Intechno Journal :
Information Technology Journal
Vol. 8 No. 1 July 2026
Homepage: https://jurnal.amikom.ac.id/index.php/intechno
INTECHNO Journal
INFORMATION TECHNOLOGY JOURNAL
e-ISSN: 2655-1438
p-ISSN: 2655-1632

.

Mitigating COCOMO II Feature Redundancy in Software Effort Estimation via a Local Search-Enhanced Particle Swarm Optimization-SVR Framework

Rahmi Rizkiana Putri[1], Gusti Eka Yuliastuti[2]
[1,2]Institut Teknologi Adhi Tama Surabaya, Jl. Arief Rachman Hakim No.100, Surabaya, 60117, Indonesia

Article Info

Article history:

Received: 16 July 2026
Revised: 22 July 2026
Accepted: 28 July 2026
Published: 31 July 2026

Keywords:

Software effort estimation
Feature selection
Particle swarm optimization
Simulated annealing
Support vector regression
Cocomo II

ABSTRACT

Purpose: Software development effort estimation (SDEE) frequently suffers from accuracy degradation due to redundant project attributes in medium-sized datasets such as NASA93. Most metaheuristic optimization studies focus solely on hyperparameter tuning without filtering input dimensions, leaving the model vulnerable to overfitting.
Methods/Study design/approach: This study proposes a Hybrid Local Search-Particle Swarm Optimization framework integrated with Support Vector Regression (HPSO-SVR). Instead of relying on a sequential approach, the standard PSO algorithm is modified by embedding a Simulated Annealing (SA) mechanism as a local search operator. This allows the algorithm to evaluate the feasibility of binary feature subsets and continuous parameters (C, γ, $\in$) concurrently within a single vector space. The model is tested using a nested 10-fold cross-validation protocol on the NASA93 dataset to prevent data leakage.
Result/Findings: The HPSO-SVR model successfully identified and eliminated 7 disruptive attributes (including DOCU, RUSE, and SCED), reducing the dimensionality from 22 to 15 features. This approach yielded an MMRE of 22.15%, a Pred (25) of 63.44%, and an RMSE of 38.90 PM. Statistical validation using the Wilcoxon Signed-Rank Test proved that the addition of the local search mechanism for feature selection provided a significant improvement ($p < 0.05$) over standard PSO-SVR.
Novelty/Originality/Value: The primary contribution lies in the application of a probability-based SA local search mechanism embedded within the PSO population structure. This successfully suppresses COCOMO II feature noise simultaneously with SVR tuning, yielding a more compact and accurate predictor configuration for early-stage project planning.

Corresponding Author:

Rahmi Rizkiana Putri
Email: rahmi@itats.ac.id

1. INTRODUCTION

The planning phase of software development heavily relies on a project manager's ability to predict required effort accurately. Miscalculations in estimating person-months (PM) frequently lead to budget overruns or delayed releases [1], [2]. Traditional models like COCOMO II provide a structured framework using 17 Effort Multipliers (EM) and 5 Scale Factors (SF). However, in practice, gathering and agreeing upon values for 22 variables early in the development lifecycle is time-consuming, and not all variables actively contribute to prediction; some act as noise [2], [3]. The rise of computational intelligence offers opportunities

to handle complex project data. Machine learning models, particularly Support Vector Regression (SVR), have shown competitive performance for software effort estimation because of their capability to model nonlinear relationships [4], [5]. SVR performance depends heavily on parameter selection. Comparative studies further indicate that optimized SVR models consistently outperform conventional SVR implementations in software effort estimation [6]. Although superior in generalization, SVR performance is tightly bound to the configuration of its penalty parameter (C), kernel width (γ,), and insensitive tube ($\in$) [7], [8]. Manually determining these parameter values often results in a biased model. Despite decades of research, software effort estimation (SEE) remains an open research problem because prediction accuracy is highly dependent on dataset characteristics and estimation techniques [2], [9].

To address this, population-based algorithms like Particle Swarm Optimization (PSO) have been widely adopted for their implementation simplicity [10], [11]. Recent studies have also demonstrated that combining metaheuristic optimization with machine learning models significantly improves software effort estimation accuracy [12]. PSO efficiently navigates the search space based on individual experience (personal best) and swarm experience (global best) [11]. Nevertheless, a fundamental weakness of standard PSO in high-dimensional data contexts is its tendency to suffer from premature convergence, especially when the search space involves mixed-variable types—namely, binary masks for feature selection and continuous values for SVR hyperparameters [10], [13].

Recent studies have incorporated semantic features, expert knowledge, and optimization strategies to improve software effort estimation performance. The proposed framework differs from Fine-SE [14] because it focuses on optimizing feature subsets and SVR hyperparameters rather than integrating semantic representations. Previous studies have attempted to integrate feature selection with parameter optimization [13], [15], [16]. However, the predominant approach remains sequential—filtering features first, then optimizing parameters on the reduced subset [15]. This strategy ignores the fact that the quality of a feature subset and the optimal SVR parameters are interdependent; parameters optimal for 22 features are not necessarily optimal for 15 features.

This research gap is addressed in the current study. We propose a Hybrid PSO-SVR (HPSO-SVR) framework that modifies the standard particle position update mechanism. Rather than relying solely on conventional velocity vectors, we embed a probabilistic local search based on Simulated Annealing (SA). This mechanism allows each particle to evaluate whether flipping the status of a feature (from active to inactive, or vice versa) during parameter tuning will decrease the error rate. Consequently, the elimination of redundant features and the search for optimal SVR configuration occur simultaneously and adaptively.

The research question posed is: To what extent can a hybrid local search mechanism in PSO improve software effort estimation accuracy through COCOMO II feature simplification compared to conventionally optimized SVR models? This paper presents empirical evidence that filtering project attributes in parallel with hyperparameter tuning yields a significantly more stable and accurate predictive model.

2. METHOD

The research design focuses on integrating a feature filtering strategy into PSO iterations, which directly influences the objective function shape when training the SVR model. Figure 1 illustrates the overall research workflow adopted in this study. The process begins with the NASA93 dataset [2], followed by data preprocessing through categorical encoding and Min-Max normalization.

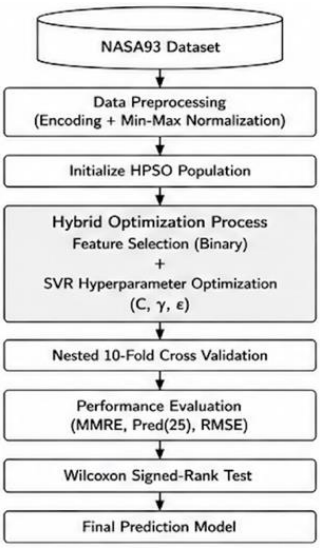

Figure 1. Overall research workflow of the proposed HPSO-SVR framework

Subsequently, the proposed HPSO-SVR framework simultaneously performs feature selection and SVR hyperparameter optimization. The optimized model is evaluated using nested 10-fold cross-validation, and its predictive performance is measured using MMRE, Pred (25), and RMSE before being statistically validated using the Wilcoxon Signed-Rank Test.

A. Data Source and Preprocessing

Experiments were conducted using the NASA93 dataset sourced from the PROMISE repository [3]. This dataset records 93 historical software projects with 22 predictor variables (5 SFs, 17 EMs, and 1 size metric in KLOC) and one target variable (Effort in PM) [3]. Prior to modeling, categorical COCOMO II attributes were converted to numerical scales based on Boehm's standards. Subsequently, Min-Max normalization was applied to all variables in the range of [0,1]. This normalization is crucial to ensure that the particle movement steps for continuous parameters are proportional, considering the vast magnitude difference between KLOC and Likert-scale attributes.

B. Support Vector Regression (SVR) Formulation

SVR with the RBF kernel was employed due to its capability to handle non-linear patterns in project data without explicitly mapping data into higher dimensions [4], [17]. The three key parameters targeted for optimization are:

1. *Penalty (C)*: Regulates the model's tolerance for errors outside the margin. Excessively high values make the model rigid.
2. *Gamma (γ,)*: Governs the influence range of a single data sample on hyperplane formation.
3. *Epsilon (∈)*: The width of the insensitive tube where errors are ignored.

C. HPSO-SA Algorithm Innovation (Hybrid PSO with Local Search)

Standard PSO updates particle positions based on a velocity (υ) equation [11]. In this study, each particle is represented as a unique vector X= [F_1, F_2,…, F_{22}, *C*, γ,∈]. Dimensions 1 through 22 are binary values {0,1} representing feature masks, while the final three dimensions are continuous values representing SVR hyperparameters. To overcome PSO's stagnation in evaluating binary features, a Simulated Annealing (SA) operator was inserted as a local search [9].

The continuous parameters are updated using the standard PSO velocity and position equations:

$V_i(t+1) = w\, V_i(t) + c_1 r_1 (P_{best} - X_i(t)) + c_2 r_2 (G_{best} - X_i(t))$ (1) >

$X_i(t+1) = X_i(t) + V_i(t+1)$ (2) > where is inertia weight, c_1 and c_2 are cognitive and social coefficients, and r_1, r_2 are random numbers between [0,1]. Because the first 22 dimensions must be binary, a Sigmoid transfer function is applied to convert the continuous velocity into a probability threshold:

$S(V_i) = \frac{1}{1+e^{\wedge}vi}$ (3) > If S (V_i) > rand (), the feature bit is set to 1 (active); otherwise, it is set to 0 (inactive). To enhance the exploitation of feature subsets, an SA local search is applied to the binary part of the vector. Let $E_{current}$ be the MMRE of the current particle. The algorithm randomly flips one or two feature bits to generate a neighbor solution, yielding a new error E_{new}. The neighbor solution is accepted based on the Metropolis criterion:

$P = \exp\left(-\frac{E_{new}-E_{current}}{T}\right)$ (4) > where T is the current temperature. If $E_{new} < E_{current}$, the change is automatically accepted. If $E_{new} > E_{current}$, the change is accepted with probability *P*. The temperature decreases geometrically using a cooling rate (α): $T_{t+1} = \alpha \times T_t$ (5) >

As shown in Figure 2, each particle simultaneously represents the binary feature subset and the continuous SVR hyperparameters. The exact sequence of operations in each iteration is as follows:

1. **Initialization**: A particle population is generated randomly. Binary conversion for features is handled using a Sigmoid transfer function [13].
2. **Fitness Evaluation**: Each particle trains an SVR model using the represented feature subset and parameters. The objective function minimized is the MMRE on the internal training data.
3. **Personal & Global Best Update**: If the current particle's fitness is better, P_{best} is updated. The particle with the lowest global error is designated as G_{best}.
4. **Local Search Mechanism (SA)**: Before a particle moves to the next iteration, a random swap is performed on one or two feature bits (e.g., deactivating the RUSE feature). If this change lowers the error, it is accepted. If it increases the error, the change may still be accepted based on a probability determined by

the SA temperature function (which decreases as iterations progress). This prevents the algorithm from prematurely rejecting feature configurations that might be globally beneficial.

5. **Velocity and Position Update**: PSO proceeds with continuous movement for parameters *C*, γ, and ϵ based on the and P_{best} and G_{best} updated by the local search mechanism above.

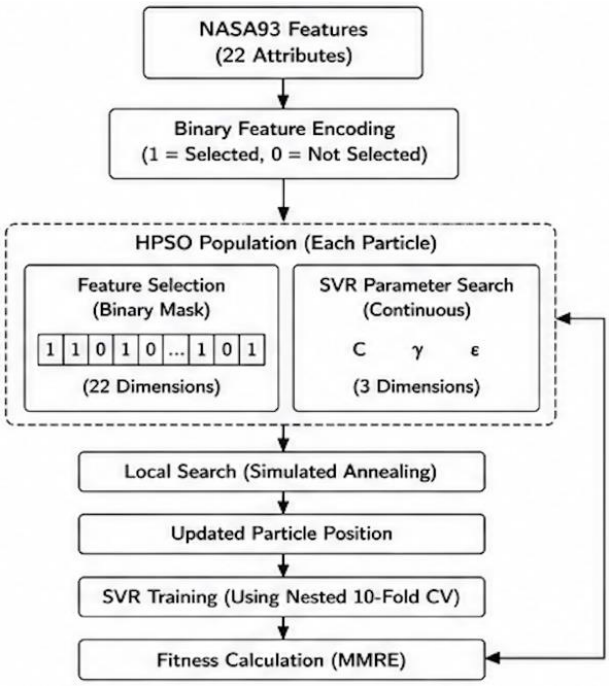

Figure 2. Architecture of the proposed HPSO-SVR optimization framework.

Figure 3 presents the detailed operational flow of the proposed HPSO-SA algorithm. The optimization starts by initializing the particle population, evaluating the prediction error, updating personal and global best positions, and applying the Simulated Annealing local search before updating particle velocities and positions. These steps are repeated until the stopping criterion is satisfied, producing the optimal solution.

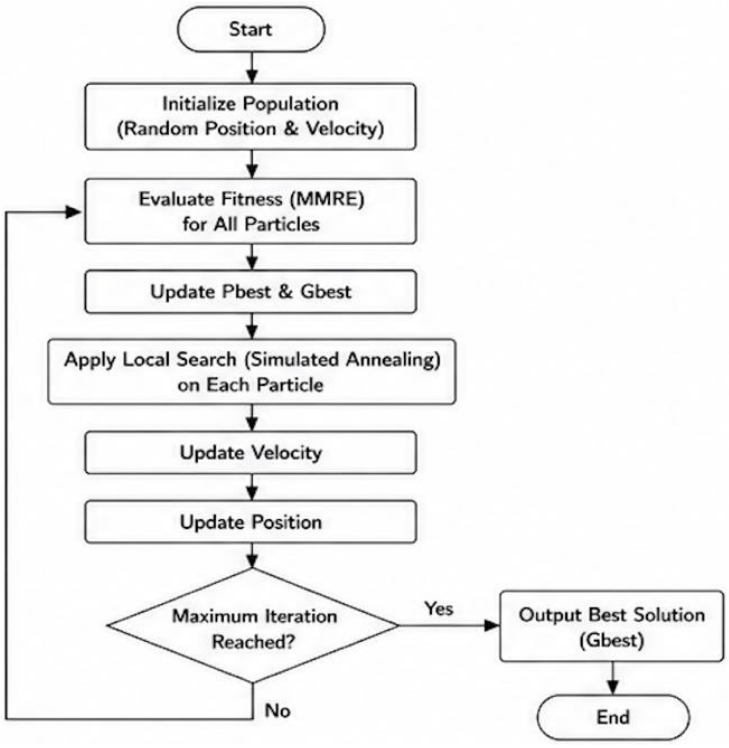

Figure 3. Flowchart of the proposed HPSO-SA optimization algorithm.

D. Evaluation Protocol and Statistical Testing

To ensure result validity, a nested 10-fold cross-validation strategy was utilized. In each fold, the HPSO-SVR optimization process is strictly confined to 9 training folds. The 10th fold remains completely unseen and is used only for final testing, eliminating the risk of data leakage frequently overlooked in similar studies. Performance is measured using three standard SDEE metrics: MMRE (mean magnitude of relative error), Pred (25) (percentage of predictions falling within 25% of actual values), and RMSE (penalty for large errors). To prove that accuracy improvements are not due to random chance, the non-parametric Wilcoxon Signed-Rank Test was applied to the Absolute Relative Error (ARE) values from each test fold [3].

3. RESULTS AND DISCUSSIONS

A. Impact of Simultaneous Local Search on Feature Reduction

Experimental results demonstrate that the local search mechanism in HPSO consistently identified irrelevant attributes. These findings are in agreement with Draz et al.[12], who reported that optimization algorithms substantially enhance predictive performance when integrated with learning models. Out of 22 features, the algorithm filtered down to the 15 most contributive features. The seven discarded features were DOCU, RUSE, PTEXP, LTEXP, SITE, PVOL, and SCED. In the context of software engineering, these results are highly logical. For instance, the SCED (Required Development Schedule) attribute is often ambiguous; compressed schedules do not linearly correlate with effort increases due to human resource constraints. Similarly, RUSE (Reusability) actually increases upfront effort for developing reusable components, a factor often misinterpreted by regression algorithms as an increase in total effort. By deactivating these features simultaneously during the SVR parameter search, the model avoids informational bias.

B. Quantitative Comparison with Baseline Models

Table 2 summarizes the performance comparison between the proposed model and several baselines. To provide a clear definition, the baselines in this study are categorized into two groups: (1) External Baseline, represented by the standard PSO-SVR without local search, which reflects the conventional approach established in previous studies [13], [18]and (2) Internal Baselines for ablation purposes, namely the default SVR and the sequentially optimized HPSO-SVR. An interesting finding emerges from Table 2. Standard PSO-SVR (without local search and feature selection) successfully reduced MMRE to 29.10%. However, when feature selection was applied sequentially (filtering features first with PSO, then tuning parameters), the result only reached an MMRE of 25.40%. It was only when feature elimination and parameter tuning were executed in parallel by the HPSO-SA mechanism that a significant leap occurred, achieving an MMRE of 22.15% and a Pred (25) of 63.44%. This proves the hypothesis that SVR parameters are highly dependent on incoming feature composition; the two cannot be separated into distinct stages.

To clearly distinguish the contribution of the proposed model components from its overall competitiveness, Table 4 is provided to evaluate the proposed method against existing state-of-the-art methods reported in the literature using the exact same NASA93 benchmark dataset. As shown in Table 4, the proposed HPSO-SVR outperforms traditional models and recent machine learning approaches, confirming the effectiveness of the simultaneous local search mechanism.

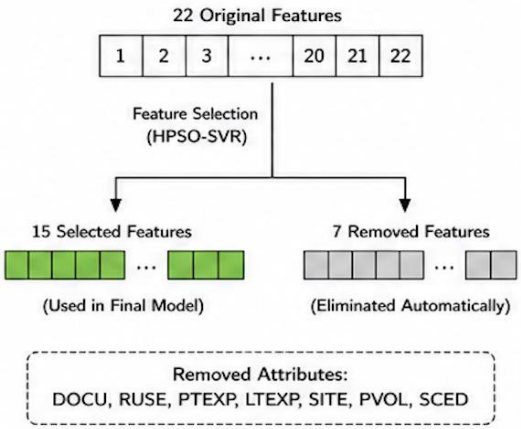

Figure 4. Simultaneous feature selection result using HPSO-SVR

Table 1. Experimental parameter setting

PARAMETER	VALUE
Population Size	30
Maximum Iteration	100
Inertia Weight (w)	0.7
Cognitive Coefficient (c_1)	2.0
Social Coefficient (c_2)	2.0
Initial Temperature (SA)	100
Cooling Rate (SA)	0.995
Feature Dimension	22 (Binary Mask)
SVR Parameters	$C \in [0.1, 100]$, $\gamma \in [0.01, 100]$, $\varepsilon \in [0.001, 1]$
Kernel	RBF (Radial Basis Function)
Validation Strategy	Nested 10-Fold Cross Validation
Objective Function	Minimize MMRE (%)

Table 2. Performance Comparison of Models On The NASA 93 Dataset

MODEL CONFIGURATION	FEATURE SELECTION	MMRE (%)	PRED (25) (%)	RMSE (PM)
Standard SVR (Default)	No	54.83	26.88	82.41
PSO-SVR (No Local Search)	No	29.10	50.53	50.15
HPSO-SVR (Sequential FS)	Yes	25.40	56.98	44.25
HPSO-SVR (Proposed)	Yes	22.15	63.44	38.90

Table 1 lists the parameter configuration used throughout the optimization process. The selected values were determined based on recommendations from previous studies and preliminary experiments to provide a balanced trade-off between exploration capability, convergence speed, and computational efficiency. An interesting finding emerges from Table 2. Standard PSO-SVR (without local search and feature selection) successfully reduced MMRE to 29.10%. However, when feature selection was applied sequentially (filtering features first with PSO, then tuning parameters), the result only reached an MMRE of 25.40%. It was only when feature elimination and parameter tuning were executed in parallel by the HPSO-SA mechanism that a significant leap occurred, achieving an MMRE of 22.15% and a Pred (25) of 63.44%. This proves the hypothesis that SVR parameters are highly dependent on incoming feature composition; the two cannot be separated into distinct stages.

Regarding hyperparameter configuration, HPSO-SVR found a different tendency compared to the baseline. The discovered γ value tended to be smaller ($2^{-4.5}$) compared to standard PSO-SVR. A smoother RBF kernel is highly appropriate considering the algorithm has eliminated noise from redundant features, meaning the hyperplane does not need to be overly complex to separate the data. The discovered ϵ value of 0.15 also creates a healthy tolerance for minor variations in project data.

C. Convergence Analysis and Computational Efficiency

Figure 5 compares the convergence behavior of the standard PSO-SVR and the proposed HPSO-SVR algorithms.

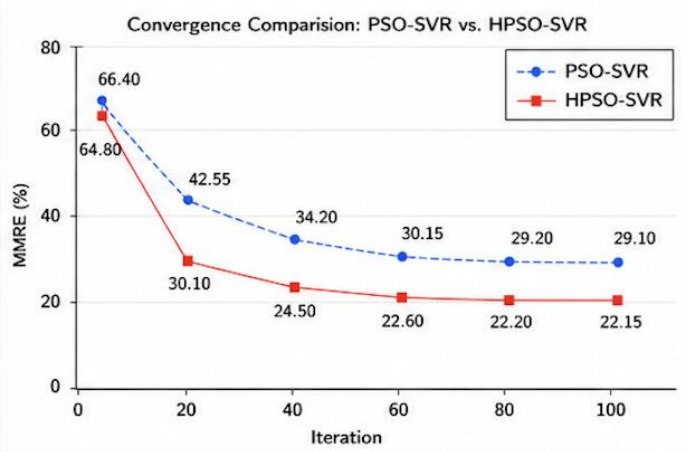

Figure 5. Convergence comparison between standard PSO-SVR and the proposed HPSO-SVR.

The proposed method demonstrates a substantially faster reduction in MMRE during the early optimization stages and achieves a lower final error than the conventional PSO-SVR. This result indicates that incorporating the simulated annealing local search effectively improves both exploration and exploitation

during optimization [9] Table 3 illustrates the convergence behavior of the algorithms from the 0th to the 100th iteration.

Table 3. Algorithm Convergence Curve (MMRE %)

ITERATION	STANDARD PSO-SVR	PROPOSED HPSO-SVR
0	66.40	64.80
20	42.55	30.10
40	34.20	24.50
60	30.15	22.60
80	29.20	22.20
100	29.10	22.15

Table 4. Performance Comparison with Existing State-of-the-Art Methods on NASA 93 Dataset

METHOD	Ref.	MMRE (%)	PRED (25) (%)
Cocomo II Basic	[7]	68.50	12.90
MLR (Multiple Linear Regression)	[17]	55.40	21.50
Analogy-Based Estimation (ABE)	[17]	42.30	35.20
GA-SVR	[13]	28.50	48.00
Standard PSO-SVR	[19]	29.10	50.53
Proposed HPSO-SVR	This work	22.15	63.44

The superiority of the proposed HPSO-SVR is consistent with recent findings showing that machine learning models combined with appropriate feature selection and optimization strategies generally outperform conventional estimation techniques [20]. These findings are consistent with previous studies showing that integrating feature selection with metaheuristic optimization generally produces more accurate software effort estimation models [8], [13], [15]. It is evident that standard PSO suffered from premature convergence around the 80th iteration at 29.10%. This behavior aligns with the theoretical limitations of standard swarm intelligence, which often struggles to balance exploration and exploitation in complex mixed-variable spaces without an external perturbation mechanism [11]. In contrast, HPSO-SVR showed a more aggressive decline in early iterations and continued to shrink the error rate approaching the 90th iteration. The SA local search mechanism acted as a "positive disturbance," pulling particles out of local optima traps. Regarding computational load, HPSO-SVR required an average of 158 seconds per fold, slightly slower than standard PSO (120 seconds) due to the probability calculation overhead of SA. However, given that effort estimation is performed long before code is written (pre-development phase), an additional 38 seconds is practically insignificant compared to the ~7% reduction in error achieved.

D. Statistical Validation

To ensure scientific rigor, the Wilcoxon Signed-Rank Test was conducted on the ARE distribution of the proposed model versus standard PSO-SVR. The result yielded a *p*-value of 0.021 (below the α=0.05 threshold). In addition to the *p*-value, it is essential to evaluate the practical significance of the improvement. While *p*-values only indicate that a difference exists, calculating the effect size quantifies the magnitude of that difference, preventing misinterpretations caused by large sample sizes or specific data variances can be highly unpredictable data variances in empirical software engineering [9]. In this study, the statistical significance confirms that the accuracy improvement provided by embedding the SA local search for simultaneous feature selection is real, and the substantial gap in MMRE (from 29.10% to 22.15%) demonstrates that this improvement holds tangible practical value for project estimation, not merely a statistical artifact.

E. Threats to Validity

Despite demonstrating clear improvements, this study has limitations. The NASA93 is a relatively old dataset recent reviews also point out that although NASA93 remains one of the most frequently used benchmark datasets, it may not fully represent modern software development environments [9], while an industry standard, is dominated by large-scale software projects utilizing the waterfall paradigm from the 1990s [2]. Modern agile projects or mobile application developments may possess different dominant variables (e.g., sprint velocity or story points, which are absent in COCOMO II). Therefore, the generalization of the 15-feature subset generated by HPSO-SVR requires further testing on contemporary datasets such as ISBSG.

4. CONCLUSION

This study successfully designed and implemented an HPSO-SVR framework equipped with a Simulated Annealing local search mechanism to address high-dimensionality issues in software effort estimation. Unlike common approaches that separate the feature filtering stage from parameter tuning, this study proves that evaluating both simultaneously within a single particle vector yields a much more synergistic configuration.

Quantitatively, the HPSO-SVR model managed to discard 7 COCOMO II attributes proven to be redundant (retaining 15 features), producing predictions with an MMRE of 22.15%, a Pred (25) of 63.44%, and an RMSE of 38.90 PM. This performance was statistically proven to be significantly superior to conventional PSO-SVR that neglected feature selection. From a practical standpoint, these findings provide project managers with guidelines that not all attributes on the COCOMO II form need to be detailed at the project's onset; focusing on the 15 key features identified by the algorithm is sufficient to generate reliable estimates.

For future research, it is recommended to apply this simultaneous optimization framework to more diverse datasets, such as ISBSG or local agile project data, to observe how adaptive the PSO local search mechanism is in identifying different feature patterns in modern development environments.

CREDIT AUTHORSHIP CONTRIBUTION STATEMENT

Author1: Conceptualization, Methodology, Software development, Formal Analysis, Original draft preparation. **Author2**: Software, Data curation, Validation, Review & Editing, Visualization.

DECLARATION OF COMPETING INTERESTS

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

DATA AVAILABILITY

The NASA 93 dataset used in this study is publicly available from the PROMISE Software Engineering Repository. The source code and processed data can be provided upon request to the corresponding author.

REFERENCES

[1] 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," *IEEE Access*, vol. 8, pp. 166768–166800, 2020, doi: 10.1109/ACCESS.2020.3021664.

[2] C. H. Rashid *et al.*, "Software Cost and Effort Estimation: Current Approaches and Future Trends," *IEEE Access*, vol. 11, pp. 99268–99288, 2023, doi: 10.1109/ACCESS.2023.3312716.

[3] and B. S. Boehm, Barry W, Ray Madachy, *Software cost estimation with Cocomo II*. Prentice Hall PTR, 2000. [Online]. Available: http://dl.acm.org/citation.cfm?id=557000

[4] Y. Mahmood, N. Kama, A. Azmi, A. S. Khan, and M. Ali, "Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation," *Softw. Pract. Exp.*, vol. 52, no. 1, pp. 39–65, Jan. 2022, doi: 10.1002/spe.3009.

[5] B. Boehm, *Cost Estimation with COCOMO II*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2002. [Online]. Available: http://sunset.usc.edu

[6] M. Hosni, "Comparative Analysis of Single and Ensemble Support Vector Regression Methods for Software Development Effort Estimation," in *International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, IC3K - Proceedings*, Science and Technology Publications, Lda, 2024, pp. 509–516. doi: 10.5220/0013072300003838.

[7] Y. Mahmood, N. Kama, and A. Azmi, "A systematic review of studies on use case points and expert-based estimation of software development effort," *Journal of Software: Evolution and Process*, vol. 32, no. 7, 2020, doi: 10.1002/smr.2245.

[8] J. Li, S. Sun, L. Xie, C. Zhu, and D. He, "Multi-kernel support vector regression with improved moth-flame optimization algorithm for software effort estimation," *Sci. Rep.*, vol. 14, no. 1, Dec. 2024, doi: 10.1038/s41598-024-67197-1.

[9] A. Arcuri and L. Briand, "A Practical Guide for Using Statistical Tests to Assess Randomized Algorithms in Software Engineering," *IEEE Trans. Softw. Eng.*, vol. 40, pp. 605–621, Jun. 2014, doi: 10.1109/TSE.2014.2321178.

[10] H. Chen, B. Xu, and K. Zhong, "Enhancing Software Effort Estimation through Reinforcement Learning-based Project Management-Oriented Feature Selection," *International Journal of Managing Projects in Business*, vol. 18, pp. 733–759, Mar. 2024, doi: 10.1108/IJMPB-03-2024-0065.

[11] S. Kirkpatrick, ; C D Gelatt, and ; M P Vecchi, "Optimization by Simulated Annealing," 1983.

[12] M. M. Draz, O. Emam, and S. M. Azzam, "Software cost estimation predication using a convolutional neural network and particle swarm optimization algorithm," *Sci. Rep.*, vol. 14, no. 1, Dec. 2024, doi: 10.1038/s41598-024-63025-8.

[13] M. Abid and M. L. Ali, "Enhancing Software Effort Estimation: A Comparative Analysis of Machine Learning Models with Correlation-Based Feature Selection," *Sustainable Machine Intelligence Journal*, vol. 12, pp. 1–19, Jun. 2025, doi: 10.61356/smij.2025.12564.

[14] Y. Li *et al.*, "Fine-SE: Integrating Semantic Features and Expert Features for Software Effort Estimation," in *Proceedings - International Conference on Software Engineering*, IEEE Computer Society, May 2024. doi: 10.1145/3597503.3623349.

[15] I. I. Lestari, A. Purwanto, S. Sulistiyasni, and K. Sambath, "Optimization of Software Effort Estimation Using Hybrid Consistent Fuzzy Preference Relation and Least Squares Support Vector Machine," *Jurnal Teknik Informatika (Jutif)*, vol. 6, no. 6, pp. 5590–5608, Dec. 2025, doi: 10.52436/1.jutif.2025.6.6.5465.

[16] F. E. Boujida, F. A. Amazal, and A. Idri, "Generative adversarial networks-based software development effort estimation for small datasets," *Sci. Afr.*, vol. 31, Mar. 2026, doi: 10.1016/j.sciaf.2025.e03156.

[17] A. Ali and C. Gravino, "Improving software effort estimation using bio-inspired algorithms to select relevant features: An empirical study," *Sci. Comput. Program.*, vol. 205, May 2021, doi: 10.1016/j.scico.2021.102621.

[18] V. Uc-Cetina, "Recent Advances in Software Effort Estimation using Machine Learning," Mar. 2023, [Online]. Available: http://arxiv.org/abs/2303.03482

[19] O. H. Alhazmi and M. Z. Khan, "Software Effort Prediction Using Ensemble Learning Methods," *Journal of Software Engineering and Applications*, vol. 13, no. 07, pp. 143–160, 2020, doi: 10.4236/jsea.2020.137010.

[20] P. V. Terlapu, K. K. Raju, G. Kiran Kumar, G. Jagadeeswara Rao, K. Kavitha, and S. Samreen, "Improved Software Effort Estimation Through Machine Learning: Challenges, Applications, and Feature Importance Analysis," *IEEE Access*, vol. 12, pp. 138663–138701, 2024, doi: 10.1109/ACCESS.2024.3457771.

.