OLCE: Optimized Learning-based Cost Estimation for Global Software Projects

K Nitalaksheswara Rao (kolukulanitla@gmail.com)
GITAM University
Jhansi Vazram Bolla
narasaraopeta engineering college
Satyanarayana Mummana
Raghu Engineering college
CH. V. Murali Krishna
NRIIT
O. Gandhi
Vignan's Foundation for Science, Technology & Research
M James Stephen
WISTM

Research Article

Keywords: Artificial Neural Network, COCOMO, Learning-based model, Prediction, Risk, Software Cost Estimation, Software Project, Software Engineering

Posted Date: September 6th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-2024296/v1

Additional Declarations: No competing interests reported.

OLCE: Optimized Learning-based Cost Estimation for Global Software Projects

K Nitalaksheswara Rao[1] Jhansi Vazram Bolla[2] Satyanarayana Mummana[3] CH.V. Murali Krishna[4] O.Gandhi [5] M James Stephen[6]

Assistant Professor Department of CSE School of Technology GITAM University Visakhapatnam India. kolukulanitla@gmail.com[1]

Professor Department of CSE Narasaraopeta Engineering College Guntur India. jhansi.bolla@gmail.com[2]

Associate Professor Department of CSE Raghu Engineering College Visakhapatnam India. Satyam369@gmail.com[3]

Associate Professor Department of CSE NRIIT Vijayawada India. muralikrishna_chinta2007@yahoo.co.in [4]

Assistant Professor Department of CSE Vignan University Guntur India. ongolegandhi@gmail.com[5]

Professor and Principal WISTM College Visakhapatnam India. jamesstephenm@yahoo.com[6]

Abstract

Software Cost Estimation (SCE) is an integral part of pre-development stage of software project with a target to accomplish a better visibility towards possible risk while gaining more information towards reaching success rate to meet the deadline of delivery. Irrespective of multiple research contribution model towards SCE, the problem and challenges towards accurate cost estimation in presence of dynamicity and uncertainty is yet not reported to be accomplished. Apart from this, learning-based models are slowly gaining pace in almost every field and yet it is still in nascent stage of progress in software engineering. Therefore, the proposed manuscript introduces Optimized Learning-based Cost Estimation (OLCE) which is a novel learning-based model capable of accurate prediction considering global and large scale software project. The proposed system harnesses the learning potential from artificial neural network integrated with novel search-based approach for optimizing the learning method considering the benchmarked COCOMO NASA 2 dataset. The study outcome shows OLCE offers 50% faster response time with approximately 73% of accuracy compared to existing models that are reportedly found to be

adopted for SCE. Hence, OLCE is found to offer a balance between accuracy and computational efficiency during SCE.

Keywords: Artificial Neural Network, COCOMO, Learning-based model, Prediction, Risk, Software Cost Estimation, Software Project, Software Engineering.

1. Introduction

Software Cost Estimation (SCE) is one of the most essential involved process in software engineering which can perform predictive analysis of cumulative effort required to construct a new software project [1]. In this process, the cost factor is associated with all the human, computational, and organizational resources being involved towards developing and dispatching the final result of software project at a given time bound [2][3]. The standard techniques involved in cost estimation are cosmic [4], user-point story [5], COCOMO [6], functional point [7], Line of Code [8], etc. Out of all existing standard SCE, COCOMO is highly preferred among the community of software engineer [9]. The other standard models are Software Life Cycle Model (SLIM) [10], ESTIMACS [11], SEER-SEM [12]. Different from other industry, computation of software cost is quite a complex process as usually its is accomplished using either inappropriate set of information or incomplete data [13][14]. Hence, majority of the underlying techniques are based on just an assumption construction by the estimator. Some of the critical challenges encountered by an estimators are i) frequent adoption of standardized approach, which may not be applicable for all software projects, ii) issue in collaboration with the team and predicting their vitals, which are quite impractical and stochastic in nature, iii) identifying possible risk, and iv) insufficient timescale of project delivery [15]-[20]. A quick look into conventional research contribution highlights that adoption of artificial intelligence-based scheme is constantly on rise to solve all the critical problems [21]. This trend of adoption is more towards other computational modelling and less towards software engineering and hence this adoption is slowly increasing in its pace to explore possibilities. This inclination towards adoption of learning-based approaches derived from artificial intelligence is quite essential as majority of the conventional SCE schemes are already reported with the pitfalls [22]-[25]. However, adoption of artificial intelligence is too shrouded with challenge associated with reliability of the accomplished predictive accuracy score. Further,

various techniques of artificial intelligence itself has beneficial and limiting characteristics and yet this adoption is quite in nascent stage of development. Therefore, the proposed scheme introduces a novel learning-based SCE method which is meant for overcoming the estimation loopholes in existing scheme as well as to address the computational efficiency while deploying learning schemes. The novelty as well as contribution of proposed scheme are as follows:

i. The proposed scheme introduces a computational framework of SCE which allows an estimator to perform a simplified predictive cost estimation for their target software project.
ii. The study model develops a predictive model using artificial neural network and search-based optimization discretely towards building framework of learning and optimization.
iii. The involved learning scheme is capable of considering multiple factors towards representation followed by indexing and optimized adaptation of learning feature weight
iv. The complete analysis is carried out over benchmarked dataset considering standard performance metric to showcase a better computational efficiency and accuracy in contrast to existing models.

The organization of the paper is as follows: Section 2 briefly discusses about the essentials of Software Cost Estimation with respect to taxonomy and challenges in existing methods, Section 3 discusses about related work, Section 4 discusses about problem identification, Section 5 discusses about research methodology, while elaborated discussion of proposed learning model is carried out in Section 6. Section 7 discusses about result analysis while conclusive remarks of paper is given in Section 8.

2. Software Cost Estimation

The core target of Software Cost Estimation (SCE) is to perform a predictive assessment of the cost involved in the development of software projects even before initiating the phases of development. The standard term *cost* is usually represented in the form of resources engaged in the development, time consumed in development, and all cumulative approximated effort to accomplish the complete development stage. A typical cost estimation practice is shown in Fig.1, where the project manager defines the target software cost. At the same time, this process is followed up by manipulation of various attributes and sizes until the cost of the target software is found justified.

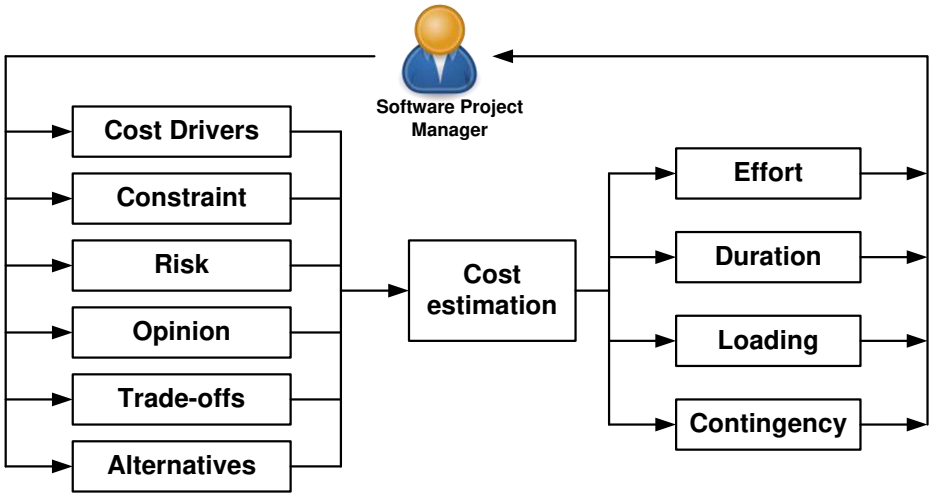


Figure 1 Typical Process of Software Cost Estimation

The process exhibited in Fig.1 is highly essential to perform analysis and forecasting of all possible risks involved in costing and realize various trade-offs and sensitivities associated with it. This process lets the project manager scrutinize the software project and filter out the evaluated risk possibilities to accomplish the cost of the target software. It should be noted that the process mentioned above of software cost estimation is carried out by the project manager and equally contributed by different teams working for testing, development, and architecture. Further, it can be seen from Fig.1 that there are varied inputs for cost estimation, viz. cost driver, constraints, risk, opinion, trade-off, and alternatives. On the other hand, the outcome of the cost estimation is assessed concerning effort, duration, loading, and contingency. Hence, it can be seen that multiple attributes are involved in software cost estimation. This section further briefs on different taxonomies and challenges associated with software cost estimation.

2.1 Taxonomies

Currently, various software models are available to enable the process of SCE, e.g., Knowledge Plan, SLIM, ESTIMACS, Checkpoint, COCOMO model, etc. Out of all these models, the community of software engineering is witnessed frequent adoption of the COCOMO model owing to its highly structured estimation techniques for software cost. It was also noted that bottom-up model usage is more frequent compared to the top-down model in software costing. Apart from this frequently adopted SCE model, the standard taxonomy of the SCE is classified into three types:

- *Empirical Cost Estimation Method* [26]: In this cost estimation method, the data required to be predicted is subjected to formulas derived empirically. This estimation technique's prime basis is certain assumptions and data from previous software projects of similar form. The estimation of an effort is carried out on the basis of the size of the software to be used. Although this technique entirely runs on various assumptions, it is still formalized by industries. An example of such a cost estimation technique would depend on numerous approaches for formalizing, e.g., Expert judgment and the Delphi method.
- *Heuristic Method of Cost Estimation* [27]: The prime basis of this cost estimation technique is related to the process of discovery. This model is used for discovery from practical approaches for accomplishing goals and a plan of learning and solving the problem. Apart from this, the model also facilitates higher flexibility and various enriched calculations and shortcuts for making faster decisions. However, these traits are applicable for only simplified forms of data; however, the technique demands a higher degree of optimal decision for this purpose. Mathematical expressions are used for constructing relationships among multiple parameters of software projects. One robust example of this technique is the Constructive Cost Model, also known as COCOMO, deployed for accelerating the decision-making and analytical speed.
- *Analytical Cost Estimation Method* [28]*:* This cost estimation technique is used for measuring the task initially decomposed to smaller operational components for better analysis. In this case, if the availability of the standard time is ascertained from specific sources, then they are applied to all components of the task. In case of unavailability of such time, work estimation is carried out based on professional work experience. This method uses fundamental assumptions to derive the outcome of software projects. One example of this technique is Halstead's software.

Apart from the above-mentioned standard taxonomies, various scientific kinds of literature offer more insight into the taxonomies of SCE [29][30]. According to such studies, the SCE techniques are further classified into algorithmic, non-algorithmic, and learning-based methods. Examples of Algorithmic methods in SCE are the SLIM/Putnam Model, Function Point Analysis, and COCOMO. Methods used for non-algorithmic-based approaches are analogy-based, price to win, bottom-up estimation, top-down estimation, and expert judgment. Learning-based techniques adopted in SCE are Regression tree, support vector regression, Bayesian network, fuzzy logic,

genetic algorithm, and artificial neural network. The following section outlines the challenges in SCE methods.

2.2 Challenges of Existing Methods

Irrespective of the availability of the standard techniques of SCE, as briefed in the prior section, it is noted that various challenges are yet to be addressed while using them. Some of the critical challenges associated with existing methods are as follows:

- *Cost Quantification*: It is quite challenging to correctly identify a quantified cost by an engineer who would target to minimize the cost by a specific value. Assume that a mathematical relationship to define cost is used using various dependable parameters, e.g., physical characteristics, operation features, and performance. Such a mathematical relationship could be highly sensitive towards a different set of components' weights while the target is to assess software cost with various materials characterized by discrete weights. In such a case, the weight of the mathematical relationship of cost could be non-sensitive towards alternatives on the strength factor. This condition calls for assessing the cost of such alternatives and finding its possible response. There is a need to determine the impact of the cost associated with intangible new approaches or products. Hence, quantifying software cost in the presence of such dynamics is still a challenging task.
- *Undefined Constraints*: There are various evolving constraints during the software development cycle. While some constraints are easier to find out, there are some which are quite sophisticated to trace. One essential constraint is the temporal factor, i.e., time, which introduces a potential influence on the process of cost estimation, aggregation of the data, validation of the evaluation process, and maintaining a higher quality of data with higher consistency. There is always a demand for enriched data by the data analyst that takes considerable resources and consumes time.
- *Data Quality*: The quantity and quality of data are significantly affected by the constraints of resources in software engineering. In case of less time available, the alternative data sources are accessed by an estimator that is amended from the original data source. However, the extent of usefulness associated with secondary data is quite restricted.
- *Massive Organization Involvement*: For a practical cost computation, an estimator must gain complete information that can be derived from the massive organizational structure as

well as multiple sources of data. For this purpose, there is a need for comprehensive accessibility of data for the estimators. The estimator may also require to go through various non-disclosure agreements to cater to the requirements of all the organization. In such a case, time constraint acts as a significant impediment. The quality of the estimation process is highly affected in case of its non-accessibility.

- *Consistency*: For an effective SCE, it is necessary to incorporate consistency as they are aggregated from multiple sources. All the collected data must carry consistencies; however, it is another computationally complex task. With the massive size of data, there are also higher possibilities of inconsistencies and discrepancies. In order to address this problem, a sophisticated database management system is built by investing more significant capital and time capable of handling highly consistent data from multiple sources [31][32]. However, this is in the very nascent stage of development, and more exhaustive frameworks for solving this consistency problem are demanded.

It is to be noted that the challenges mentioned above are potentially associated with the majority of the existing standard SCE techniques where there is no full-proof mechanism yet to control them. However, there are various dedicated attempts among the scientific community to evolve with more solution-based strategies to mitigate the such challenge. The following section discusses existing research contributions toward improving SCE techniques concerning its strength and weakness.

3. Related work

To date, various approaches have been evolving toward improving SCE methods. Multiple use-cases were considered for addressing a specific set of problems solved by adopting different techniques.

The most commonly designed software development applications are in the form of mobile apps in the current era, as noted in the study of Autili et al. [33]. According to this study, mobile application tools and development platforms are quite platform-specific, often posing challenges in existing development. The author has also advocated the *static analysis* method for predicting the performance of such mobile apps. This study's contribution is designing a classification model to assess the static analysis of mobile applications.

Many recent studies have emphasized considering the linkage between allocating human resources and SCE to decide on outsourcing the projects for faster delivery. The solution to this problem is

seen in the work of Chiang and Lin et al. [34], where an integer-based programming methodology has been used for formulating a decision process. The mechanism also introduces a framework for combinatorial optimization used for forming development teams and allocating human resources. From the viewpoint of SCE, effort estimation toward testing operation is highly challenging, especially in defense projects. A study towards this direction has been carried out by Cibir and Ayyildiz [35] that has introduced the adoption of unique software test metrics, viz. quantity of defective scenarios of testing, number of meetings, time to construct a test environment, review period, time to construct test plan, number of methods. The study's outcome assessed over defense industry projects shows that adopting linear regression offers better estimation performance. Apart from this, it is also noted that agile methodology is highly adopted for effort estimation. A notable work by Diego et al. [36] has addressed the problems of SCE associated with cross-company data, which poses a significant challenge towards an effort estimation irrespective of various existing deployable models. The outcome of the study exhibited an increasing range of improvement. A survey of agile methodologies and their associated risk (budget overrun and project failures) is carried out by Lunesu et al. [37]. The presented model considers temporal factors associated with software project duration and completion time using Monte-Carlo simulation.

Further, it was also noticed that improving functional point analysis offers better estimation modeling for handling complexity weight management (Hai et al. [38]). Although this study has discussed various factors in modeling, they are mainly generalized to use cases. This phenomenon calls for more exploration towards identifying the more elaborated number of factors influencing cost estimation (Khan et al. [39][40]). According to this recent work, it was noted that there are non-inclusion of essential cost drivers in conventional SCE techniques, which cannot retain better accuracy standards. Different optimization techniques have also experimented with better SCE methods. Work in this direction has been carried out by Fadhil et al. [41], where the dolphin algorithm is considered for accomplishing higher accuracy in SCE. The estimation is optimized over the COCOMO-II model, where the primary model is subjected to a dolphin algorithm. The second model is subjected to an algorithm combining a bat algorithm and a dolphin algorithm. The outcome exhibited a lower error score than conventional optimization algorithms and models. Another optimization technique was discussed in the work of Nhung et al. [42], where a parametric approach toward effort estimation is applied. This work predominantly focuses on numerous regression models to control the errors in estimation using least squared regression towards all the

elements in points of use cases. The presented algorithm contributes to algorithmic optimization statistically by adopting efficient correction factors.

In the line of exploration towards software engineering techniques adopted in SCE, it was noted that Artificial Intelligence (AI) had played a significant role. The scope of AI is relatively high in software engineering as it can potentially assist in the automation of tedious jobs involved in software development and can also be utilized for exploring data poot of big data (Barenkamp et al. [43]). The outcome of the study concludes that adopting AI could significantly speed up the process of software development and increase efficiency. The utilization of AI has been reported in the existing system concerning its different variants under its standard taxonomies of the learning-based approach. Existing SCE schemes are often said to adopt learning-based techniques to estimate effort. Carvalho et al.[44] have used an extreme learning machine to identify all the essential parameters that potentially influence the SCE technique. The study has used multiple machine learning approaches for this purpose which finally exhibited reliable effort estimation where the outcome showed better estimation performance in comparison to Multi-Layer Perceptron (MLP), Logistic Regression (LR), Support Vector Machine (SVM), K-Nearest Neighboring (KNN). Machine learning is reportedly deployed in SCE to gain control over the excessive consumption of time during the testing cycle, resulting in the identification of a lesser number of bugs present in the program. Chen et al. [45] presented the solution model to this problem. Machine learning has been used to predict the statical coverage of testing operation towards the complier, followed by the clustering scheme for prioritizing the test program. With approximately 68% of the accelerated testing speed, this scheme offers a robust platform for compiler testing. The summarized version of existing approaches towards improving SCE is tabulated in Table 1 with respect to issues being considered, adopted methodology with its dataset/tool, and its associated beneficial and limiting factors.

Table 1 Essential Findings of Existing SCE

Authors	Problems	Methodology	Dataset/Tools	Advantages	Limitation
Autili et al. [33]	Assessing the effectiveness of static analysis in mobile apps	Classification framework	Assessment of 261 studies	A clear understanding of the applicability	Restricted to mobile apps
Chiang and Lin et al. [34]	Allocation of human resource	Integer programming	Data from the case study	Better cost evaluation	It doesn't consider dynamic constraints
Cibir and Ayyildiz [35]	Effort estimation for testing	Linear regression, new software metric	15 projects of different size	Higher scope of new metric towards effort estimation	Study not benchmarked.
Diego et al. [36]	Agile-based effort estimation	Review work	61 research papers	A comprehensive discussion of Agile methods	No disclosure of research gap
Lunesu et al. [37]	Addressing risk in Agile	Monte-Carlo Simulation	JIRA tool	Effective time evaluation	Study yet to benchmark
Hai et al. [38]	Complex weight management	Functional point analysis	ISBSG dataset	Improved accuracy	It doesn't consider local constraints associated with use cases
Khan et al. [39][40]	Higher accuracy in SCE	Statistical Model, Expert	ISBSG dataset	Identified lack of formal models in	Narrowed scope of

		opinion, COCOMO-II		Global software development	empirical assessment
Fadhil et al. [41]	Optimizing SCE	Integrated bat and dolphin algorithm	NASA-60 data	Reduced error score	Involves higher processing time
Nhung et al. [42]	Estimation accuracy	Least Squared Regression, multiple linear regression	Industry dataset	Reduced prediction error	Works on the static timeline of project development
Barenkamp et al. [43]	Assessing the applicability of AI	Theoretical Framework	Assessment of 87 studies	Significant highlights of AI scope	Lack of extensive evaluation of AI
Carvalho et al.[44]	Reliable effort estimation	Extreme machine learning	NASA dataset, COCOMO, ISBSG	Excel better performance compared to MLP, SVM, LR, KNN	Highly iterative scheme
Chen et al. [45]	Longer testing time	Test coverage prediction	C-Compilers (GCC, LLVM)	68% of speedy response	No comparison with another learning scheme

Apart from the studies mentioned above in Table 1, various other studies also underwent a rigorous reviewing process to have insights into their methodology, viz. neural network-based effort estimation (Rankovic et al. [46]), Artificial bee-colony-based predictive model toward SCE (Shah et al. [47]), Stacked ensemble approach using random forest-based effort estimation (Varshini et

al. [48]), dynamic programming based model (Wang et al. [49]), sequential optimization of the model (Xia et al. [50]), Function point-based estimation (Zhang et al. [51]). These methods offer unique contributions toward improving SCE by adopting various case studies. The following section highlights the problems being explored in due course of review of related work.

4. Problem Identification

From the prior sections, it is noted that there is various availability of SCE techniques where the prime inclination of methods is mainly machine learning based as well as metaheuristic based approaches. Most existing studies on SCE have been carried out over standard datasets and tools, overlooking the challenges associated with nonlinearity and complexity. Apart from this, various intrinsic factors are required to be adopted to improve the accuracy of SCE techniques. In contrast, most existing learning-based methods don't seem to consider these essential factors, e.g., feature engineering, data modeling, and inclusion of network parameters. Hence, accuracy is accomplished by deploying sophisticated learning models at the cost of computational complexity, which is not evaluated in performance analysis. Based on this, it is almost challenging to confirm if the existing learning-based approach will maintain its consistency as theoretically proven in the literature. Another observation in the current learning-based SCE scheme is that it doesn't emphasize much possible linkage among various nodes before training. Hence, the evolution of an ideal learning network is questionable. From the context of effort estimation, human resources are included in constructing the learning model followed by training it; however, this fact is not considered in constraint modeling. Apart from this, existing approaches are carried out considering a specific contextual scenario, applicable when the system could alter with the progress of an unknown time. Hence, when such a temporal parameter is subjected to minor changes, it can also affect the outcome of the training model in existing temporal-based modeling approaches. Existing approaches using mathematical-based methods towards effort estimation are highly dependent on varied attributes of software projects. They are not feasible; hence, their applicability and cost effectiveness of modeling are not identified with concrete justification. It is also noted that non-algorithmic-based approaches are increasing in their adoption towards improving SCE performance with promising accuracy outcomes using its learning-based schemes. However, aside from the accuracy, the computational complexity factor is not emphasized much, which should be considered for any part of the problem solution space. It was also noted that adoption of expert judgment is relatively frequent; however, their estimation is required to be trained to minimize the

possibilities of inconsistencies. The scope of non-algorithmic based approaches is somewhat higher owing to its better possibility towards modeling and fine-tuning of network parameters. Therefore, the identified research problems are:

- *Non-Inclusion of Dynamicity in Dataset*: Adopting the dataset, be it standard or industry, is always a better way to justify proof of concept; however, it is necessary to include specific ranges of dynamic parameters to ensure the sustainability of the model when exposed to a real-world environment. Most existing studies do not include uncertainty-based constraint modeling in their data management before / during training, which narrows down the scope of applicability in reliable SCE.
- *Processing Complexity*: Irrespective of knowing that learning-based schemes could offer better error reduction control with increasing iteration, the prime emphasis was only on accomplishing predictive error reduction in SCE and not much on reducing the processing complexity of learning-based schemes. If the complexity is not controlled, then its subjectivity towards error reduction is also questionable.
- *Less Emphasis on Consistency*: Irrespective of any form of adoption of existing SCE techniques, there is non-availability of any standard SCE technique that confirms consistency even for software projects of similar structure. Software projects of a similar domain will inevitably have timely revision for either inclusion or exclusion design entities to suit the final product requirement. Moreover, the human-resource-based factor is another attribute that could lead to a reduction in consistency apart from autonomous SCE models.
- *Narrowed Evaluation Scope*: Most existing techniques are evaluated on individual datasets using less varied test environments. This reduces the applicability of presented methods irrespective of claimed accuracy. Apart from this, fewer reported benchmarked studies on SCE techniques could not comprehensively reflect its effectiveness from a global software development perspective. The presented scheme needs to be under a rigorous test scenario with varied datasets and compared with maximum standard approaches to claim the SCE methodology's applicability.

The following section presents a solution to addressing the above-identified research problem through research methodology.

5. Research Methodology

The prime aim of the proposed system is to introduce a novel and yet simplified computational framework for SCE considering global software project development using machine learning approach. With availability of various forms of machine learning approaches, the proposed system considers neural network-based learning approach which offers an intensive processing capabilities of nonlinear information associated with various attributes of cost estimation in software engineering. From the perspective of estimating cost associated with global software development, the existing system shows adoption of activation function and learning techniques e.g. sigmoid function, back-propagation algorithm, multi-layer perceptron, and feed forward. However, from the prior section, it was noted that adoption of learning-based approaches has inclusion of various challenges irrespective of its beneficial aspect. Similarly, it was noted that suitability of neural network is more towards solving categorization and classification problem whereas in reality, the demand of generalization is more in cost estimation towards global software development in contrast to emphasize on classification-based issues. Apart from this, it is also noted that there is inclusion of various non-linear attributes towards SCE process e.g. duration, number of human resources, availability of enabling technologies, degree of utilization of varied organizational resources, target budget to comply with, etc. All this information, in the form of dataset, finally formulates a non-linear and complex form of dataset in software engineering, which when subjected to neural network has higher possibility to yield maximized computational complexity as well as minimal accuracy of predictive score of SCE. Therefore, this research challenge is identified to be addressed using search-based optimization approach by harnessing the process of selecting the unique feature that can assists in optimizing the class selection of global software project. Different from any existing learning scheme towards SCE, the proposed system introduces a novel approach where search-based optimization approach is developed in order to balance the predictive accuracy performance as well as computational complexity while performing SCE operations.

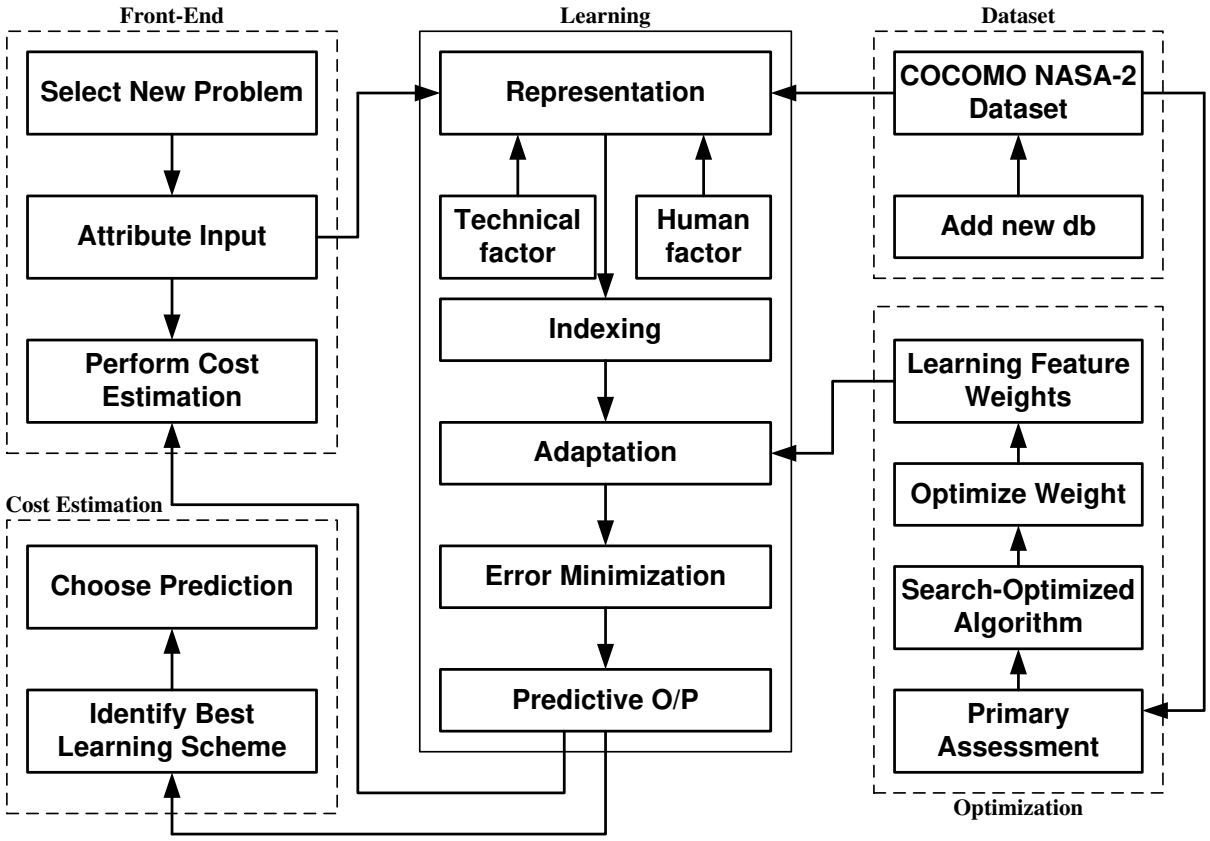


Figure 2 Proposed Architecture of Optimized SCE

The above Fig.2 highlights the architecture that is constructed in other to implement the proposed concept named as Optimized Learning-based Cost Estimation (OLCE) in global software projects. A closer look into the above architecture shows a highly interconnected set of block-operation which are mainly classified into 5 types viz. i) front end operation, ii) dataset handling operation, iii) learning-operation, iv) optimized operation, and v) cost estimation operation. All the classified blocks are meant to carry out a discrete set of operation in order to achieve a common goal of SCE performance. The briefing of the involved block operations are as follows:

- *Front End Operation*: This is basically a user-defined interface that is meant for the stakeholder to select the design of the upcoming project which is subjected for SCE. The block of *select new problem* pertains to selection of software project design related to specific software domain whereas the block of *attribute input* is related to all possible anticipated attributes considered by stakeholder e.g. duration of development, duration of testing, number of software resources, number of human resources, assigned probability of uncertainty score associated with risk, possible overall development cost etc. The block of

perform cost estimation is meant to execute the proposed scheme where estimation of software cost is carried out considering the taken software project and its selected attributes. This interface also offers a stakeholder a greater deal of flexibility to edit the inclusion / exclusion of number of attribute input for clearly evaluating the cost.

- *Dataset Handling Operation*: This is the second main block of operation which retains *COCOMO NASA-2 dataset* as well as it also enables the evaluator to *add new fields or values* in the existing dataset in order to perform extensive assessment of given dataset.
- *Learning-Operation*: The third essential block of operation which carry out a series of operation in order to carry out learning operation. Different from any existing learning-based schemes, the proposed scheme performs new set of operation prior to performing actual learning in order to boost up accuracy along with equal emphasis towards computational efficient. The block of *representation* is responsible for depicting the identification number of a use-case of software project along with discrete set of problems from input attributes and its anticipated solution. The representation is carried out with respect to technical problem as well as human related errors. The block of *indexing* is responsible for discrete allocating identification number for both technical and human-related representation which is further followed by block operation of *adaptation*. The prime task of adaptation block is towards fine-tuning the cost of development (F_c) as follows:

$$F_c = D_c \; x \; R_c \tag{1}$$

In the above expression (1), the variable D_c and R_c represents proportional dimension of new case and revised cost of development. This empirical expression is further amended as follows:

$$F_{c1} = (\alpha/\alpha_i).\ Fc \tag{2}$$

In the above expression, F_{c1} represents final evaluated cost of new development whereas the variable α and α_i represents correlated value of software project use case with i^{th} identification number. The above expression, thereby, represent that proposed scheme performs finetuning of the cost of development of new use case considering two dimensional attributes of D_c and R_c while the first component (α/α_i) represents the mean weight obtained from revised cost of development connected with retrieved case of proximity of anticipated target of SCE on the basis of correlation. Further the *minimization*

of error block carries out using artificial neural network followed by reaching the terminating stage of learning to yield primary *outcome of prediction*.

- *Optimized Operation*: The proposed learning operation contributes towards inclusion of series of operation in order to optimize the learning performance using this block of operation. For this purpose, a block of *primary assessment* captures the input from COCOMO NASA-2 dataset and constructs use cases which will be used for both learning as well as validation in order to find out the difference in cost of software development. The core idea is to find out the proportionate cost of new development and also finding the use-case which doesn't demand any new development using correlation analysis. This difference in the use-case values are the subjected to *search optimized algorithm* adopted for finding solution towards multiple set of constrained as well as unconstrained problems by iteratively amending the population of individual values using expression (1). The outcomes are further used for *learning features of weight* using component (α/α_i) in expression (2) which basically act as an input for *adaptation* block in learning operational block. The core contribution of this block is to reduce the effort of learning by selecting optimal features from the dataset.
- *Cost Estimation Operation*: It can be noted that majority of the operation towards yield learning outcome and optimization is already carried out in *learning block* and *optimization block*, but still they are carried out for all the local values of OLCE. The local values pertain to best feature selected for all individual use cases that are found to be considered for newly development. Therefore, in order to simplify the operation, the proposed system introduce a final block of *cost estimation* which is responsible for evaluating the effectiveness of proposed learning model. For this purpose, a block *identify best learning scheme* executes a series of available learning scheme considering global attributes of new use cases of software while it also facilitates the stakeholder to opt for their self-selected predictors by using *choose prediction block*. It should be noted that the output of this block is not meant for final user and is just meant to give a second window of validation of adopted learning model for benchmarking while the final predictive score is evaluated at the end of learning block operation itself. However, in case, of encountering a less satisfactory predictive performance score, it is now feasible for an evaluator to alter the attribute input in front-end block operation and assess the final predictive score of SCE.

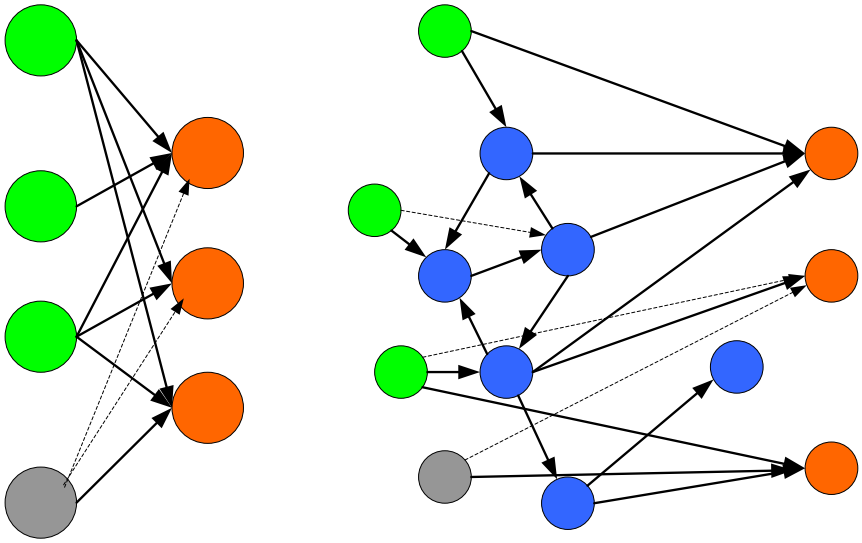
Therefore, on the basis of above block operation associated with Fig.2, it can be noted that novelty of proposed research methodology is that it offers a simple user-defined interface that runs a sophisticated series of machine learning operation, which is not only novel but also less iterative and more progression, targeting towards computationally cost effective SCE model.

6. Optimized Learning-based Cost Estimation (OLCE)

This section elaborates about the internal operation being carried out in *learning block* and *optimization block* in prior Fig.2. The prime purpose of OLCE is to incorporate an automation towards software development as well as performing training operation for neural network frameworks using search-based optimization scheme. This scheme is meant for evolving the units of neural network along with its structure and parameters of learning in order to perform predictive evaluation of the stability of computing SCE. According to the proposed search-based optimization scheme, the scheme constructs initial set of population followed by scoring and scaling it. It further retains the best outcome while parents are selected in order to generate an outcome that is further used for scoring and scaling population. In all the above mentioned steps of operation, the scheme adopts three core rules in order to finetune the outcome of population viz.

- *Rule for Selection*: This is the primary rule which is responsible for choosing the individual termed as parent that yields a revised set of population in consecutive rounds of operation.
- *Rule for Aggregation*: This is the secondary rule which is responsible for generating a revised population by integrating two parent information.
- *Rule for transformation*: This is the ternary rule which is responsible for transforming the single parent in order to generate a revised set of single population arbitrarily.

The novelty of OLCE scheme are as follows: i) this scheme is capable of yielding multiple outcomes of population over every computation where the stopping point of iteration is confirmed after obtaining optimal score based on fitness function, ii) the scheme uses arbitrarily selected number in order to choose the next cycle of population thereby making the system quite fast responsive, and iii) it is capable of processing multiple evaluation function directing towards converging rate. In order to understand the novelty of proposed scheme, Fig.2 elaborates the operation of it in comparison to conventional technique.



(a) Conventional Learning (b) OLCE Learning

Figure 3 Difference between conventional learning and OLCE

Fig.3(a) highlights the difference between conventional learning approach using neural network while Fig.2(b) highlights the amended learning scheme using OLCE. A closer look into the topological difference in Fig.3(b) highlights that they are augmented from the conventional one where neural network is integrated with search-based optimization scheme. Fig.2 also represents an initial topology of neural network (Fig.3(a)) which after several rounds of initial iteration yields a final topology of OLCE that uses neural network with search-based optimization scheme (Fig.3(b)). The internal operation carried out towards amending the conventional topology in OLCE is shown in Fig.3. For this purpose, the OLCE considers loss function and network hyperparameter for initializing the variables for undergoing the process of amening the neural network training. The study considers *number of neuron* and *rate of learning* as the *hyperparameter variable*, whose initialization is considered as highly significant operation step in order to evaluate the performance of network during the training stage especially during applying rules of aggregation and rule of transformation involved in OLCE. Similarly, the optimality factor associated with the weight involved in learning phase and bias is determined using *loss function*, where the study consider weight as synapse genes and bias as neuron gene. It is to be noted that proposed OLCE scheme considers its fitness function to be this loss function while the genomes are considered to be synapse genes and neuron genes.

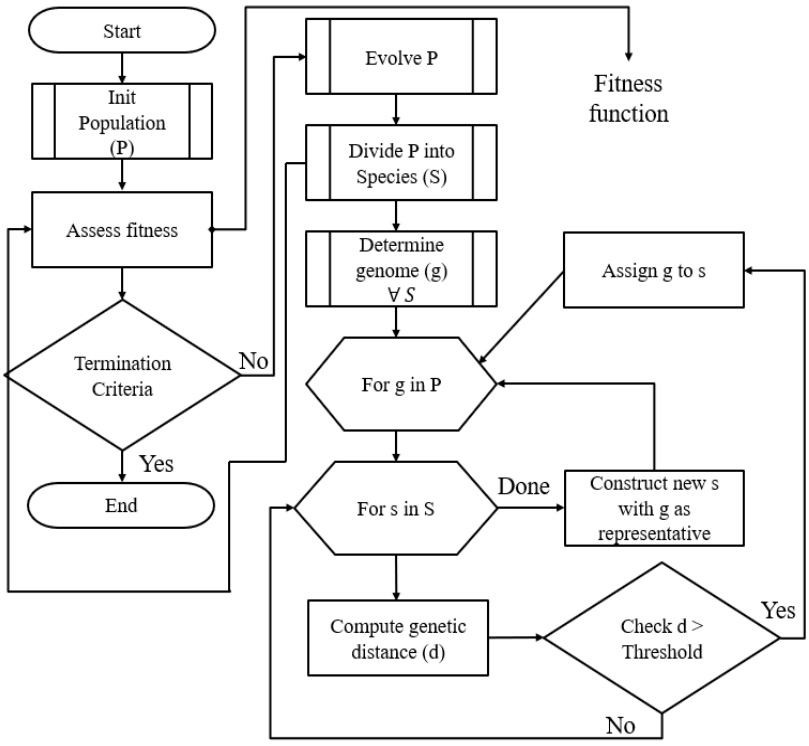


Figure 4 Internal Operational flow for proposed OLCE Scheme

After the process of initializing possible set of candidate solution, the scheme considers single layer of input and output when the genome is generated by the OLCE operation. This is the prime reason where the genomes are found to slightly vary with respect to its bias and weight; however, the network topology remains all the same in initial generation of genome. Upon reaching the termination criterion, the system framework stops the algorithm upon evaluating the fitness value linked with all the genomes. In case the termination criterion is not met than the novel set of candidate solutions are generate using rule of aggregation among the genomes followed by rule of transformation among the generated population from primary implementation in prior step. All the above mentioned process are performed arbitrarily. An evolutionary distance between biases and all the set of neuron weights are computed followed by splitting the candidate solution in specific classes that bears certain common characteristic. This process is carried out before computation of fitness of synapse genes and neuron genes. An empirical expression towards computation of evolutionary distance E_{dis} can be mathematically expressed as follows:

$$E_{\text{dis}} = dis_1 + dis_2 \tag{3}$$

In the above expression (3), the variable dis_1 and dis_2 represents bias and weight respectively. The implementation of the proposed OLCE is carried out in python environment by splitting the dataset

into training data (80%) and testing data (20%). OLCE uses neural evolution method in order to perform design configuration which carries out evaluation of biases and weights from the features extracted from the input observation. An evolutionary scheme is used for determining the optimality of neural network architecture via augmentation of topology in this process. The system further constructs neural network using following parameters of configuration e.g. set of transfer function, hidden layers, and neurons at each hidden layers. For an effective analysis of proposed OLCE scheme, multiple transfer functions e.g. non-linear sigmoid, relu, and linear functions has been used in the form of transfer function. The evaluation of the fitness function is carried out considering reduced error rate by considering mean squared error. Further the optimization of fitness function is carried out using inverse roulette selection. Fig.4 highlights the augmentation in the considered topology for performing both the internal essential operation of neural network as well as its associated process of training. By initializing the set of candidate solution, the system begins with the augmentation of the topology using a pool of arbitrary neural network. Upon multiple levels of iterations (also known as generation), the proposed scheme selects an optimal neural network on the ground of value of fitness function, which is further subjected to rule of aggregation in the line of selection or decision making process. Continuation of this process leads to generation of neural network that upon implication of rule of transformation yields an evolved topology of neural network. This evolved topology is further subjected to training process. The complete process is iterated until the system meets the termination condition.

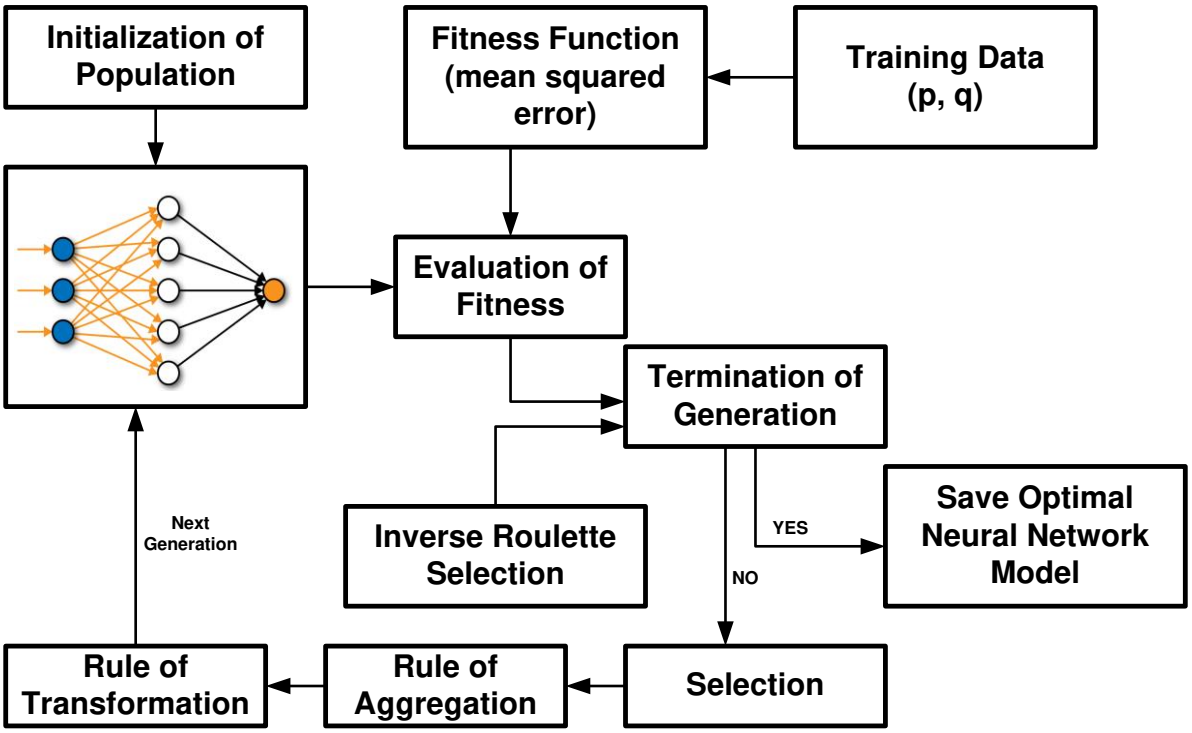


Figure 5 Yield of an Optimal Neural Network

It should be noted that the termination criterion is essentially dependent upon the designated number of generation while the evaluation of a training model is carried out on each generation followed by performing selection of best topology in order to arrive to predicted outcome. The steps of implementation of the OLCE training operation are highlighted as follows:

- *Constructing a Pool of Population*: This is the primary step of OLCE which is responsible for generating a pool of population along with a yield of arbitrary activation function and a set of an arbitrary neural network with neurons and arbitrary layers. The neurons as well as definitive number of layers are considered as an input for this process in this step along with set of activation function. The proposed scheme uses sigmoid, relu, and linear as activation function.
- *Evaluation of Population Fitness*: The scheme uses mean squared error in order to measure the population fitness where the mean squared error of input data is considered with the outcome linked with set of training.
- *Choosing the Fittest Individual*: The proposed scheme implements an inverse Russian roulette process in order to choose the individual for the purpose of repopulating. The probability of the selection is considered to be higher if the value of the fitness function is

minimal. The computation of the probability of selection $Prob_{sel}$ is carried out by following process:

$$Prob_{sel} = 1 - \frac{E_i}{\sum_{i=1}^{n} E_i} \quad (4)$$

In the above empirical expression (4), the variable E_i represents mean squared error for n number of population.

- *Fittest network for Repopulation*: This step is responsible for performing repopulation considering the available scores of fittest networks. The selection of the maximum fit individuals is carried out for all available selected population that is considered for further processing. The rule of aggregation is applied on these individuals followed by applying rule of transformation depending upon its probability.
- *Updating Network Weights*: This is the final step of OLCE process where the normally distributed rules of transformation are used towards the network weights. This step also finalizes the process of generation of optimal topology of neural network while the newly generates networks are finally introduced to the pool of population as the updating process.

Therefore, it can be seen that proposed OLCE scheme adopts more of conditional assessment towards reaching optimal state of network in order to facilitate an effective SCE, unlike iterative and sophisticated form of conventional learning approach. The next section discusses about the outcome obtained from proposed OLCE implementation.

7. Result Analysis

This section elaborates about the proposed scheme of SCE on the basis of neural network framework using OLCE approach in prior section. The evaluation process of proposed model is based on predicting Kilo Lines of Code (KLOC) for investigating the regression problem associated with cost estimation of large software projects.

7.1 Strategy of Implementation

One of the essential strategy towards implementing the proposed OLCE scheme is to perform an extraction of an appropriate input as well as task of feature engineering. This is accomplished using preprocessing operation which mainly uses normalization of data as well as correlation analysis. From the perspective of correlation analysis, a simplified mathematical approach is used to explore the linkage among numerous variables in order to find similarity among multiple cost drivers in SCE. The mathematical formulation of correlation $C_{p, q}$ this purpose is represented as following:

$$C_{p,q} = \frac{\sum \Delta p.\Delta q}{\sqrt{\sum \Delta p^2.\Delta q^2}} \tag{5}$$

In the above expression (5), the variable Δp and Δq represents (p_i-p`) and ($q_i$-$q$`) respectively, where the cost drivers are represented by variables p_i and q_i and mean value of cost driver is represented as p` and $q$`. The numerical score of $C_{p,q}$ is considered to be within the range of [-1, +1]. From the above mathematical formulation, it can be noted that if the variable p is directly proportional to variable q than $p=\theta.q$, where θ is a constant. In such case, the expression (5) can be further modified as following:

$$C_{p,q} = \frac{\sum \Delta p.\theta \Delta q}{\sqrt{\sum \Delta p^2.\theta^2 \Delta q^2}} \tag{6}$$

$$C_{p,q} = \frac{\sum \theta.\Delta p^2}{\theta\sqrt{(\sum \Delta p^2)^2}} \tag{7}$$

$$C_{p,q} = \frac{\theta \sum.\Delta p^2}{\theta \sum \Delta p^2} \tag{8}$$

$$C_{p,q} = 1 \tag{9}$$

In the above expression (12), it can be seen that final value of correlation is always found to be unity when there is a presence of proportional cost drivers to each other. Equivalently, decreasing of one cost driver will lead to increase of another cost driver and in such condition the correlation can be stated to be -1. This is considered to be an ideal environment with two cost drivers to be having a perfectly linear relationship. In case there is a zero correlation than this state will depict its total randomness with absence of any link between two cost drivers. The plot of the same is shown in Fig.6.

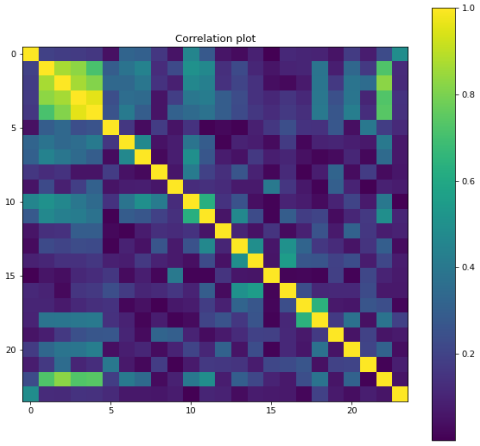


Figure 6 Plot of Correlation between cost driver and KLOC

Further, the proposed OLCE scheme converts the input data in vector form in order to make its suitable towards acting as input for learning process. The usable vectors are obtained by converting the rows of values using feature vectorization. The scheme uses scaling method using min-max for carrying out the normalization of the data. Finally, the transposition of rows is carried out following by subjecting it to neural network. It is empirically expressed as,

$$d_{norm} = \frac{d_1}{d_2} \tag{10}$$

In the above expression (10), the variable d_{norm} represents normalized data while the variable d_1 and d_2 represents (d-min(d)) and (max(d)-min(d) respectively, where variable d represents input data, which represents original cost driver rescaled in range of [0, 1]

7.2 Dataset Adoption

The assessment of the proposed study is carried out considering COCOMO NASA-2 dataset [52]. The dataset is acquired in the form of raff form consisting of multiple essential segments viz i) title, ii) prior usage, iii) pertinent information, iv) quantity of instances, v) quantity of attributes vi) information of attributes, vii) missing attributes, viii) distribution of class, and ix) data. This data is finally stored in *.csv* form. The imported dataset is now characterized by 124 entries in the range of [0, 123] structured within a 24 columns for both numeric and categorical form. Executed over conventional windows machine the scheme consumed closer to 25 KB of memory in order to upload the data. The study considers 25 predictors from the dataset with no missing values. The preliminary analysis of the dataset shows the difference between reference pairs are not always lower than the computed standard deviation thereby representing presence of artifacts within the data. Further, it was also noted that decision to rectify the outliers is dataset is considered if the numerical score of mean squared error and root mean squared error is quite higher than mean value of KLOC. It was also noted that there is prevalence of particular correlation, either positive or negative, with the effort parameter within observed dataset. With the decrease of parameter values, the effort is also found to be decreased in positive correlation while its vice-versa in negative correlation. According to the observation, it was found that cost drivers associated with *analyst's capability* (acap) and *programmer's capability* (pcap) are positively correlated while the parameters e.g. *required software reliability* (rely), *process complexity* (Cplx), *data base size* (data), *time constraint for cpu* (time), *main memory constraint* (stor), *schedule constraint* (sced) are negatively correlated. Moreover, it was noticed that effort (or software cost) is potentially affected due to minimal value of reusability and site attribute in dataset while performing linear

regression. Hence, the conclusive remark of the dataset is that data point correlation associated with actual efforts bears the characteristic of non-uniformity. This demands an implication of feature engineering using proposed OLCE scheme.

7.3 Result Accomplished

As the proposed scheme is associated with applying learning approach to perform a predictive computation of software cost associated with the proposed scheme, therefore, the analysis of the outcome is assessed using 5 standard performance metric e.g. Mean Magnitude of Relative Error (MMRE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Predicted Cost (PC), and Response Time (RT). Apart from using benchmarked dataset of COCOMO NASA-2, it is required that proposed scheme be compared with maximum possible existing scheme in order to showcase its sustainability of SCE. A quick look into the historical trend shows various evolving approaches for this purpose. The *first approach* considered as existing system to be compared with is Logistic Regression Approach (E1) that usually performs analysis of probability of specific event in software engineering considering independent variables present in dataset [53][54][55]. The *second approach* considered for comparative analysis of Support Vector Regression (E2) which uses its supervised learning capabilities in order to computed predictive effort estimates for a defined case study [56][57][58]. The *third approach* considered is genetic algorithm (E3) which is basically a conventional search-based optimization algorithm which arrives to final result from a set of population on the basis of fitness function [59][60][61]. The *fourth approach* for comparison is the most frequently used bat algorithm (E4) which is basically a metaheuristic method capable of fulfilling demands of global optimization for large software global project [62][63][64]. The *fifth and sixth approach* considered in analysis is another most frequently used Dolphin algorithm (E5) and hybrid model of Dolphin-Bat algorithm (E6) [41]. The *seventh approach* for performance comparison analysis is currently most adopted practice of artificial neural network (E7) that is characterized by multiple processing elements working on predefined activation function in order to arrive at an elite outcome [46]

7.3.1 Analysis of MMRE

This performance metric is used for assessing the effort estimation ideal for global software projects which is empirically computed as follows:

$$MMRE = \frac{1}{N}\sum_{i=1}^{n}\frac{ae-ee}{ae} \quad (11)$$

In the above expression (11), MMRE is computed on the basis of *N* number of software project considering actual effort *ae* and estimated work effort *we*. The computation of MMRE yields a value to reflect the average score of error between actual effort and estimated work effort. Hence, lower the value of MMRE, higher is the accuracy of approaches being evaluated on it. The outcome of the MMRE is showcased in Fig.7 where it can be seen that proposed OLCE is found better comparison to all the existing approaches of SCE (i.e. E1-E7). It can be noted that performance for Logistic Regression (E1) as well as Support Vector Regression (E2) showcase higher ranges of RMSE exhibiting its unsuitability over larger global project cost estimation. The prime issue is associated with impractical dependency on all the variables in Logistic Regression (E1) associated with linearity which is never the frequent cases of global software project while the issue of Support Vector Regression models (E2) is found highly unsuitable for larger dataset. In contrast, the performance of Artificial Neural Network (E7), Dolphin Bat algorithm (E6), Dolphin Algorithm (E5), and Genetic Algorithm (E4) has similar performance scale with less significance performance difference among them. The prime reason behind this is the capability of learning scheme to find out effective features towards training resulting in lesser error. However, the prime pitfalls of all the above learning models is that its mechanism of confirming the accuracy is least updated and based on static attributes of dataset. On the contrary, proposed OLCE offers two distinguished layers of operation i.e. learning and optimization operation, where without using much iterative steps, more emphasis is given on feature engineering resulting in evolution of progressively higher accuracy score.

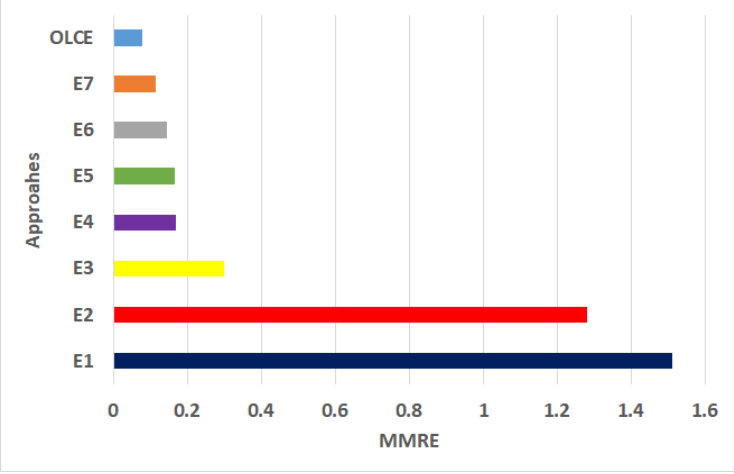


Figure 7 Comparative Analysis of MMRE

7.3.2 Analysis of MSE

The prime motive of analyzing MSE is to evaluate the proximity of regression line towards target set of data points. An empirical formulation considered for this purpose is,

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(ae - we)^2 \quad (12)$$

In the above expression (12), the computation of MSE is carried out on similar variables as that of MMRE where the idea of this evaluation is to showcase the effectiveness of optimality of learning models. From Fig.8, it can be seen that nearly similar trends of results for both regression approach (i.e. E1 and E2) is witnessed. However, MSE for Artificial Neural Network (E7) is found superior to that of Genetic Algorithm (E3), bat algorithm (E4), and two variants of dolphin algorithm (i.e. E5, E6). Fig.8 eventually showcase that OLCE scheme offers better predictive accuracy by harnessing the potential of Artificial Neural Network (E7) and Genetic Algorithm (E3). The novelty is that although OLCE retains legacy benefits of E7 and E3 models, but still it incorporates a second layer of validation and updating the predictive outcomes after the optimization is performed. This is not witnessed in any existing schemes.

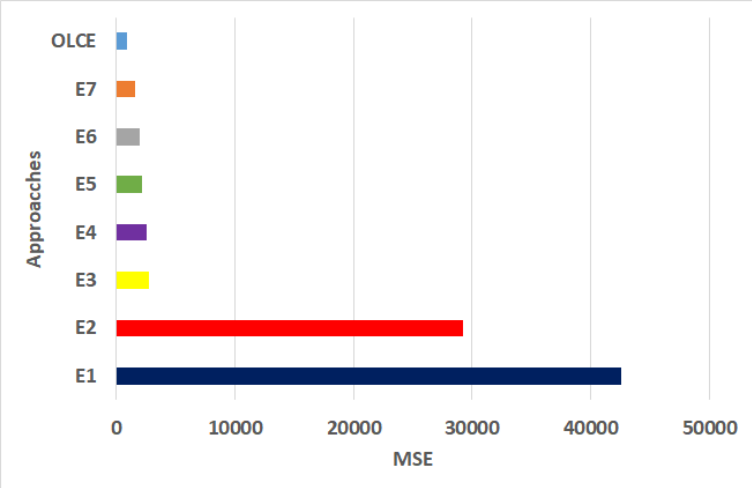


Figure 8 Comparative Analysis of MSE

7.3.3 Analysis of RMSE

The evaluation of RMSE is carried out in order to furnish more interpretable inference compared to MSE. The computation of MSE involves squaring of LOC^2 where LOC is lines of codes involved in software project, however, the values of MSE is quite challenging to interpret sometimes. Empirically, the computation of RMSE is carried out as follows,

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{n}(ae - we)^2} \quad (13)$$

Similar variables used in computation of MSE is also used here for RMSE evaluation to arrive for a graphical trend shown in Fig.9, where more granular information and interpretation can be arrived. The graphical outcome shown in Fig.9 exhibits that there is only a slight difference in performance of Logistic Regression (E1) and Support Vector Regression (E1) unlikely that of MSE with significant difference. This exhibit non-applicability of both these models in SCE evaluation. Another obvious trend observed is that of two set of models with distinction. The first set of existing models (i.e. Dolphin model (E5), Dolphin bat model (E6), and Artificial Neural Network (E7)) shows distinctive difference with the second set of existing models (i.e. bat algorithm (E4) and Genetic Algorithm (E3)). It also infers that hybrid version of dolphin model i.e. E6 is not much significantly different from legacy version of Dolphin model i.e. E5. Similarly, applying conventional genetic algorithm (E3) couldn't yield to higher accuracy in contrast to conventional optimization model of bat algorithm (E4) owing to usage of static assumption in formulating fitness function. On the contrary, proposed scheme of OLCE is witnessed to significantly score higher accuracy owing to inclusion of progressive optimization steps linked to adaptation operation in learning block of proposed architecture.

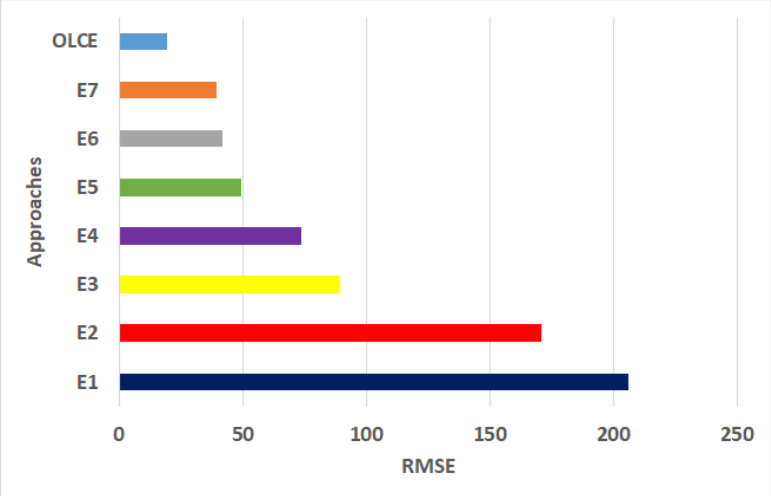


Figure 9 Comparative Analysis of RMSE

7.3.4 Analysis of Predictive Cost

This performance metric is used for analyzing the accuracy of predictive cost model. According to this concept of evaluation, the proposed scheme considers a milestone point x, which is well-defined point of accuracy and the idea of this evaluation is to find out if the predictive accuracy falls under the range of $x\%$. Mathematically, the Predictive Cost PC is expressed as follows,

$$PC = \frac{1}{N}\sum_{i=1}^{n}\left|\frac{ee-ae}{ae}\right| x\% \quad (14)$$

In the above expression (14), the system considers x% as the percentage of error existing between actual effort *ae* and estimated effort *ee* while the computation of Predictive Cost (PC) is represented in the form of percentage of quantity of software project whose threshold for cost overrun and under is considered hypothetically to be 20% and 30% respectively. These values can always be amended if the domain of the software project is found to be different. This can be carried out through the common interface module constructed in the front end handled by the estimator/stakeholder of software projects. Computed values of PC is shown in Fig.10 where it can be seen that proposed OLCE has excelled with superior accuracy compared to all existing scheme frequently adopted for SCE. A closer look into the trends of model will show the outcome in agreement with error computation shown in MMRE, MSE, and RMSE. The unsuitability of Logistic Regression (E1), Support Vector Regression (E2) and Genetic Algorithm (E3) is shown to be quite higher due to potentially reduced PC values. Whereas, the suitability of remnant existing approaches (i.e. E4, E5. E6, and E7) is good enough. However, proposed OLCE operational beneficial are potentially more to overcome the limitations of all these approaches with respect to accuracy towards predictive evaluation.

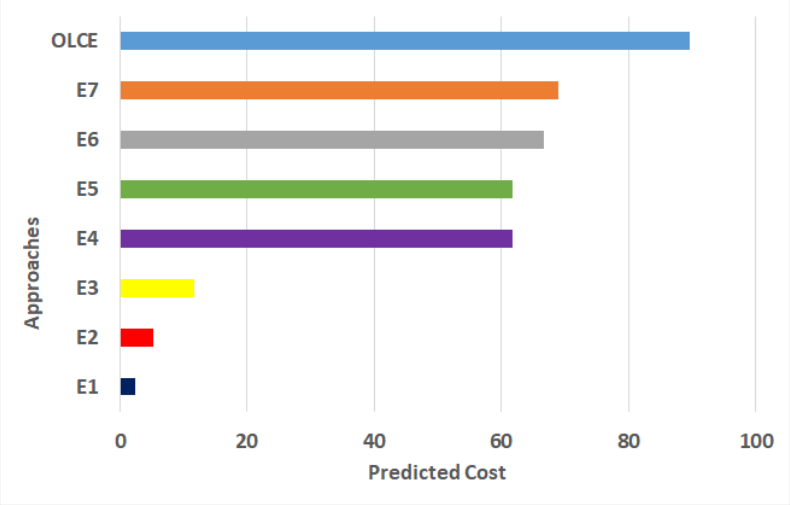


Figure 10 Comparative Analysis of Predicted Cost

7.3.5 Analysis of Response Time

This performance metric of response time is basically responsible for assessing the computational burden while performing the predictive operation of OLCE. Lower the response time, the model

can be stated to perform efficiently without much computational overburden. The mathematical expression deployed to compute response time is as follows,

$$RT = \frac{1}{S}[T_1 + T_2 + T_3] \quad (15)$$

In the above expression (15), the computation of the response time is carried out considering three temporal attributes i.e. i) time consumed for data processing and normalization T_1, ii) Time required for execution of learning block of operation T_2, and iii) Time required for optimizing and validating T_3. Deployed in conventional 64 bit windows machine, without any external threads running, the programmatic computation of overall time shows that proposed OLCE scheme offers quite a faster response in seconds while other conventional process is found to consume a higher ranges of time. One of the justified reason behind this is almost all the existing approaches has inclusion of higher number of iterative steps in order to perform computation of cost estimates whereas proposed scheme addresses this computational burden by emphasizing more on feature extracting and enhancing them in due course of learning thereby offering better convergence performance. Apart from this, proposed scheme has a separate blocks of optimization unlike any existing scheme where conditional assessment towards the convergence of target outcome is rendered quite faster due to adaptation of learning weights and biases. This offers more balance between achieving error minimization over reduced period of time. Hence, the proposed scheme excels better response time and is capable of handling large scale and dynamic information associated with estimating cost for software projects with consideration of uncertainty in it, unlike any existing scheme.

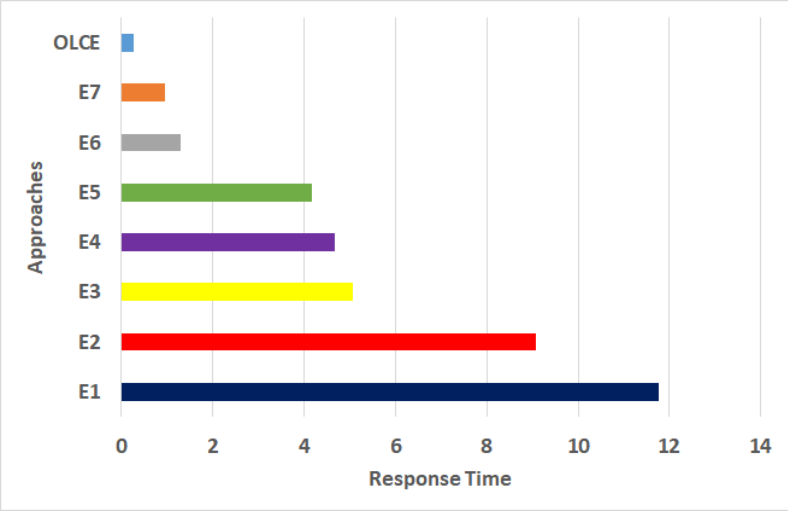


Figure 11 Comparative Analysis of Response Time

8. Conclusion

This paper has discussed about the challenges involved in estimating cost of software projects, which is found to be yet an unsolved problem irrespective of various archives of solution present in literatures. Upon reviewing the existing literatures, it is found that there are open-end issues related to the topic viz. non-inclusion of dynamicity in dataset, processing complexity, less emphasis on consistency, and narrowed evaluation scope. Apart from this, the most significant concern is absence of benchmarked model towards cost drivers and lack of precise and suitable system to carry out accurate estimation. This paper contributes towards introducing a novel learning-based approach in order to perform predictive computation of SCE associated with large global software projects. The contribution of the proposed study are as follows: i) the proposed study evaluates the effectiveness of the existing literatures with potential contribution to find that machine learning-based approaches have higher scope towards cost estimation whereas other research methodology still encounters problems with applicability to complex environment, ii) different from existing learning schemes where the dataset is directly applied to learning model, the proposed OLCE scheme makes a discrete multiple blocks of operation with defined operation which offers higher accountability and transparency towards SCE, iii) The integration of multiple block operation (front-end, learning, dataset, optimization, cost estimation) are carried out in such a way that they reduces the iterations and becomes more progressive enough to balance both accuracy demands as well as computational efficiency, which is overlooked in existing approaches, iv) the novelty of proposed scheme is to introduce a hybrid SCE model which harnesses the potential of artificial neural network as well as search-based optimization for developing adaptable learning scheme as well as optimization scheme respectively, and v) benchmarked with standard COCOMO NASA-2 dataset, the quantified outcome of proposed scheme revealed that proposed system exhibits 44.80% of lower MMRE, 99.96% of reduced MSE, 76.15% of reduced RMSE, 49.90% of increased predictive accuracy, and 50% of faster response time in contrast to existing scheme. The future direction of work will be carried out towards further optimizing the outcome with more emphasis towards constraint modelling.

Data Availability Statement:

The assessment of the proposed study is carried out considering COCOMO NASA-2 dataset

Conflict of Interest Statement:

The authors Does not have any conflict of interest.

10. Reference

[1] F. Tayyari, Cost Analysis for Engineers and Scientists, CRC Press, ISBN: 9780429778179, 0429778171, 2021

[2] Y. Shen, Y. Yang, Y. Wang, D. Chang, "Software crowdsourcing task pricing based on topic model analysis", IET Software (Open Access), 2021. doi: https://doi.org/10.1049/iet-sen.2019.0168

[3] K. Rak, Z. Car, I. Lovrek, "Effort estimation model for software development projects based on use case reuse", Wiley Software: Evolution and Process, pp.1-17, 2019. doi:10.1002/smr.2119

[4] G. Kumar, P.K. Bhatia, "A Detailed Analysis of Software Cost Estimation Using Cosmic-FFP", Review of Computer Engineering Research, vol.2, No.2, pp.39-46, 2015. doi:10.18488/journal.76/2015.2.2/76.2.39.46

[5] M.A. Kuhail and S. Lauesen, "User Story Quality in Practice: A Case Study", MDPI-Software, vol.1, pp.223-243, 2022. doi:https://doi.org/10.3390/software1030010

[6] S. Denard, A. Ertas, S. Mengel and S.E.Osire, "Development Cycle Modeling: Resource Estimation", MDPI-applied Science, vol.10, No.5013, 2020. doi:10.3390/app10145013

[7] V.V. Hai, H.L.T.K. Nhung, Z. Prokopova, R. Silhavy, P. Silhavy, "A New Approach to Calibrating Functional Complexity Weight in Software Development Effort Estimation", MDPI-Computers, vol.11, No.2, 2021. doi: https://doi.org/10.3390/computers11020015

[8] L. Mak and P. Taheri, "An Automated Tool for Upgrading Fortran Codes", MDPI-Software, vol.1, pp.299-315, 2022. doi: ttps://doi.org/10.3390/software1030014

[9] C. Rosen, Guide to Software Systems Development-Connecting Novel Theory and Current Practice, Springer International Publishing, ISBN: 9783030397302, 3030397300, 2020

[10] C. Subramanian, S. Dutt, C. Seetharaman, B. G Geetha, Software Engineering, Pearson India, ISBN: 9789332541696, 9332541698, 2015

[11] Y. Keim, M. Bhardwaj, S. Saroop, A. Tandon, "Software Cost Estimation Models and Techniques: A Survey", International Journal of Engineering Research & Technology, vol.3, No.2, 2014. doi: 10.17577/IJERTV3IS20384

[12] W.L. Du, D. Ho, L.F. Capretz, "A Neuro-Fuzzy Model with SEER-SEM for Software Effort Estimation", ArXiv Software Engineering, 2015. doi: https://doi.org/10.48550/arXiv.1508.00032

[13] N. Ramasubbu, R.K. Balan, "Overcoming the challenges in cost estimation for distributed software projects", Proceedings - International Conference on Software Engineering, 2012. doi: 10.1109/ICSE.2012.6227203

[14] M.Jorgensen, T. Halkjelsvik, "Sequence effects in the estimation of software development effort", Elsevier-Journal of Systems and Software
Volume 159, January 2020, doi:https://doi.org/10.1016/j.jss.2019.110448

[15] A. A. Abdulmajeed, M.A. Al-Jawaherry and T.M. Tawfeeq, "Predict the required cost to develop Software Engineering projects by Using Machine Learning", Journal of Physics: Conference Series, Conf. Ser. 1897 012029, 2021

[16] S. S. Ali, M. Shoaib Zafar and M. T. Saeed, "Effort Estimation Problems in Software Maintenance – A Survey," 2020 3rd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), 2020, pp. 1-9, doi: 10.1109/iCoMET48670.2020.9073823.

[17] J. A. Khan, S. U. R. Khan, J. Iqbal and I. U. Rehman, "Empirical Investigation About the Factors Affecting the Cost Estimation in Global Software Development Context," in IEEE Access, vol. 9, pp. 22274-22294, 2021, doi: 10.1109/ACCESS.2021.3055858.

[18] L. Servadei et al., "Accurate Cost Estimation of Memory Systems Utilizing Machine Learning and Solutions from Computer Vision for Design Automation," in IEEE Transactions on Computers, vol. 69, no. 6, pp. 856-867, 1 June 2020, doi: 10.1109/TC.2020.2968888.

[19] T. Aoshima and K. Yoshida, "Pre-Design Stage Cost Estimation for Cloud Services," 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), 2020, pp. 61-66, doi: 10.1109/COMPSAC48688.2020.00018.

[20] M. El Bajta, A. Idri, "Identifying Software Cost Attributes of Software Project Management in Global Software Development: An Integrative Framework", ACM-Digital Library, No.39, pp.1-5, 2020. doi: https://doi.org/10.1145/3419604.3419780

[21] A. Mashkoor, T. Menzies, A. Egyed and R. Ramler, "Artificial Intelligence and Software Engineering: Are We Ready?," in Computer, vol. 55, no. 3, pp. 24-28, March 2022, doi: 10.1109/MC.2022.3144805.

[22] L.J. Gonçales, K. Farias, T.C. De Oliveira, M. Scholl, "Comparison of Software Design Models: An Extended Systematic Mapping Study", ACM-Digital Library-Computing Surveys, vol.52, No.3, pp.1-41, 2020. doi: https://doi.org/10.1145/3313801

[23] S. Shafiq, A. Mashkoor, C. Mayr-Dorn and A. Egyed, "A Literature Review of Using Machine Learning in Software Development Life Cycle Stages," in IEEE Access, vol. 9, pp. 140896-140920, 2021, doi: 10.1109/ACCESS.2021.3119746.

[24] S.Z. Iqbal, M. Idrees, A.B. Sana, N. Khan, "Comparative Analysis of Common Software Cost Estimation Modelling Techniques", Mathematical Modelling and Applications, vol.2, No,3, pp.33-39, 2017.doi: 10.11648/j.mma.20170203.12

[25] M.Azzeha, A.B. Nassif, I.B.Attili, "Predicting software effort from use case points: A systematic review", Elsevier-Science of Computer Programming vol.204, No.1, 2021.doi:https://doi.org/10.1016/j.scico.2020.102596

[26] S. Basha, P. Dhavachelvan, "Analysis of Empirical Software Effort Estimation Models", International Journal of Computer Science and Information Security, Vol. 7, No. 3, 2010

[27] R. Puri, I. Kaur, "Novel Meta-Heuristic Algorithmic Approach for Software Cost Estimation", International Journal of Innovations in Engineering and Technology, vol.5, No.2, 2015

[28] P. Pandey, "Analysis of the Techniques for Software Cost Estimation," 2013 Third International Conference on Advanced Computing and Communication Technologies (ACCT), 2013, pp. 16-19, doi: 10.1109/ACCT.2013.13.

[29] S.M.R. Chirra, H. Rez, "A Survey on Software Cost Estimation Techniques", Journal of Software Engineering and Applications, Vol.12 No.6, June 2019. doi: 10.4236/jsea.2019.126014

[30] A. Jadhav, M. Kaur, F. Akhtar, "Evolution of Software Development Effort and Cost Estimation Techniques: Five Decades Study Using Automated Text Mining Approach", Hindawi-Mathematical Problems in Engineering, Article ID 5782587, 2022. doi:https://doi.org/10.1155/2022/5782587

[31] K. Awada, M.Y.Eltabakh, C. Tang, M. Al-Kateb, S.Nair, G. Au, "Cost Estimation Across Heterogeneous SQL-Based Big Data Infrastructures in Teradata IntelliSphere", Industry and Applications Paper, 2020. doi: 10.5441/002/edbt.2020.64

[32] H. Lan, Z. Bao, Y. Peng, "A Survey on Advancing the DBMS Query Optimizer: Cardinality Estimation, Cost Model, and Plan Enumeration", Springer-Data Science and Engineering, vol.6, pp.86–101, 2021. doi: https://doi.org/10.1007/s41019-020-00149-7

[33] M. Autili, I. Malavolta, A. Perucci, G.L. Scoccia, and R. Verdecchia, "Software engineering techniques for statically analyzing mobile apps: research trends, characteristics, and potential for

industrial adoption", Springer Journal of Internet Services and Application, vol.12, No.3, doi: 2021.https://doi.org/10.1007/13174.1869-0238

[34] H. Y. Chiang and B. M. T. Lin, "A Decision Model for Human Resource Allocation in Project Management of Software Development," in IEEE Access, vol. 8, pp. 38073-38081, 2020, doi: 10.1109/ACCESS.2020.2975829.

[35] E. Cibir and T. E. Ayyildiz, "An Empirical Study on Software Test Effort Estimation for Defense Projects," in IEEE Access, vol. 10, pp. 48082-48087, 2022, doi: 10.1109/ACCESS.2022.3172326.

[36] M. Fernández-Diego, E. R. Méndez, F. González-Ladrón-De-Guevara, S. Abrahão and E. Insfran, "An Update on Effort Estimation in Agile Software Development: A Systematic Literature Review," in IEEE Access, vol. 8, pp. 166768-166800, 2020, doi: 10.1109/ACCESS.2020.3021664.

[37] M. I. Lunesu, R. Tonelli, L. Marchesi and M. Marchesi, "Assessing the Risk of Software Development in Agile Methodologies Using Simulation," in IEEE Access, vol. 9, pp. 134240-134258, 2021, doi: 10.1109/ACCESS.2021.3115941.

[38] V.V. Hai, H.L.T.K. Nhung, Z. Prokopova, R. Silhavy and P. Silhavy, "A New Approach to Calibrating Functional Complexity Weight in Software Development Effort Estimation", MDPI-Computers, vol.11, No.15, 2022. doi: https://doi.org/10.3390/computers11020015

[39] J. A. Khan, S. U. R. Khan, J. Iqbal and I. U. Rehman, "Empirical Investigation About the Factors Affecting the Cost Estimation in Global Software Development Context," in IEEE Access, vol. 9, pp. 22274-22294, 2021, doi: 10.1109/ACCESS.2021.3055858.

[40] J. A. Khan, S. U. R. Khan, T. A. Khan and I. U. R. Khan, "An Amplified COCOMO-II Based Cost Estimation Model in Global Software Development Context," in IEEE Access, vol. 9, pp. 88602-88620, 2021, doi: 10.1109/ACCESS.2021.3089870.

[41] A. A. Fadhil, R. G. H. Alsarraj and A. M. Altaie, "Software Cost Estimation Based on Dolphin Algorithm," in IEEE Access, vol. 8, pp. 75279-75287, 2020, doi: 10.1109/ACCESS.2020.2988867.

[42] H. L. T. K. Nhung, V. Van Hai, R. Silhavy, Z. Prokopova and P. Silhavy, "Parametric Software Effort Estimation Based on Optimizing Correction Factors and Multiple Linear Regression," in IEEE Access, vol. 10, pp. 2963-2986, 2022, doi: 10.1109/ACCESS.2021.3139183.

[43] M. Barenkamp, J. Rebstadt, and O. Thomas, "Applications of AI in classical software engineering", Springer Open, vol.2, No.1, 2020. Doi: https://doi.org/10.1186/s42467-020-00005-4

[44] H. D. P. De Carvalho, R. Fagundes and W. Santos, "Extreme Learning Machine Applied to Software Development Effort Estimation," in IEEE Access, vol. 9, pp. 92676-92687, 2021, doi: 10.1109/ACCESS.2021.3091313.

[45] J. Chen et al., "Coverage Prediction for Accelerating Compiler Testing," in IEEE Transactions on Software Engineering, vol. 47, no. 2, pp. 261-278, 1 Feb. 2021, doi: 10.1109/TSE.2018.2889771.

[46] N. Rankovic, D. Rankovic, M. Ivanovic and L. Lazic, "A New Approach to Software Effort Estimation Using Different Artificial Neural Network Architectures and Taguchi Orthogonal Arrays," in IEEE Access, vol. 9, pp. 26926-26936, 2021, doi: 10.1109/ACCESS.2021.3057807.

[47] M. A. Shah, D. N. A. Jawawi, M. A. Isa, M. Younas, A. Abdelmaboud and F. Sholichin, "Ensembling Artificial Bee Colony with Analogy-Based Estimation to Improve Software Development Effort Prediction," in IEEE Access, vol. 8, pp. 58402-58415, 2020, doi: 10.1109/ACCESS.2020.2980236.

[48] P. Varshini, A. Kumari, and V. Varadarajan, "Estimating Software Development Efforts Using a Random Forest-Based Stacked Ensemble Approach", MDPI-electronics, vol.10, No.1195, 2021. doi:https://doi.org/10.3390/electronics10101195

[49] M. Wang, Y. Ma, G. Li, W. Zhou and L. Chen, "Multi-Value Models for Allocation of Software Component Development Costs Based on Trustworthiness," in IEEE Access, vol. 8, pp. 122673-122684, 2020, doi: 10.1109/ACCESS.2020.3007158.

[50] T. Xia, R. Shu, X. Shen and T. Menzies, "Sequential Model Optimization for Software Effort Estimation," in IEEE Transactions on Software Engineering, vol. 48, no. 6, pp. 1994-2009, 1 June 2022, doi: 10.1109/TSE.2020.3047072.

[51] K. Zhang, X. Wang, J. Ren and C. Liu, "Efficiency Improvement of Function Point-Based Software Size Estimation with Deep Learning Model," in IEEE Access, vol. 9, pp. 107124-107136, 2021, doi: 10.1109/ACCESS.2020.2998581.

[52] http://promise.site.uottawa.ca/SERepository/datasets/cocomonasa_2.arff

[53] A.B. Nassif, M. Azzeh, A.Idri, and A. Abran, "Software Development Effort Estimation Using Regression Fuzzy Models", Computational Intelligence and Neuroscience Journal, Article ID 8367214, 2019. doi:https://doi.org/10.1155/2019/8367214

[54] P. Sentas, L. Angelis, "Categorical missing data imputation for software cost estimation by multinomial logistic regression", Elsevier-The Journal of Systems and Software, vol.79, pp.404–414, 2006. doi:10.1016/j.jss.2005.02.026

[55] F. Yücalar, D. Kilinc, E. Borandag, and A. Ozcift, "Regression Analysis Based Software Effort Estimation Method", International Journal of Software Engineering and Knowledge Engineering, Vol. 26, No. 5, pp.807–826, Article ID 8367214, 2016. doi:https://doi.org/10.1155/2019/8367214

[56] A. Corazza & S. Di Martino, & F. Ferrucci & C. Gravino, E. Mendes, "Investigating the use of Support Vector Regression for web effort estimation", Springer Empirical Software Engineering, vol.16, pp.211-243, 2011. doi:10.1007/s10664-010-9138-4

[57] A. L.I. Oliveira, "Estimation of software project effort with support vector regression", Elsevier-Neurocomputing, vol.69, pp.1749–1753, 2006. doi: doi:10.1016/j.neucom.2005.12.119

[58] A. Zakrani, A. Najm and A. Marzak, "Support Vector Regression Based on Grid-Search Method for Agile Software Effort Prediction," 2018 IEEE 5th International Congress on Information Science and Technology (CiSt), 2018, pp. 1-6, doi: 10.1109/CIST.2018.8596370.

[59] S. Bilgaiyan, D.K. Goswami, S. Mishra, M. Das, "Effort Estimation of Back-end Part of Software using Chaotically Modified Genetic Algorithm", International Journal of Intelligent Systems and Applications, vol.1, pp.32-42, 2019. doi:10.5815/ijisa.2019.01.04

[60] J. Murillo-Morera, C. Quesada-López, C. Castro-Herrera, and M. Jenkins, "A genetic algorithm based framework for software effort prediction", Springer Journal of Software Engineering Research and Development, vol.5, No.4, 2017. doi: 10.1186/s40411-017-0037-x

[61] R.K. Sachan, A. Nigam, A. Singh, S. Singh, M. Choudhary, A.Tiwari, and D.S. Kushwaha, "Optimizing Basic COCOMO Model using Simplified Genetic Algorithm", Elsevier-Procedia Computer Science, vol.89, pp.492–498, 2016. doi: 10.1016/j.procs.2016.06.107

[62] D. Nandal, O.P. Sangwan, "An Improved Bat Algorithm for Software Cost Estimation",International journal of simulation: systems, science & technology, 2018. doi:10.5013/IJSSST.a.19.04.10

[63] M. Padmaja, D. Haritha, "A Heuristic Effort Estimation Method using BAT

Algorithm through Clustering", International Journal of Innovative Technology and Exploring Engineering, ISSN: 2278-3075, Volume-8 Issue-9, July 2019.

[64] J. Zheng and Y.Wang, "A Hybrid Bat Algorithm for Solving the Three-Stage Distributed Assembly Permutation Flowshop Scheduling Problem", MDPI-Applied Science, vol.11, No.10102, 2021. doi:https://doi.org/10.3390/app112110102