The comparison of PRED in different methods for each dataset is depicted in Fig. 6.

The evaluation results indicate that the integration of PSO and genetic algorithms yields improved points of optimization and reduces the chances of being trapped in local optima, as compared to using these algorithms individually. As a result, by utilizing the global optimization weights for project features and applying them in the analogy-based method, more accurate estimations can be made with fewer errors.

Furthermore, we conducted additional experiments to determine the best match between different cases for solution and similarity functions. This involved implementing a grid search of all possible cases, including the number of projects selected that are most similar to the target project ranging from 1 to 4. Six distance functions, including Euclidean, Manhattan, Gray relational grade, Minkowski, Canberra, and Bray-Curtis, are being considered as similarity functions. Additionally, four statistical functions - mean, median, weighted mean, and inverse distance weighted mean - are being considered in relation to the number of projects. The results of implementing the grid search for the Maxwell

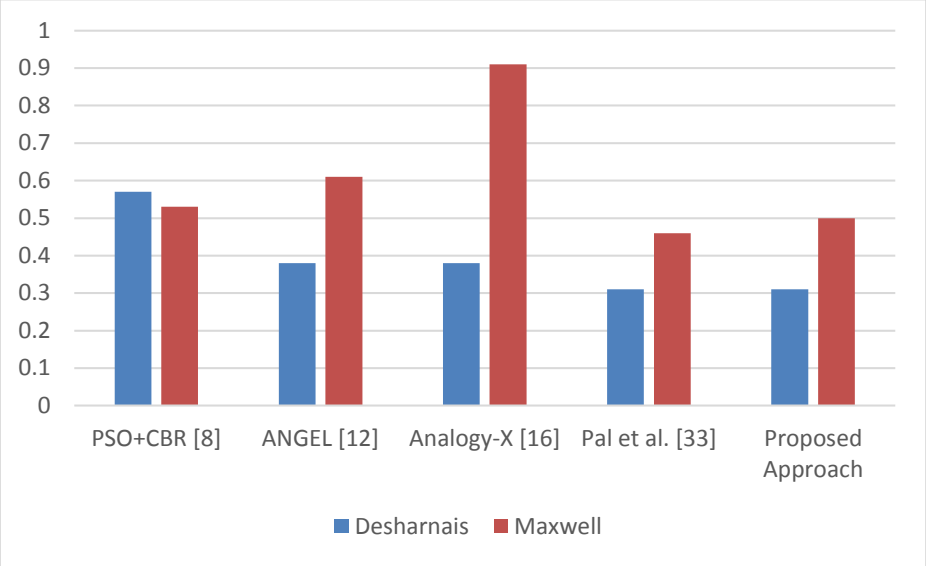

Fig. 4. Comparison of MMRE in Different Methods

Table 4. Evaluation of MdMRE in Different Methods

Method	Desharnais	Maxwell
PSO + ABE [7]	0.4	0.47
PSO+CBR [8]	0.41	0.44
RF [24]	0.39	0.32
ACO + ABE [26]	0.36	0.48
FEEM [32]	0.22	0.24
Proposed Approach	0.2	0.2

and Desharnais dataset can be found in Fig. 7 and Fig. 8, respectively.

It is apparent that the cost of estimation varies depending on the similarity and solution functions utilized. However, the number of projects considered for cost estimation has a significant impact on the error rate. In both the Maxwell and Desharnais datasets, the error rate reaches its minimum when four similar projects are chosen. In these instances, making changes to the similarity or solution functions does not significantly affect the error rate. However, if the number of selected projects is less than four, the error rate increases, and modifications to the similarity or solution functions may have minimal influence on it. Therefore, in the analogy-based method, the crucial factor for cost estimation is the number of similar projects selected, rather than the choice of similarity and solution functions.

5- Conclusion and Future Works

The analogy-based estimation method is a popular and valuable non-algorithmic approach employed for software effort estimation, primarily due to its practicality and effectiveness. However, this method faces challenges in accurately estimating effort when project features are not independent or have different levels of importance. To overcome this challenge, this paper proposes a new approach for analogy-based estimation by combining PSO and genetic algorithms to improve feature weighting and prevent local optimum trapping. The evaluation results show

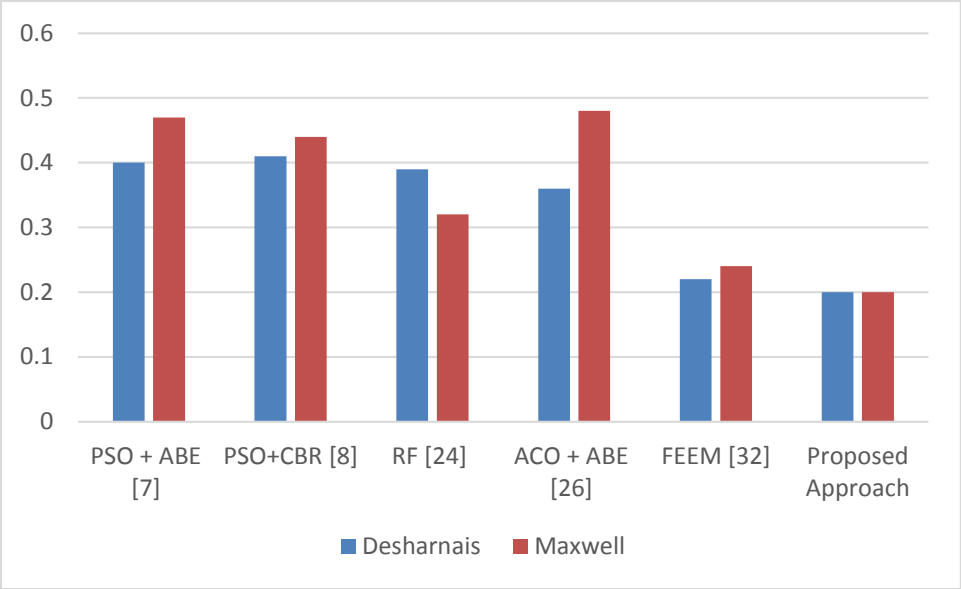

Fig. 5. Comparison of MdMRE in Different Methods

Table 5. Evaluation of PRED in Different Methods

Method	Desharnais	Maxwell
PSO + ABE [7]	0.4	0.29
PSO+CBR [8]	0.36	0.32
ANGEL [12]	0.43	0.21
Analogy-X [16]	0.43	0.23
RF [24]	0.36	0.4
ACO + ABE [26]	0.36	0.32
FEEM [32]	0.51	0.5
Pal et al. [33]	0.45	0.33
Proposed Approach	0.56	0.55

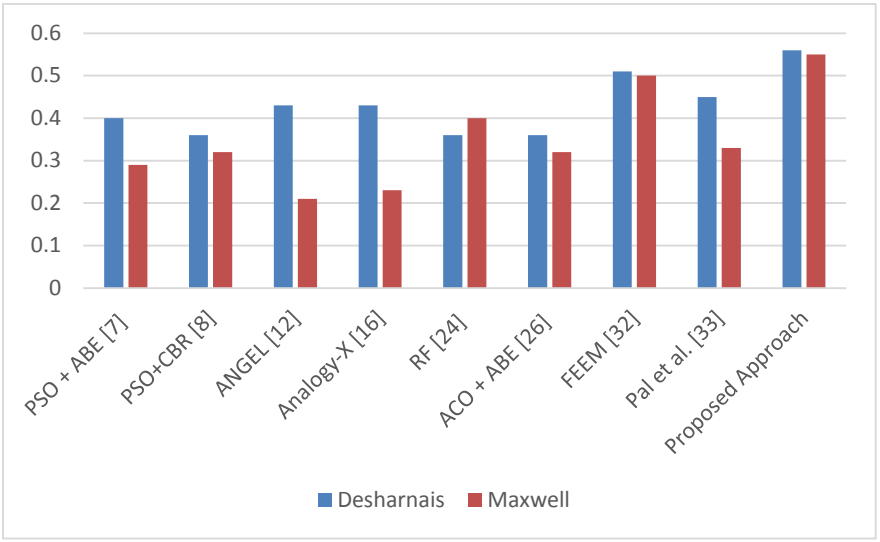

Fig. 6. Comparison of PRED in Different Methods

No. of Projects	Sim. Func.	Sol. Func.	Total Cost	No. of Projects	Sim. Func.	Sol. Func.	Total Cost	No. of Projects	Sim. Func.	Sol. Func.	Total Cost	No. of Projects	Sim. Func.	Sol. Func.	Total Cost
1	EUC	M	1.835	2	EUC	M	1.238	3	EUC	M	0.943	4	EUC	M	0.943
1	EUC	Md	1.835	2	EUC	Md	1.238	3	EUC	Md	0.943	4	EUC	Md	0.943
1	EUC	WM	1.835	2	EUC	WM	1.238	3	EUC	WM	0.943	4	EUC	WM	0.765
1	EUC	IDWM	1.835	2	EUC	IDWM	1.238	3	EUC	IDWM	0.943	4	EUC	IDWM	0.765
1	MAN	M	1.793	2	MAN	M	1.238	3	MAN	M	0.943	4	MAN	M	0.765
1	MAN	Md	1.793	2	MAN	Md	1.238	3	MAN	Md	0.943	4	MAN	Md	0.765
1	MAN	WM	1.793	2	MAN	WM	1.238	3	MAN	WM	0.943	4	MAN	WM	0.765
1	MAN	IDWM	1.793	2	MAN	IDWM	1.238	3	MAN	IDWM	0.943	4	MAN	IDWM	0.765
1	GRA	M	1.793	2	GRA	M	1.238	3	GRA	M	0.943	4	GRA	M	0.765
1	GRA	Md	1.793	2	GRA	Md	1.238	3	GRA	Md	0.943	4	GRA	Md	0.765
1	GRA	WM	1.793	2	GRA	WM	1.238	3	GRA	WM	0.943	4	GRA	WM	0.765
1	GRA	IDWM	1.793	2	GRA	IDWM	1.238	3	GRA	IDWM	0.943	4	GRA	IDWM	0.765
1	MIN	M	1.793	2	MIN	M	1.238	3	MIN	M	0.943	4	MIN	M	0.765
1	MIN	Md	1.793	2	MIN	Md	1.238	3	MIN	Md	0.943	4	MIN	Md	0.765
1	MIN	WM	1.793	2	MIN	WM	1.238	3	MIN	WM	0.943	4	MIN	WM	0.765
1	MIN	IDWM	1.793	2	MIN	IDWM	1.238	3	MIN	IDWM	0.943	4	MIN	IDWM	0.765
1	CAN	M	1.793	2	CAN	M	1.238	3	CAN	M	0.943	4	CAN	M	0.765
1	CAN	Md	1.793	2	CAN	Md	1.238	3	CAN	Md	0.943	4	CAN	Md	0.765
1	CAN	WM	1.793	2	CAN	WM	1.238	3	CAN	WM	0.943	4	CAN	WM	0.765
1	CAN	IDWM	1.793	2	CAN	IDWM	1.238	3	CAN	IDWM	0.943	4	CAN	IDWM	0.765
1	BRC	M	1.793	2	BRC	M	1.238	3	BRC	M	0.943	4	BRC	M	0.765
1	BRC	Md	1.793	2	BRC	Md	1.238	3	BRC	Md	0.943	4	BRC	Md	0.765
1	BRC	WM	1.793	2	BRC	WM	1.238	3	BRC	WM	0.943	4	BRC	WM	0.765
1	BRC	IDWM	1.793	2	BRC	IDWM	1.238	3	BRC	IDWM	0.943	4	BRC	IDWM	0.765

Fig. 7. Grid Search Result for Maxwell Dataset

No. of Projects	Sim. Func.	Sol. Func.	Total Cost	No. of Projects	Sim. Func.	Sol. Func.	Total Cost	No. of Projects	Sim. Func.	Sol. Func.	Total Cost	No. of Projects	Sim. Func.	Sol. Func.	Total Cost
1	EUC	M	1.065	2	EUC	M	0.934	3	EUC	M	0.734	4	EUC	M	0.701
1	EUC	Md	1.065	2	EUC	Md	0.934	3	EUC	Md	0.734	4	EUC	Md	0.701
1	EUC	WM	1.065	2	EUC	WM	0.824	3	EUC	WM	0.734	4	EUC	WM	0.701
1	EUC	IDWM	1.065	2	EUC	IDWM	0.824	3	EUC	IDWM	0.734	4	EUC	IDWM	0.701
1	MAN	M	0.981	2	MAN	M	0.824	3	MAN	M	0.734	4	MAN	M	0.646
1	MAN	Md	0.981	2	MAN	Md	0.824	3	MAN	Md	0.734	4	MAN	Md	0.361
1	MAN	WM	0.981	2	MAN	WM	0.824	3	MAN	WM	0.734	4	MAN	WM	0.361
1	MAN	IDWM	0.981	2	MAN	IDWM	0.824	3	MAN	IDWM	0.734	4	MAN	IDWM	0.361
1	GRA	M	0.981	2	GRA	M	0.824	3	GRA	M	0.734	4	GRA	M	0.361
1	GRA	Md	0.981	2	GRA	Md	0.824	3	GRA	Md	0.734	4	GRA	Md	0.361
1	GRA	WM	0.981	2	GRA	WM	0.824	3	GRA	WM	0.734	4	GRA	WM	0.361
1	GRA	IDWM	0.981	2	GRA	IDWM	0.824	3	GRA	IDWM	0.734	4	GRA	IDWM	0.361
1	MIN	M	0.981	2	MIN	M	0.747	3	MIN	M	0.734	4	MIN	M	0.361
1	MIN	Md	0.981	2	MIN	Md	0.747	3	MIN	Md	0.734	4	MIN	Md	0.361
1	MIN	WM	0.981	2	MIN	WM	0.734	3	MIN	WM	0.701	4	MIN	WM	0.361
1	MIN	IDWM	0.981	2	MIN	IDWM	0.734	3	MIN	IDWM	0.701	4	MIN	IDWM	0.361
1	CAN	M	0.981	2	CAN	M	0.734	3	CAN	M	0.701	4	CAN	M	0.361
1	CAN	Md	0.981	2	CAN	Md	0.734	3	CAN	Md	0.701	4	CAN	Md	0.361
1	CAN	WM	0.981	2	CAN	WM	0.734	3	CAN	WM	0.701	4	CAN	WM	0.361
1	CAN	IDWM	0.981	2	CAN	IDWM	0.734	3	CAN	IDWM	0.701	4	CAN	IDWM	0.361
1	BRC	M	0.981	2	BRC	M	0.734	3	BRC	M	0.701	4	BRC	M	0.361
1	BRC	Md	0.981	2	BRC	Md	0.734	3	BRC	Md	0.701	4	BRC	Md	0.361
1	BRC	WM	0.981	2	BRC	WM	0.734	3	BRC	WM	0.701	4	BRC	WM	0.361
1	BRC	IDWM	0.981	2	BRC	IDWM	0.734	3	BRC	IDWM	0.701	4	BRC	IDWM	0.361

Fig. 8. Grid Search Result for Desharnais Dataset

that this approach outperforms similar works. The grid search analysis reveals that the number of similar projects selected for estimating effort is more important than the choice of similarity and solution functions.

Future research could explore the use of other optimization algorithms for feature weighting or investigate the limitations and challenges of using machine learning methods for estimating software efforts, such as small training datasets, data uncertainties, qualitative metrics, and human factors.

References

[1] A. Sharma, D.S. Kushwaha, Estimation of software development effort from requirements based complexity, Procedia Technology, 4 (2012) 716-722.

[2] G.-H. Kim, S.-H. An, K.-I. Kang, Comparison of construction cost estimating models based on regression analysis, neural networks, and case-based reasoning, Building and environment, 39(10) (2004) 1235-1242.

[3] B. Boehm, Cost estimation with COCOMO II, in, University of Southern California, Center for Software Engineering, 2002.

[4] F.J. Heemstra, Software cost estimation, Information and software technology, 34(10) (1992) 627-639.

[5] I. Sommerville, Software engineering 9th Edition, ISBN-10, 137035152 (2011) 18.

[6] A.R. Gray, S.G. MacDonell, A comparison of techniques for developing predictive models of software metrics, Information and software technology, 39(6) (1997) 425-437.

[7] V.K. Bardsiri, D.N.A. Jawawi, S.Z.M. Hashim, E. Khatibi, A PSO-based model to increase the accuracy of software development effort estimation, Software Quality Journal, 21(3) (2013) 501-526.

[8] D. Wu, J. Li, C. Bao, Case-based reasoning with optimized weight derived by particle swarm optimization for software effort estimation, Soft Computing, 22(16) (2018) 5299-5310.

[9] N. Dalkey, O. Helmer, An experimental application of the Delphi method to the use of experts, Management science, 9(3) (1963) 458-467.

[10] A.J. Albrecht, J.E. Gaffney, Software function, source lines of code, and development effort prediction: a software science validation, IEEE transactions on software engineering, (6) (1983) 639-648.

[11] B.W. Boehm, Software engineering economics, IEEE transactions on Software Engineering, (1) (1984) 4-21.

[12] M. Shepperd, C. Schofield, B. Kitchenham, Effort estimation using analogy, in: Proceedings of IEEE 18th International Conference on Software Engineering, IEEE, 1996, pp. 170-178.

[13] F. Walkerden, R. Jeffery, An empirical study of analogy-based software effort estimation, Empirical software engineering, 4 (1999) 135-158.

[14] A. Idri, A. Zahi, A. Abran, Generating fuzzy term sets for software project attributes using fuzzy c-means and real coded genetic algorithms, in: Proceedings of the International Conference on Information and Communication Technology for the Muslim World (ICT4M), Malaysia, 2006, pp. 120-127.

[15] M. Azzeh, D. Neagu, P. Cowling, Software project similarity measurement based on fuzzy C-means, in: Making Globally Distributed Software Development a Success Story: International Conference on Software Process, ICSP 2008 Leipzig, Germany, May 10-11, 2008 Proceedings, Springer, 2008, pp. 123-134.

[16] J.W. Keung, B.A. Kitchenham, D.R. Jeffery, Analogy-X: providing statistical inference to analogy-based software cost estimation, IEEE Transactions on Software Engineering, 34(4) (2008) 471-484.

[17] I. Attarzadeh, S.H. Ow, Proposing a new software cost estimation model based on artificial neural networks, in: 2010 2nd International Conference on Computer Engineering and Technology, IEEE, 2010, pp. V3-487-V483-491.

[18] M. Azzeh, D. Neagu, P.I. Cowling, Analogy-based software effort estimation using Fuzzy numbers, Journal of Systems and Software, 84(2) (2011) 270-284.

[19] F.-A. Amazal, A. Idri, A. Abran, An analogy-based approach to estimation of software development effort using categorical data, in: 2014 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement, IEEE, 2014, pp. 252-262.

[20] V. Khatibi Bardsiri, E. Khatibi, Insightful analogy-based software development effort estimation through selective classification and localization, Innovations in Systems and Software Engineering, 11 (2015) 25-38.

[21] S. Kumari, S. Pushkar, A framework for analogy-based software cost estimation using multi-objective genetic algorithm, in: Proceedings of the world congress on engineering and computer Science, 2016.

[22] A. Idri, I. Abnane, Fuzzy analogy based effort estimation: An empirical comparative study, in: 2017 IEEE International Conference on Computer and Information Technology (CIT), IEEE, 2017, pp. 114-121.

[23] S. Ezghari, A. Zahi, Uncertainty management in software effort estimation using a consistent fuzzy analogy-based method, Applied Soft Computing, 67 (2018) 540-557.

[24] H. Mustapha, N. Abdelwahed, Investigating the use of random forest in software effort estimation, Procedia computer science, 148 (2019) 343-352.

[25] M.A. Shah, D.N.A. Jawawi, M.A. Isa, M. Younas, A. Abdelmaboud, F. Sholichin, Ensembling artificial bee colony with analogy-based estimation to improve software development effort prediction, IEEE Access, 8 (2020) 58402-58415.

[26] S. Ranichandra, Optimizing non-orthogonal space distance using ACO in software cost estimation, Mukt Shabd J, 9(4) (2020) 1592-1604.

[27] Z. Shahpar, V.K. Bardsiri, A.K. Bardsiri, An evolutionary ensemble analogy-based software effort estimation, Software: Practice and Experience, (2021).

[28] S. Samavatian, K. Mohebbi, Improving the Estimation of Software Development Effort Using the Combination of Cuckoo Search and Particle Swarm Optimization Algorithms, Journal of Soft Computing and Information Technology, 10(3) (2021) 86-98.

[29] Z. Shahpar, V.K. Bardsiri, A.K. Bardsiri, Polynomial analogy-based software development effort estimation using combined particle swarm optimization and simulated annealing, Concurrency and Computation: Practice and Experience, 33(20) (2021) e6358.

[30] M. Dashti, T.J. Gandomani, D.H. Adeh, H. Zulzalil, A.B.M. Sultan, LEMABE: a novel framework to improve analogy-based software cost estimation using learnable evolution model, PeerJ Computer Science, 7 (2022) e800.

[31] S. Hameed, Y. Elsheikh, M. Azzeh, An optimized case-based software project effort estimation using genetic algorithm, Information and Software Technology, 153 (2023) 107088.

[32] A. Moradbeiky, FEEM: A Flexible Model based on Artificial Intelligence for Software Effort Estimation, Journal of AI and Data Mining, 11(1) (2023) 39-51.

[33] N. Pal, M.P. Yadav, D.K. Yadav, Appropriate number of analogues in analogy based software effort estimation using quality datasets, Cluster Computing, 27(1) (2024) 531-546.

[34] R. Eberhart, J. Kennedy, Particle swarm optimization, in: Proceedings of the IEEE international conference on neural networks, Citeseer, 1995, pp. 1942-1948.

[35] M. Mitchell, An introduction to genetic algorithms, MIT press, 1998.

[36] S.D. Conte, H.E. Dunsmore, V.Y. Shen, Software engineering metrics and models, Benjamin-Cummings Publishing Co., Inc., 1986.

[37] K.D. Maxwell, Applied statistics for software managers, Applied Statistics for Software Managers, (2002).

[38] J. Desharnais, Analyse statistique de la productivitie des projects informatique a partie de la technique des point des function, Masters Thesis University of Montreal, (1989).

[39] K. Khan, The Evaluation of Well-known Effort Estimation Models based on Predictive Accuracy Indicators, in, 2010.

[40] E. Kocaguneli, T. Menzies, Software effort models should be assessed via leave-one-out validation, Journal of Systems and Software, 86(7) (2013) 1879-1890.

[41] J. Wen, S. Li, Z. Lin, Y. Hu, C. Huang, Systematic literature review of machine learning based software development effort estimation models, Information and Software Technology, 54(1) (2012) 41-59.

HOW TO CITE THIS ARTICLE

E. Nasr, K. Mohebbi, Improving Software Effort Estimation through a Hybrid Approach of Metaheuristic Algorithms in Analogy-based Method, AUT J. Model. Simul., 56(2) (2024) 155-170.

DOI: 10.22060/miscj.2024.23316.5366