Optimization of COCOMO Model using Particle Swarm Optimization

Noor Azura Zakaria [a,1], Amelia Ritahani Ismail [a,2,*], Nadzurah Zainal Abidin [a,3], Nur Hidayah Mohd Khalid [a,4], Afrujaan Yakath Ali [a,5]

[a] Department of Computer Science, International Islamic University Malaysia, Kuala Lumpur, Malaysia
[1] azurazakaria@iium.edu.my; [2] amelia@iium.edu.my; [3] nadzurah.zabidin@gmail.com, [4] hidayahkhalid7@gmail.com, [5] afrujaan@gmail.com
* corresponding author

ARTICLE INFO

Article history
Received October 30, 2020
Revised December 27, 2020
Accepted April 24, 2021
Available online April 24, 2021

Keywords
Particle Swarm Optimization
COCOMO model
Software effort
Estimation model
NASA

ABSTRACT

Software effort and cost estimation are crucial parts of software project development. It determines the budget, time, and resources needed to develop a software project. The success of a software project development depends mainly on the accuracy of software effort and cost estimation. A poor estimation will impact the result, which worsens the project management. Various software effort estimation model has been introduced to resolve this problem. COnstructive COst MOdel (COCOMO) is a well-established software project estimation model; however, it lacks accuracy in effort and cost estimation, especially for current projects. Inaccuracy and complexity in the estimated effort have made it difficult to efficiently and effectively develop software, affecting the schedule, cost, and uncertain estimation directly. In this paper, Particle Swarm Optimization (PSO) is proposed as a metaheuristics optimization method to hybrid with three traditional state-of-art techniques such as Support Vector Machine (SVM), Linear Regression (LR), and Random Forest (RF) for optimizing the parameters of COCOMO models. The proposed approach is applied to the NASA software project dataset downloaded from the promise repository. The proposed approach has been compared with the three traditional algorithms; however, the obtained results confirm low accuracy before hybridizing with PSO. Overall, the results showed that PSOSVM on the NASA software project dataset could improve effort estimation accuracy and outperform other models.

1. Introduction

Estimating software effort is an essential and crucial activity for the software development life cycle as it requires estimating the effort and cost at the initial stage of the project. An accurate effort estimation leads to the effective and efficient development of software and decreased risks [1][2]. Estimations aim to accurately and control the cost and time boundaries of the project planning [3][4]. The key parameters of effort estimation are time and cost, which are based on two reasons: to present software that is adaptable in a limited time frame and fill the gap between software and hardware progressions; and to generate software under the budget and time as responding to changeable customer demands [5][6]. An overflow of time and cost usually occurs in software project development activities, which often forces to cut the development costs at the cost of software quality. This overflow can impose budget deficit, lack of human force, delayed planning, low-quality software, and eventually project failure [1][7].

Project estimation is an essential part of completing a project. Projects are planned in terms of cost, effort, and budget at the beginning phase of development [8]. Precise effort estimation of software

development plays a main task to predict how much workforce should be prepared during the works of a software project to be completed on time and with the planned budget without ignoring the quality of a software [9]. Accuracy of development cost estimation is a key factor in the success of a construction project and influenced the decision-making by the stakeholders of a software project [10] and to bid a contract with them [11]. The capacity of a budget estimating model is determined by calculating its bias, stability, and precision. Measures of bias, stability, and precision are concerned with the average difference between actual and the estimated costs, considering both the degree of variation around the average and the combination with bias and consistency [12]. By far, the most popular evaluation criteria used involve statistics such as mean, standard deviation, and coefficient of variation [10].

Identifying and calculating software metrics are important for various reasons, including estimating programming execution, measuring the effectiveness of software processes, estimating required efforts for processes, reducing defects during software development, and monitoring and controlling software project executions [13]. An example of the wrong cost estimation that happened recently was in estimating the budget of the international arrivals facility built at Seattle-Tacoma International Airport in Seattle, Washington, USA. Initially, in 2013 the budget was estimated at US$ 300 million, but then the budget increased up to US$ 968 million in September 2018 [14]. Research shows that usually, projects seem to be unclear at the beginning and become less vague as they progress [15].

One of the software metrics used to estimate the cost and effort is called the lines of code (LOC) metric and is considered a basic software metric [16] as it is used in most software project estimation techniques. It is hard to quickly and accurately predict the development budget at the planning stage because the documentation is generally incomplete. For this reason, various procedures have been created to accurately predict construction costs with the limited project data available in the early phase [3]. Three known models are used to estimate the project effort, cost, and resources: Constructive Cost Model (COCOMO), Analogy-based Model, and Use Case Points model.

Banimustafa [11] has used three machine learning techniques; Naïve Bayes, Logistic Regression, and Random Forest on the COCOMO dataset. The software project estimation methods used are COCOMO, function point analysis, and use case point. All the machine learning techniques used successfully predicted most of the project effort even better than COCOMO. The researcher highlighted that in future work to improve the drivers of COCOMO. Rekha [17] studied the comparative analysis between traditional techniques and Machine Learning (ML) techniques that successfully found the suitable technique for a kind of project. The finding highlights that many other researchers develop different estimation methods, but there is no best method because each technique can suit particular projects. Bhatia and Attri [18] compared two Machine Learning algorithms, Linear Regression and Multi-Layer perceptron, to predict the effort estimation of a software project using the COCOMO dataset. The comparison was made based on the evaluation criteria: Correlation Coefficient, Mean Absolute error, Root Mean Square Error, Relative absolute Error, and Root Relative Squared Error. The research proved that the Multi-Layer perceptron model could predict the effort of a software project at the beginning stage more accurately than the linear regression model.

This paper presents the use of Particle Swarm Optimization (PSO) as an optimization algorithm to optimize the parameters of the COCOMO model to estimate a more practical and accurate effort. The remainder of the paper is organized as follows: Section 2 describes the methodological steps of work used in this experimentation and presents the evaluation criteria and the dataset used. Section 3 discusses the experiment and results findings from the comparison. Finally, Section 4 concludes the testing and experimentation with the findings.

2. Method

This research paper aims to develop an optimized software project cost and effort estimation of the COCOMO model with Particle Swarm Optimization (PSO). The research starts with a literature review study to gain insight into previous researchers' detailed problem statement and approaches (Fig. 1). A

NASA dataset is collected to demonstrate the performance of optimized Support Vector Machine and Random Forests. The dataset can be assessed publicly from http://promise.site.uottawa.ca/serepository/datasets-page.html. Sources of the datasets are from COCOMO NASA 1 that contains 17 attributes and 60 number of projects.

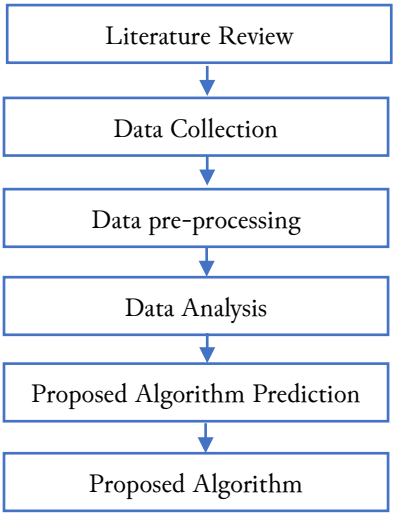

Fig. 1. Research Methodology

The application will require the user to insert five inputs about a project, which are the number of Lines of Code (LOC), Database Size (DATA), Required Software Reliability (RELY), Execution Time Constraint (TIME), and Main Storage Constraint (STOR). The application's output will estimate the effort needed for that particular project in a person-month unit. The data is pre-processed in order to calculate the effort estimation. In this experiment, the data is imported into r studio. The mice package is used to check the missing values, and the datasets contain no missing values. The value of the drivers is in numerical weight converted to numerical values due to avoid bias while constructing the machine learning model. The mode constant is assigned based on the COCOMO predefined values. Three traditional regression machine learning algorithms are used for this experiment: Linear Regression, Support Vector Machine, and Random Forest.

The linear regression model summarizes a relationship between two variables, independent and dependent variables. In this experiment, the practical use of linear regression is to find the approximate prediction as a predictive model. The relationship of the prediction and the actuals data is then observed from the best fit line. The best fit line is where the total error prediction is as small as possible. Support Vector Machine (SVM) model is a linear model for classification and regression problems. A support vector machine model can solve linear and non-linear problems. This model aims to create a hyperplane and separate the data into classes. The support vector machine model can find the maximum margin between the data points and the hyperplane to reduce misclassifications. Also, it can be used to solve unbalanced data problems.

The Random Forest (RF) model is made up of many decision trees that depend on random vector values. This model is called random because, during building trees, it uses random sampling for training data points, and during splitting nodes, it uses random subsets of features considered. Each tree in a random forest learns from a random sample of the data points. The RF is used due to it can produce high accurate classifiers. The most powerful parameters to evaluate the performance and measures the error differences between values are by employing Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE). These parameters are negatively oriented, which implies better lower values. These three criteria provide significantly a meaningful representation that computes an error between two numeric vectors. Mean Squared Error (MSE), the mean squared error or mean squared deviation of an estimator, measures the average of the squares of the errors, the average squared difference between the estimated values, and what is estimated. The lower the value of MSE, the better accuracy.

$$MSE = 1 / n \sum_{j=1^n} (y_j - \hat{y}_j) \quad (1)$$

Root Means Squared Error (RMSE). It represents the sample standard deviation between predicted and observed values (called residuals).

$$RMSE = \sqrt{(1 / n \sum_{j=1^n} (y_j - \hat{y}_j)^2)} \quad (2)$$

Mean Absolute Error (MAE) is the average of the absolute difference between predicted and observed values. The MAE is a linear score which means that all the individual differences are weighted equally in the average.

$$MAE = 1 / n \sum_{j=1^n} |y_j - \hat{y}_j| \quad (3)$$

Mean Absolute Percentage Error (MAPE) measures prediction accuracy as a percentage, also known as the average absolute percent error for each predicted minus actual value and then divided by actual values. The lower MAPE, the better the accuracy.

$$MAPE = 1 / n \sum_{j=1^n} | (y_j - \hat{y}_j) / y_j | \quad (4)$$

The accuracy of the cost estimation models is evaluated by the Magnitude of Relative Error (MRE) and the Mean Magnitude of Relative Error (MMRE). The optimum value of MRE and MMRE is closest to zero.

$$MRE = | (Effort_{estimated} — Effort_{actual}) / Effort_{actual} | \quad (5)$$

Mean Magnitude of Relative Error (MMRE) measures predicted effort and actual effort value relative to the actual effort value.

$$MRE = (| (Effort_{estimated} — Effort_{actual}) / Effort_{actual} |) / N \quad (6)$$

Min-Max Accuracy is a good metric to see how close they are, considering the average between the minimum and the maximum prediction. The higher the value of Min-max accuracy, the better the accuracy. Correlation Accuracy is the correlation between predicted and actuals used as an accuracy measure. The Pearson product-moment correlation coefficient is used to measure the strength of the predicted and actuals value of the experiment. The predicted and actuals value has similar directional movements when the correlation accuracy is high. P-Value, also known as the calculated probability, determines the significance of the experiment's results. The P-Value is lower than 0.05 shows strong proof against the null hypothesis; thus, the null hypothesis is rejected. The smaller the P-Value, the stronger the evidence to reject the null hypothesis.

Null hypothesis of this project is, the population correlation coefficient is not significantly different from zero. There is no significant linear correlation between control and experimental values in the population. The alternative hypothesis of this project is, the population correlation coefficient is significantly different from zero. There is a significant linear relationship between control and experimental values in the population. Vargha and Delaney A (VDA) measure is an example of effect size differentiation between two observations, control and experimental samples. The range of VDA is from 0 to 1. VDA suggested a threshold for interpreting the effect size where 0.5 means no difference at all; up to 0.56 indicates a small difference; up to 0.64 indicates medium; and anything over 0.71 is large [19]. Wilcoxon Rank Sum Test is a nonparametric test used to compare two related samples on a single sample to see if their population ranks differ. The null hypothesis is difference between the two samples has equal medians. The alternative hypothesis is that there is no difference between the two samples. If the p-value is larger than 0.05, we must accept the null hypothesis because there is enough evidence to conclude. The null hypothesis is rejected; there is sufficient evidence to conclude the sample

has no identical distributions. The training dataset is 80%, and the testing dataset is 20% for COCOMO NASA 1.

2.1. Software Effort Estimation

Software project management demands an accurate software effort estimation that provides sufficient support to judge the amount of effort and resources efficiently and effectively [20][21]. Software effort estimation is the process of predicting the most realistic amount of effort required to develop or maintain software based on incomplete, uncertain, and noisy data [22]. Software is the most expensive component in many computer-based systems that affects a number of bugs that produce enormous differences between gain and loss during effort estimation [23].

A project manager often faces many problems with estimating the effort needed when developing a project. An estimating effort task is a great challenge for any software company, especially to develop a new and high-quality software project within a predetermined budget and time [24]. Although estimating an effort can be measured with numerous techniques, some of the techniques used require additional data, while the others are time-consuming and difficult to follow [25]. Besides, there are several drawbacks in effort estimation despite having made an initial estimation. Previously, most project managers estimate their brand-new project blindly based on their experience-based judgment and estimation by analogy, which is often unsuitable and incompatible with the project. Therefore, using optimization algorithm approaches is highly advantageous to use a more structured estimation process [26]–[28]. This paper aims to objectively estimate software effort by using optimization algorithm approaches despite using subjective and time-consuming estimation methods like expert judgment and estimation by analogy [3][29].

2.2. COCOMO Model

The constructive Cost Model (COCOMO) is the most commonly used model for effort estimation. It was proposed by Barry Boehm and is based on the study of 63 projects, making it one of the best-documented models [30]. Although COCOMO is among the most common, COCOMO seems to become less accurate due to increased complexity and over-demanding software requirements [23][31]. COCOMO model uses a regression formula based on a number of lines of code (LOC). This model is often used to reliably predict the various parameters associated with making a project, such as size, effort, cost, time, and quality [21][32].

Effort and schedule are two parameters and outcomes that define the quality of any software product. An effort is the amount of labor required to complete a task, measured in person-month units [12]. Schedule simple means the amount of time required to complete the job, which is proportional to the effort. It is measured in the units of time such as weeks, months. Therefore, this paper aims to improve the accuracy of COCOMO effort estimation by optimizing the coefficients by using Particle Swarm Optimization (PSO) for assessing the performance. The optimized COCOMO with PSO is evaluated based on error, minimum, maximum, correlation, and statistical significance tests.

COCOMO (Constructive Cost Model) is a screen-oriented, interactive software package that assists in budgetary planning and schedule estimation of a software project [30]. The intermediate COCOMO model used 15 drivers to estimate the cost of a project. The drivers are classified into four attributes; Product attributes, Hardware attributes, Personnel attributes, and Project attributes [11].

2.3. Particle Swarm Optimization (PSO)

Particle swarm optimization is one of the most popular swarm intelligence algorithms inspired by the social behavior of bird flocking or fish schooling. These swarms comply with cooperative food, and each member of the swarms continues to change the search pattern on its own and other members according to their learning experiences.

The straightforward behavior followed by individuals in a flock imitates their own successes and the success of neighboring individuals. In a PSO algorithm, a population is called the swarm, the candidate of solutions in the swarm is called particles, while the food is called an objective function [33]. Particles can update their positions and velocities according to environmental change. In addition, the swarm in

PSO does not limit its movement but continuously searches for the optimal solution in the possible solution space [12][28]. Particles in PSO can keep their stable movement in the search space while change their movement mode to adapt to the change in the environment [34].

Each particle representing a potential solution is maintained within a swarm. In simple terms, the particles are, therefore, 'flown' through a multidimensional search space where the position of each particle is adjusted according to itself and its neighbor's experiences [35]. Let $xi(t)$ denote the position of particle i in the search space at time-space t, which denotes discrete time steps unless otherwise stated. The position of the particle is changed by adding a velocity vector, $vi(t)$, to the current position:

$$x_i\ (t+1) = x_i\ (t) + v_i\ (t+1) \tag{7}$$

Each particle also has to maintain its P*best*, the personal local best position, and G*best*, the global best position among all particles [36]. Following equations are used to update the position and velocity of the particle.

$$v_i\ (t+1)\ =\ v_i\ (t)\ +\ c_1 r_1\ (P_{best} —\ x_i\ (t))\ +\ c_2 r_2\ (G_{best} —\ x_i\ (t)) \tag{8}$$

where r_1 and r_2 are two random numbers range [0,1) and c_1 and c_2 are the learning factors. The PSO algorithm is shown in Fig. 2.

Algorithm 2 Particle Swarm Optimization (PSO)
1. Initialize the parameter NP, w, c_1, and c_2, maximum iteration
2. Initialize particle velocity and position
3. Evaluate the fitness value
4. If (the current fitness value < particle best value)
5. Assign the current value for P*best*
6. End if
7. Set particle with the best fitness value to G*best*
8. Iteration = 1
9. While stopping criteria is not reached
10. Do
11. For i = 1 to NP
12. Update particle velocity according to the second equation
13. Update particle position according to the first equation
14. End for
15. End while
16. Return the best solution

Fig. 2. The PSO algorithm

3. Results and Discussion

In this project, the correlation matrix uses to evaluate the correlation of the two variables. The dependent variable is the actual effort attribute, while the 15 cost drivers and the line code are the independent variables. From Fig. 3, five attributes positively correlate towards actual effort attributes, LOC, DATA, TIME, TOOL, and STOR for COCOMO NASA 1 dataset. In comparison, other attributes show negative correlations towards actual effort attributes.

The project builds predictive machine learning models with COCOMO NASA 1, using all attributes. The predictive machine learning models are Support Vector Machine, Linear Regression, and Random forest. There are two additional optimized algorithms with Support Vector Machine and Random Forest using Particle Swarm Optimization (PSO). The performance of PSO Random Forest (PSORF) and PSO Support Vector Machine (PSOSVM) are analyzed by comparing all results from error accuracy, minimum, maximum accuracy, correlation accuracy, and statistical accuracy; with basic machine learning algorithms such as Support Vector Machine, Linear Regression, and Random Forest. The project evaluates the result and records it in the following table.

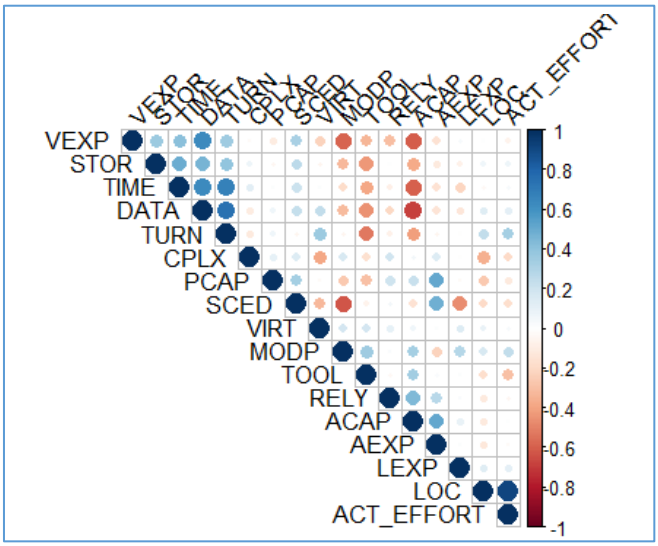

Fig. 3. Correlation of COCOMO NASA 1 Attributes

Table 1 explains the experiment made on COCOMO NASA 1 with a training dataset of 80 percent and the testing dataset of 20 percent. Based on Table 1, all five proposed algorithms, Support Vector Machine, Linear Regression, Random Forest, PSOSVM, and PSORF, are significant due to the p-value less than 0.005. The best error accuracy for the MAE value among the five models is the PSOSVM model, the lowest, 0.88737150. The second-best MAE is the PSORF model with 1.042509902, followed by Support Vector Machine and Random Forest with MAE 31.5403163 and 36.8215429, respectively. The worst MAE with the highest value is Linear Regression with 47.7723733.

Table 1. Comparison of algorithms for COCOMO NASA 1 Dataset

Algorithm	Support Vector Machine	Linear Regression	Random Forest	PSORF	PSOSVM
MAE	31.5403163	47.7723733	36.8215429	1.042509902	0.88737150
MSE	2755.54974	5078.17341	2421.11237	1.75629437	1.07215260
MAPE	0.2901647	0.4632032	0.4032879	0.01249371	0.01126849
RMSE	52.4933209	71.2613038	49.2048003	1.32525257	1.03544802
Min-Max Accuracy	0.8019366	0.5631169	0.7328254	0.6957113	0.6545856
Correlation Accuracy	0.9561591	0.9561002	0.9400819	0.9999501	0.9999951
P-value	3.67e-05	1.193e-06	5.498e-06	2.2e-16	2.2e-16
Significant Value	Significant	Significant	Significant	Significant	Significant

The best MSE among the five models is PSOSV, which has the lowest MSE, 1.07215260. Next, the PSORF model was found as the second-lowest with an MSE value of 1.75629437. Then, followed by Random Forest with MSE, 2421.1123716, the highest error accuracy for MSE is Linear Regression with MSE, 5068.1734135. The best MAPE among the five models is PSOSVM, with values of 0.01126849. The second-lowest MAPE is PSORF with 0.01249371. Then, the third-lowest in the Support Vector model with MAPE, 0.2901647. Linear Regression demonstrates the highest value of MAPE with 0.5614929. The best RMSE among the five models is PSOSVM, with values of 1.03544802. The second best RMSE is PSORF with 1.32525257, followed by Random Forest and Support Vector Machine model with RMSE of 49.2048003 and 52.4933209, respectively. Lastly, Linear Regression shows the worst RMSE with a value of 71.2613038.

The perfect min-max accuracy is Support Vector Machine with a value of 0.8019366 near 1. Next is the Random Forest model, with a value of 0.732854. PSOSVM and PSORF demonstrate min-max accuracy values of 0.6957113 and 0.6545856, respectively. The lowest value of min-max accuracy is Linear Regression, 0.5631169. The highest value of Correlation accuracy is PSOSVM, 0.9999951. The second

highest is followed closely by PSORF with a value of 0.9999501. The correlation accuracy for Support Vector Machine is 0.9561591, and for Linear Regression is 0.9561002. The lowest value of correlation accuracy is Random Forest, 0.9400819. As Table 1 indicates, Random Forest and Support Vector Machine algorithms are chosen to be further investigated with a metaheuristics optimization algorithm, Particle Swarm Optimization (PSO). This is considering that the result illustrates in Table 1 where linear regression consistently falls behind between Random Forest and Support Vector Machine. PSORF and PSOSVM are two optimized algorithms proposed to reach the level of satisfaction of estimation models. The results conclude that an extensive experiment (PSOSVM and PSORF) improves the accuracy made from the proposed traditional methods. Another promising finding from Table 1, PSOSVM outperformed almost all performance metrics except for Min-Max accuracy. The performance of PSOSVM and PSORF is further measured based on their comparison with the traditional methods. Table 2 illustrates the performance metrics used to assess all three proposed algorithms using three attributes, including all attributes, five selected attributes, and one attribute (LOC only).

Table 2. Comparison of COCOMO NASA 1 Dataset with Different Attributes

	Algorithm	Correlation Accuracy	P-value	Significant Value	MMRE	A Measure	Rank-Sum	Significant Rank Sum
All Attributes	SVM	0.8157183	0.00122	Significant	127.50	0.45138	0.7074	Significant
	Random Forest	0.6622912	0.01895	Significant	110.70	0.40972	0.4704	Significant
	PSORF	0.9999501	2.2e-16	Significant	1.2493	0.47222	0.8398	Significant
	PSOSVM	**0.9999951**	2.2e-16	Significant	1.1268	0.47917	0.8852	Significant
5 Attributes	SVM	0.7636789	0.00384	Significant	109.68	0.42361	0.5443	Significant
	Random Forest	0.8642062	0.00288	Significant	108.41	0.42361	0.5443	Significant
	PSORF	**0.9999768**	2.2e-16	Significant	**0.8860**	0.53472	0.7949	Significant
	PSOSVM	**0.9999958**	2.2e-16	Significant	**0.7756**	0.50694	0.977	Significant
LOC Only	SVM	0.7772104	0.00293	Significant	66.986	0.45833	0.7508	Significant
	Random Forest	0.8552052	0.00039	Significant	67.671	0.48611	0.931	Significant
	PSORF	0.9999691	2.2e-16	Significant	1.4920	0.5	1	Significant
	PSOSVM	**0.9999840**	2.2e-16	Significant	0.9382	0.52778	0.8398	Significant

The result highlights that between the trained algorithms for all attributes and five selected attributes, the five selected attributes demonstrate performance better in producing more accurate results. The correlation accuracy of the five selected attributes has a higher relationship between the actual effort and the predicted effort values than all attributes. Besides, the MMRE of five selected attributes has lower values than all attributes, and this shows that five selected attributes have a more accurate estimation between the actual and predicted effort values. A Measure shows only slight differences between all attributes and the five selected attributes. Support Vector Machine with all attributes has no effect differences compared to Random Forest model. The rank-sum of all attributes and five selected attributes are statistically significant; there is enough evidence to support a null hypothesis and reject the alternative hypothesis.

The result of this analysis is then compared with an experiment using only one attribute, LOC. The correlation accuracy of using LOC is an increase compared to five attributes and all attributes. The p-value of Pearson's correlation also shows that the models are statistically significant. The MMRE illustrates in the LOC table has better results than using all attributes and five selected attributes. The Vargha and Delaney A measure for LOC also shows no difference, which means the distribution of actual and predicted effort values are identical. The Wilcoxon rank-sum test is statistically significant where there is enough evidence to support a null hypothesis and reject the alternative hypothesis since the p-value of both machine learning models is higher than 5 percent.

Another analysis that can be deduced from Table 2 is that among three different experiments with attributes, PSOSVM continuously outperformed other trained models such as traditional Random Forest and Support Vector Machine and PSORF: for all the performance metrics. However, there are

two notable exceptions here, where PSORF outperformed A measures and Wilcoxon Rank Sum test for LOC attributes experiment.

All these analyses conclude that not all attributes are much needed to be trained by the trained algorithms. From the experiment, using one attribute, LOC, can have closer MMRE towards the COCOMO prediction model, higher correlation accuracy, and identical distribution of actual and predicted effort values. The results also prove that optimizing COCOMO models with Particle Swarm Optimization illustrates significantly better results due to increasing accuracy rates while reducing the error accuracy and statistically significant, which implies the differences between actual effort and predicts effort are very small. Overall, the PSOSVM yields increasingly good results for the estimation software model, COCOMO.

4. Conclusion

Many existing machine learning algorithms can train predictive models; however, the right and suitable machine learning model is needed to estimate accurately. In this research, the five selected attributes with high positive correlation toward actual effort attribute are obtained from the correlation matrix, DATA, STOR, LOC, TIME, and TOOL. The five important attributes give better results compared to using all the attributes in COCOMO dataset. Hence, not all attributes in the dataset are relevant to be used for estimating software development. In this paper, we analyzed the efficacy of applying metaheuristics swarm intelligence, Particle Swarm Optimization (PSO), as an optimization algorithm, by optimizing its parameters to increase the accuracy level of the COCOMO model. An optimized Support Vector Machine and Random Forest algorithms (PSOSVM and PSORF) impressively result with the COCOMO NASA dataset. However, among these two optimized PSO-based algorithms, PSOSVM outperformed all evaluation criteria when compared with all attributes. To conclude, optimizing COCOMO parameters with the PSO method provides an improved estimate compared to the conventional COCOMO model.

Acknowledgment

The authors thank International Islamic University Malaysia (IIUM) Research Acculturation Grant Scheme (IRAGS) that supports this research.

Declarations

Author contribution. All authors contributed equally to the main contributor to this paper. All authors read and approved the final paper.
Funding statement. This research was supported by the IIUM Research Acculturation Grant Scheme (IRAGS), with grant number IRAGS18-012-0013.
Conflict of interest. The authors declare no conflict of interest.
Additional information. No additional information is available for this paper.

References

[1] S. Sabbagh Jafari and F. Ziaaddini, "Optimization of software cost estimation using harmony search algorithm," *1st Conf. Swarm Intell. Evol. Comput. CSIEC 2016 - Proc.*, pp. 131–135, 2016, doi: 10.1109/CSIEC.2016.7482119.

[2] S. Chhabra and H. Singh, "Optimizing Design of Fuzzy Model for Software Cost Estimation Using Particle Swarm Optimization Algorithm," *Int. J. Comput. Intell. Appl.*, vol. 19, no. 1, pp. 1–16, 2020, doi: 10.1142/S1469026820500054.

[3] O. Hidmi and B. E. Sakar, "Software Development Effort Estimation Using Ensemble Machine Learning," *Int. J. Comput. Commun. Instrum. Eng.*, vol. 4, no. 1, 2017, doi: 10.15242/ijccie.e0317026.

[4] A. Kumar, B. D. . Patro, and B. K. Singh, "Parameter Tuning for Software Effort Estimation Using Particle Swarm Optimization Algorithm," *Int. J. Appl. Eng. Res.*, vol. 14, no. 2, pp. 139–144, 2019. Available at: Google Scholar.

[5] V. Khatibi Bardsiri, D. N. A. Jawawi, S. Z. M. Hashim, and E. Khatibi, "A PSO-based model to increase the accuracy of software development effort estimation," *Softw. Qual. J.*, vol. 21, no. 3, pp. 501–526, 2013, doi: 10.1007/s11219-012-9183-x.

[6] S. K. Sehra, Y. S. Brar, N. Kaur, and G. Kaur, "Optimization of COCOMO Parameters using TLBO Algorithm," *Int. J. Comput. Intell. Res.*, vol. 13, no. 4, pp. 525–535, 2017. Available at: Google Scholar.

[7] S. Basha and D. Ponnurangam, "Analysis of Empirical Software Effort Estimation Models," *Int. J. Comput. Sci. Inf. Secur.*, vol. 7, no. 3, pp. 68–77, 2010. Availabel at: Google Scholar.

[8] S. Hajar Arbain, N. Azizah Ali, and N. Haszlinna Mustaffa, "Adoption of Machine Learning Techniques in Software Effort Estimation: An Overview," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 551, no. 1, 2019, doi: 10.1088/1757-899X/551/1/012074.

[9] S. Ardiansyah, A., Mardhia, M. M., & Handayaningsih, "Analogy-based model for software project effort estimation," *Int. J. Adv. Intell. Informatics*, vol. 4, no. 3, pp. 251–260, 2018, doi: 10.26555/ijain.v4i3.266.

[10] G.-H. Cho, H.-G., Kim, K.-G., Kim, J.-Y., & Kim, "A Comparison of Construction Cost Estimation Using Multiple Regression Analysis and Neural Network in Elementary School Project.," *J. Korea Inst. Build. Constr.*, vol. 13, no. 1, pp. 66–74, 2013, doi: 10.5345/jkibc.2013.13.1.066.

[11] A. Banimustafa, "Predicting Software Effort Estimation Using Machine Learning Techniques," *8th Int. Conf. Comput. Sci. Inf. Technol. CSIT 2018, (October),* pp. 249–256, 2018, doi: 10.1109/CSIT.2018.8486222.

[12] K. Langsari and R. Sarno, "Optimizing effort and time parameters of COCOMO II estimation using fuzzy multi-objective PSO," *Int. Conf. Electr. Eng. Comput. Sci. Informatics*, vol. 2017-Decem, no. September, pp. 19–21, 2017, doi: 10.1109/EECSI.2017.8239157.

[13] R. Kalaivani, N., & Beena, "Overview of Software Defect Prediction Using Machine Learning Algorithms," *Int. J. Pure Appl. Math.*, vol. 118, pp. 3863–3873, 2018. Available at: Google Scholar.

[14] J. Thomas, "The Science of Uncertainty: Blown Budgets and Destroyed Schedules. Sometimes, It's Weak Project Estimation That's to Blame," pp. 56–61, 2019. Available at: https://www.pmi.org/

[15] R. M. H. Ghatasheh, N., Faris, H., Aljarah, I., & Al-Sayyed, "Optimizing Software Effort Estimation Models Using Firefly Algorithm," *J. Softw. Eng. Appl.*, vol. 08, no. 03, pp. 133–142, 2015, doi: 10.4236/jsea.2015.83014.

[16] M. Z. Alsaeedi, A., & Khan, "Software Defect Prediction Using Supervised Machine Learning and Ensemble Techniques: A Comparative Study," *J. Softw. Eng. Appl.*, vol. 12, no. 05, pp. 85–100, 2019, doi: 10.4236/jsea.2019.125007.

[17] Rekha, "Effort Estimation Using ML Models," vol. 6, no. 1, pp. 1–2, 2017.

[18] V. K. Bhatia, S., & Attri, "Machine Learning Techniques in Software Effort Estimation Using COCOMO Dataset," vol. 2, no. 6, pp. 101–106, 2015. Available at: Google Scholar.

[19] H. D. Delaney and A. Vargha, "A Critique and Improvement of the CL Common Language Effect Size Statistics of McGraw and Wong," *J. Educ. Behav. Stat.*, vol. 25, no. 2, pp. 101–132, 2000. doi: 10.3102/10769986025002101.

[20] M. Parwita, R. Sarno, and A. Puspaningrum, "Optimization of COCOMO II Coefficients using Cuckoo Optimization Algorithm to Improve The Accuracy of Effort Estimation," *Int. Conf. Inf. Commun. Technol. Syst.*, pp. 99–104, 2017. doi: 10.1109/ICTS.2017.8265653

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

[22] A. Baghe, M. Rathod, and P. Singh, "Software Effort Estimation using parameter tuned Models," 2020. Available at: Google Scholar.

[23] C. E. Carbonera, K. Farias, and V. Bischoff, "Software development effort estimation: A systematic mapping study," *IET Softw.*, vol. 14, no. 4, pp. 328–344, 2020, doi: 10.1049/iet-sen.2018.5334.

[24] R. Saljoughinejad and V. Khatibi, "A new optimized hybrid model based on COCOMO to increase the accuracy of software cost estimation," *J. Adv. Comput. Eng. Technol.*, vol. 4, no. 1, pp. 27–40, 2018. Available at: Google Scholar.

[25] L. Radlinski and W. Hoffmann, "On Predicting Software Development Effort using Machine Learning Techniques and Local Data," *Int. J. Softw. Eng. Comput.*, vol. 2, no. 2, pp. 123–136, 2010. Available at: Google Scholar.

[26] F. Nayebi, A. Abran, and J.-M. Desharnais, "Automated selection of a software effort estimation model based on accuracy and uncertainty," *Artif. Intell. Res.*, vol. 4, no. 2, 2015, doi: 10.5430/air.v4n2p45.

[27] 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. 08, no. 03, pp. 133–142, 2015, doi: 10.4236/jsea.2015.83014.

[28] R. K. Sachan *et al.*, "Optimizing Basic COCOMO Model Using Simplified Genetic Algorithm," *Procedia Comput. Sci.*, vol. 89, pp. 492–498, 2016, doi: 10.1016/j.procs.2016.06.107.

[29] A. Khatoon and R. Kaur, "Optimization Estimation Parameters of COCOMO Model II Through Genetic Algorithm," *Int. J. Comput. Sci. Eng.*, vol. 6, no. 5, pp. 221–226, 2018, doi: 10.26438/ijcse/v6i5.221226.

[30] B. W. Boehm *et al.*, *Software Cost Estimation with COCOMO II*. Upper Saddle River, NJ: Prentice Hall, 2000. Available at: Google Scholar.

[31] I. C. Suherman, R. Sarno, and Sholiq, "Implementation of random forest regression for COCOMO II effort estimation," *Proc. - 2020 Int. Semin. Appl. Technol. Inf. Commun. IT Challenges Sustain. Scalability, Secur. Age Digit. Disruption, iSemantic 2020*, pp. 476–481, 2020, doi: 10.1109/iSemantic50169.2020.9234269.

[32] S. Asija, "Software Engineering | COCOMO Model," *Geeks for Geeks*, 2017.

[33] N. A. Samat *et al.*, "A Study of Data Imputation Using Fuzzy C-Means with Particle Swarm Optimization," *Recent Adv. Soft Comput. Data Min.*, vol. 549, no. January, 2017, doi: 10.1007/978-3-319-51281-5.

[34] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm : an overview," *Soft Comput.*, vol. 22, no. 2, pp. 387–408, 2018, doi: 10.1007/s00500-016-2474-6.

[35] J. Mercieca and S. G. Fabri, "A Metaheuristic Particle Swarm Optimization Approach to Nonlinear Model Predictive Control," *Int. J. Adv. Intell. Syst.*, vol. 5, no. 3, pp. 357–369, 2012. Available at: Google Scholar.

[36] M. Imran, R. Hashim, N. Elaiza, A. Khalid, and H. Onn, "An Overview of Particle Swarm Optimization Variants," *Procedia Eng.*, vol. 53, no. 1, pp. 491–496, 2013, doi: 10.1016/j.proeng.2013.02.063.