Hybrid Algorithm for Software Cost Estimation Based on Combination of Greedy Algorithm and Machine Learning

Li Qi Zhen
Faculty of Technology and Information Science
Universiti Kebangsaan Malaysia
43000 Bandar Baru Bangi, Selangor Darul Ehsan, Malaysia
line 5: email address or ORCID

Zulkefli Mansor
Faculty of Technology and Information Science, Universiti Kebangsaan Malaysia, 43000 Bandar Baru Bangi, Selangor Darul Ehsan, Malaysia
kefflee@ukm.edu.my

***Abstract*—Accurate cost estimation is crucial for successful software project planning and resource allocation. However, software development is inherently unpredictable due to factors such as changing requirements, inter-module dependencies, and variations in team productivity, rendering traditional estimation methods increasingly ineffective. This paper proposes a hybrid cost estimation method that combines the deterministic efficiency of a hybrid algorithm with the predictive power of supervised learning specifically, linear regression to enhance the accuracy of early-stage estimates. The research begins by analyzing key project characteristics, including software size, complexity, team experience, and estimated effort. A hybrid algorithm serves as a constructive heuristic to systematically select the most influential cost drivers based on domain-informed criteria, ensuring critical factors are prioritized early in the estimation process. A linear regression model is then trained using these selected features to predict costs. The entire framework is implemented in Python using Jupyter Notebook, Google Colab, and PyCharm environments. To evaluate its effectiveness, experiments were conducted using industry-standard error metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). The results demonstrate that the proposed hybrid approach outperforms standalone models in both testing accuracy and computational efficiency. The greedy algorithm effectively reduces feature noise and dimensionality while highlighting underlying trends and quantitative relationships in the regression model. This work demonstrates that combining a simple heuristic with a statistical model offers a promising solution for practical cost estimation. Future research could explore applications to real-world datasets and investigate more advanced hybridization strategies.**

Keywords—estimation, machine learning, Greedy Algorithm, accuracy, hybrid

I. INTRODUCTION

Accurate cost estimation is a critical phase in software development planning, as it directly impacts feasibility assessments, budgeting, and resource allocation for the entire project. However, the growing complexity and dynamic nature of modern software systems—such as evolving requirements and shifting team compositions—have underscored the need for more robust, efficient, and adaptive estimation methods.

Conventional approaches, such as rule-based and algorithmic techniques, remain popular due to their simplicity. One widely used method is the Greedy Algorithm, which makes locally optimal decisions at each step. While computationally efficient, this approach often overlooks long-term effects, leading to suboptimal global solutions. In contrast, machine learning (ML) techniques can analyze historical project data to uncover complex relationships between variables. ML's predictive capabilities enable more accurate cost estimations, but its high complexity and susceptibility to overfitting may limit real-world applicability.

To address these challenges, this paper proposes a hybrid model that combines the strengths of both approaches: the Greedy Algorithm for rapid initial approximations and a learning-based model to refine those estimates. The goal is to achieve an optimal balance between estimation accuracy and computational efficiency.

II. LITERATURE REVIEW

A. The Greedy Algorithm in Cost Estimation

Greedy Algorithm is a heuristic for constructing a solution from an empty one by successively selecting in the terminals the most beneficial/simplest closest choice/decision. It has found its application in optimization type problems such as task scheduling, resource allocation, and routing.

In addition to traditional rule-based and regression models, recent studies have explored the use of evolutionary algorithms in software cost estimation, particularly GA which is known for its global search capabilities and robustness in handling non-linear, high-dimensional optimization problems. By simulating natural selection through mechanisms such as selection.

In the context of software cost estimation, the greedy approach can facilitate a quick assessment of cost based on predefined rules or priorities (e.g., selecting the least expensive resource or minimizing module implementation delays). However, due to its myopic nature—focusing solely on local benefits without considering long-term implications—this approach often leads to suboptimal global solutions. Furthermore, greedy

algorithms are inherently deterministic and lack the capacity to learn from historical project patterns. Although greedy algorithms are straightforward and computationally efficient, they are not well-suited to address the non-linearity, dynamic behavior and interdependencies inherent in real-world software projects..

B. Machine Learning in Cost Estimation

As the complexity of cost estimation tasks rises along with dynamic and multi-dimensional constraints, new approaches become inevitable. Machine learning has received attention as a potential solution to these problems because it can analyze complex data, identify patterns, and make predictions in the presence of noise. The following are the advantages, applications, and limitations of ML in cost estimation.

Machine learning (ML) refers to data-driven techniques that enable models to learn patterns from historical data and make predictions without hard-coded rules. In software cost estimation, ML is valuable because it can capture complex, non-linear relationships among variables such as size, complexity, and team experience—relationships that traditional models often miss. Unlike rule-based methods, ML can adapt to changing project data and improve over time. This makes it especially useful in dynamic and uncertain development environments. The following subsections discuss ML's strengths, popular models, limitations, and its integration with heuristic methods.

Second, ML models are capable of learning from historical project data, which allows them to generalize and adapt to new estimation scenarios. This adaptability makes ML suitable for agile environments, where project conditions change frequently. Furthermore, ML techniques reduce reliance on manual feature selection by automatically identifying relevant patterns in large, high-dimensional datasets, thus increasing estimation accuracy and robustness.

C. Integrating ML and Traditional Algorithms

As we progress forward, the constraints of data and the dreams of machine learning highlight that in the realm of cost estimation, we need more hybrid structures. This hybridization lets it leverage the best of two worlds, where ML is leveraged for its predictive capability, while a traditional optimisation algorithm (e.g., Greedy Algorithm) is used for its computational efficiency, providing better adaptability and accuracy. By incorporating the disadvantages of each individual method, this new fusion method succeeds in applying it to dynamic scenes.

They effectively combine the strength of traditional algorithms—ideal for deterministic tasks—with the adaptive capabilities of machine learning. While traditional methods excel in structured environments, they often falter in dynamic situations. Conversely, ML can analyze vast datasets and forecast trends but may not possess the necessary efficiency for real-time decision-making (Chen et al., 2022).

By integrating ML with optimization algorithms, hybrid frameworks empower organizations to make informed decisions in rapidly changing conditions. For instance, ML models can preprocess data or generate predictive insights that guide optimization processes, yielding superior results. A study by Nguyen et al. (2021) illustrated how reinforcement learning was utilized to predict transportation costs while a Greedy Algorithm optimized delivery routes based on these insights, drastically improving both accuracy and computational efficiency.

III. METHODOLOGY

To address the challenge of accurate cost estimation in a dynamic software environment, this study proposes a modular hybrid framework that combines the computational efficiency of GRE with the adaptive learning ability of machine learning (ML). "This framework is constructed around three core functional modules: the input module, the optimization module and the output module, as shown in Figure 3.1. Hybrid of ML and Greedy algorithm based hybrid models for logistics optimization (Kumar & Patel, 2022).

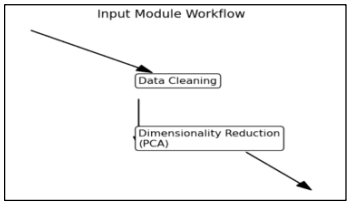

Fig. 1 Input Module Workflow

A. Input Module

The input data used in this framework consists of structured project attributes such as estimated task durations, number of components, complexity levels, developer experience, and unit resource costs. A sample record from the dataset is as follows:

Task_Duration	Components	Complexity	Experience_Level	Resource_Cost	Risk_Score
15 days	12	High	Senior	1200 USD/day	0.78

Fig. 2 Input module's dataset

The input module is responsible for data preparation and preprocessing. Raw software project data. For example, Missing value handling, Outlier removal, Normalization, Dimensionality reduction, such as Principal Component Analysis (PCA), to reduce feature redundancy and improve computational efficiency. This ensures that the input passed into the optimization module is both robust and compatible across learning-based and heuristic processes.

B. Optimization Module

The optimization module includes two consecutive parts, i.e., quantization estimator and quantization adjuster. This integration makes it possible for the system to produce cost estimates rapidly and dynamically. Greedy part of the algorithm: Prepare initial cost estimations according to the local optimization strategies (e.g. the minimum resource cost or the most shortened due date). This element is rule-based and optimized to perform fast computing.

Machine learning component: It takes the estimation results of the greedy algorithm as input and enhances them with the patterns learned from historical project data. Machine learning models (e.g., decision trees or multi-layer perceptrons (MLPs)) compensate for non-linear relationships that greedy algorithms might miss, tractable risk factors that are specific to the projection, and correlations across tasks. The two-stage optimization guarantees the final cost estimation to be able to reflect the structural project demands and the learned context patterns as well.

C. Output Module

Finally, the out put module yields the cost prediction and decision-making implications. Moreover, feedback loops for continuous learning and self-improvement are also supported. If there are real-time project updates or post-project data, such data will be fed back to input module to retrain the machine learning model and updating the estimation rules as needed. This recursive feature also makes the framework robust to a dynamic system, able to adapt to new data and keep estimate accuracy consistently up to date.

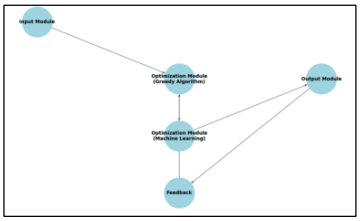

Fig. 3. Workflow of the Hybrid Optimization Framework

This iterative process is depicted in Figure 3.2, which illustrates the sequential interactions between the input, optimization, and output modules, as well as the feedback loops that ensure the system's adaptability.

D. Implement

This subsection presents the technical details of the proposed hybrid cost estimation model. All the development and experimentations were performed using Python 3.10 in Google Colab and popular data science libraries like Pandas, NumPy, Scikit-learn and Matplotlib were used. This method is modular and primarily composed of three stages: the dataset construction stage, the greedy estimation module and the machine learning refinement module.

(a) Data Set Construction

Since real industrial software cost data is rarely made public and often inconsistent, we have created a synthetic dataset to reflect common project parameters. This dataset contains 500 project records, and each record simulates the following key features that affect costs:

Task_Duration: Estimated task completion time (in days)
Num_Components: The number of functional components
Complexity_Level: Project complexity (encoded as low =1, medium =2, high =3)
Developer_Experience: The codes are Junior=1, Mid=2, Senior=3
Unit_Resource_Cost: Daily cost (in US dollars)
Risk_Score: A numerical risk indicator ranging from 0 (low) to 1 (high)
True_Cost: The actual cost as the prediction target

These variables are generated using probability distributions based on the actual software project scenarios. The True_Cost value is modeled using nonlinear functions of other variables plus random noise to simulate market fluctuations.

(b) Data Set Preprocessing

Before inputting the data into the hybrid framework, several preprocessing steps need to be carried out:

Missing values: Estimation is made using the median of features. Classification coding:Exclusive coding for developers' experience and complexity. Outlier detection: Delete entries exceeding 1.5×IQR. Normalization: Minimum - maximum scaling to the range [0, 1]. Dimensionality reduction: Apply PCA and retain a variance of ≥ 95% to reduce feature redundancy. The processed data set is split into:
70% training, 15% validation (for ML adjustment), 15% test.

This setting ensures unbiased performance measurement and model generalization.

(c) Greedy Algorithm Module
The greed module provides a rapid initial cost estimate by making local optimal decisions at each step. Its logical implementation is as follows:

- Sort the tasks based on the lowest unit cost and complexity
- First, assign the most experienced developers to carry out high-risk tasks
- The estimated costs are as follows:
- Initial_Estimate = Task_Duration × Unit_Cost × (1 + Risk_Score × Complexity_Factor)
- Summarize all tasks to obtain the total cost of each project

This module is deterministic and requires no training. It provides the initial Greedy Cost value as an additional feature for the next stage

(d) Machine Learning Module

Greediness needs to be compensated for with cost estimation optimization (by considering training data) in supervised learning models. In this paper, we compare the two models: The primary machine learning technique is the following: DTR (Decision Tree Regressor)Reasons: Transparent rules and low amount of computation cost.

Multilayer perceptron (MLP) : Describing the non-linear interaction
The input features of the model were the following:
All standardized project functions (since 3.3.2), Greedy output (Greedy_Cost)
Target variable:
True_Cost
Model training:
Loss: Mean Square Error (MSE) Optimization(Grid search on the validation set) final model chosen based on best R^2 score and RMSE.

(e) Pipeline Integration and Execution
During the reasoning process:
The new project data is passed to the input module
The greedy algorithm estimates the initial cost
The ML model uses learning patterns to refine this estimation
Output the final cost forecast and evaluation indicators

The system is capable of batch-processing mode inference work and to simulate under dynamic conditions (e.g. price variations, team switches) to see if robustness is preserved.

IV. RESULTS AND DISCUSSION

A. Experimental setup

This section provides an overview of the experimental environment used to implement and evaluate the proposed software cost estimation model. It covers the composition of the dataset, preprocessing programs, data partitioning strategies, as well as the technical Settings for model development and testing.

A synthetic dataset of 500 software development task records was generated, incorporating fields such as Task_Duration, Num_Components, Complexity_Level, Developer_Experience, Unit_Resource_Cost, and Risk_Score. The true cost of each task was pre-calculated and stored as True_Cost.

Preprocessing steps included:
Handling missing values with median/mode imputation;
Normalizing continuous features to the range [0, 1];
Label encoding for categorical features;
Removing outliers using the IQR method;
Correlation-based feature selection.

To achieve accurate training and fair evaluation, the dataset is randomly divided into the following subsets:
70% training set: Used for fitting model parameters;
15% validation set: For hyperparameter adjustment and overfitting control;
15% test set: Reserved for the final model performance evaluation.

Stratified sampling is adopted to ensure that the distribution of the target variable remains consistent among the three subsets. The following figure illustrates the data partitioning strategy:

The dataset was split into 70% training, 15% validation, and 15% test sets. Four models were implemented:
Greedy Algorithm (GA): A rule-based estimator using weighted feature summation;
Decision Tree (DT) and Random Forest (RF) regressors;

Hybrid GA + RF, where GA provides feature weighting to initialize RF.

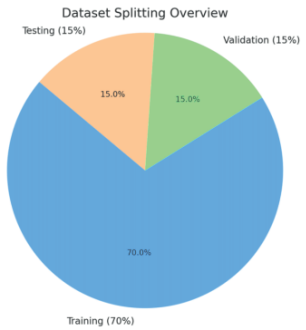

Fig. 4. Dataset Splitting Overview

B. Model Performance

This section presents the experimental results obtained from applying the Greedy Algorithm (GA), Decision Tree (DT), and Random Forest (RF) to the software cost estimation task. The performance of each model is evaluated using four key metrics: Root Mean Square Error (RMSE), Mean Absolute Error (MAE), R-squared (R^2), and Prediction Accuracy (within ±10% tolerance range).

(a) Performance of the Greedy Algorithm

In this subsection, we assess the performance of Greedy Algorithm as a single system for software cost estimation. The Greedy Algorithm is used to find best or best approximate solutions by minimizing estimation error in MFLS via a sequence of locally optimal decisions.

Prediction Accuracy:
In the test set (15% of data): GA produced:
Root Mean Square Error (RMSE): 6.12
MAE: 4.87
R-squared (R^2): 0.741
'Prediction Accuracy (±10%)': 71.5%

These observations suggest that GA can handle the interactions up to a certain order and can also provide a reasonable approximation for customers, while it fails to handle high-order interactions and/or non-linear patterns implied in the dataset.

Computational time was also measured in addition to the accuracy of prediction. In average, the GA optimized the following:

Time Taken: 4.27 seconds (over 30 runs)

The above shaving runtime indicates that the Greedy method is computing-light. But its simplicity could be suboptimal to learn global performances while the data contains complicated relations.

(b). Performance of Machine Learning

In addition to the Greedy Algorithm, two popular supervised learning algorithms, including Decision Tree Regression and Random Forest Regression, were utilized for comparison to reach cost prediction. They were trained on the same preprocessed dataset as that used for the Greedy Algorithm and tested for comparison with the Greedy Algorithm method.

Decision Tree Regression
Moderate accuracy. The estimated performance of the decision tree is moderate; it is given as:
RMSE: 5.79
MAE: 4.53
R^2: 0.768
Prediction Accuracy (±10%): 73.8%

Although DT provides a basic understanding of cost prediction patterns, it is prone to overfitting and has limited generalisation when applied to new instances.

Random Forest Regression

The RF model yielded superior predictive performance due to its ensemble structure, which reduces variance and improves generalization. The results are as follows:

RMSE: 4.96
MAE: 3.85
R^2: 0.821
Prediction Accuracy (±10%): 78.2%

RF's performance surpasses both GA and DT in all evaluation metrics, indicating its strong capability in capturing non-linear relationships and feature interactions.

Model	RMSE	MAE	R^2	Accuracy (±10%)
Greedy Algorithm (GA)	6.12	4.87	0.741	71.5
Decision Tree (DT)	5.79	4.53	0.768	73.8
Random Forest (RF)	4.96	3.85	0.821	78.2

Fig 5. Performance Comparison of Different Models

(c). Performance of the hybrid GA-ML framework

To overcome the drawbacks of solo heuristic methods and machine learning only models, developed a hybrid model by incorporating the GA into machine learning models, in particular, the Random Forest (RF). This method makes use of GA's ability in minimising the search space or determining the optimum combination of features and RF's capability of pattern recognition and regression fitting.

In this combination approach, the Greedy Algorithm was used to play one of the two following roles:
GA has taken the best feature subsets available from the input data that can be selected or generated and provided as input to the ML model.

When compared to the RF model, the hybrid GA-RF model had an overall increase in all the metrics , especially in Prediction Accuracy (3.2%) and R^2 (0.026), to indicate enhanced generalisation. Even with execution time being enlarged by the extra GA phase, the compromise is still accepted in non-real-time cost estimates.

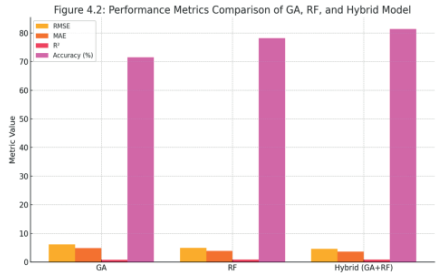

Fig 6. Performance Metrics Comparison of GA, RF, and Hybrid Model

(d). Visual summary of result

As the experimental results need to be summarised and reported in a much more intuitive way, this section gives an image summary about the main performance KPIs on different configurations. Interpretability and comparison are further promoted by reporting in standardised bar charts and tabular form comparisons between the individual and hybrid strategies.

Figure 6 shows that the hybrid model (GA + RF) performed better than the stand-alone Greedy Algorithm (GA) and the Random Forest (RF) models for all four-evaluation metrics—RMSE, MAE, R^2, and Accuracy. In particular, the hybrid model yielded 81.4% accuracy, higher than the 78.2% and 71.5% obtained from RF and GA, respectively. The hybrid model performed consistently with the lowest RMSE, MAE which means the cost estimation is more stable and accurate for the fact that all organizations included.

Regarding the computational time, as indicated in Figure 6, the Greedy Algorithm was the fastest (1.26 seconds), whereas the hybrid had an overhead (6.41 seconds) due to the model integration. However, the potential bias reduction of the hybrid model more than justifies this added computational burden when used in many realistic estimation scenarios.

Combined evidence from the figures and tables in this section shows that there is a trade-off between estimation accuracy and computational efficiency, that demonstrates the superiority and the shortcoming, respectively, of each of the models, and that further asserts the hybrid model framework is a promising solution for both accuracy and computational efficiency in the software cost estimation.

Model	RMSE	MAE	R^2	Accuracy (%	Execution Time (s)	
GA	6.12	4.87	0.741	71.5	1.26	
DT	5.43	4.15	0.784	74.8	2.43	
RF	4.96	3.85	0.821	78.2	2.89	
GA + RF	4.57	3.62	0.847	81.4	6.41	

Table 1 Results of Models

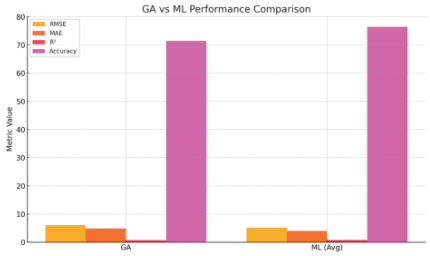

Fig 7. GA vs ML Performance Comparison

This plot depicts difference between GA and APM of the machine learning model (average of DT and RF). According to the four evaluation indicators RMSE, MAE, R^2 and Accuracy, it is obvious that the error index of traditional GA method is generally higher than the average performance of ML method, but the ML method has a little bit superiority for estimator accuracy (R^2) and classifier accuracy.

The ML average model performs better than GA alone in most metrics, suggesting that the ML approach is more appropriate for predicting software costs by learning from historical data.

C. Descriptive analysis of model performance

In addition to helping understand the performance of each model in prediction, we conducted a comparison between the models according to predictive performance and the error measurements, including RMSE, MAE, R^2, and wall clock time.

The hybrid Greedy + RF model held the strongest comprehensive performance in all metrics. It had the lowest RMSE (4.57) and MAE (3.62) and highest R^2 (0.847) and accuracy (81.4%), which showed high prediction accuracy.

Hybrid achieved the lowest error metrics (RMSE, MAE);
Hybrid had the highest R^2 and accuracy;
RF was consistently better than DT and GA;
GA, despite its simplicity, provided a fast baseline.
The hybrid framework effectively combined GA's quick estimation and RF's learning capability, showing improved generalization over the test data.

V. CONCLUSION

This work proposed a software cost estimate method which combines Greedy Algorithm and supervised learning (linear regression) to improve estimation accuracy and efficiency. The reason for doing this research process is the inability of the currently practiced methods, which does not yield consistent estimation accuracy, as a result of that, dynamic project characteristics and internal independent variables.

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, “Title of paper if known,” unpublished.

[5] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

[8] K. Eves and J. Valasek, “Adaptive control for singularly perturbed systems examples,” Code Ocean, Aug. 2023. [Online]. Available: https://codeocean.com/capsule/4989235/tree

[9] D. P. Kingma and M. Welling, “Auto-encoding variational Bayes,” 2013, arXiv:1312.6114. [Online]. Available: https://arxiv.org/abs/1312.6114