



Enhanced Software Defect Prediction Using Ensemble Learning by Adaptive Variable Sparrow Search Algorithm

Vanama Nikhith

23VV1F0027

September 3, 2025

Problem Statement

- Software defect prediction plays a crucial role in improving software quality and reducing maintenance costs. However, traditional machine learning models often suffer from limited accuracy due to the imbalanced and noisy nature of software defect datasets.
- Many optimization-based approaches have been introduced to enhance classifier performance, but existing algorithms often face challenges such as slow convergence, premature stagnation in local optima, and poor generalization to diverse datasets.
- There is a need for an advanced and adaptive optimization method that can effectively balance exploration and exploitation, overcome limitations of current techniques, and improve defect prediction accuracy across multiple software repositories.

Introduction

- Prediction of software defects is essential to ensure high quality software and reduce development and maintenance costs.
- Traditional prediction models often struggle due to noisy, imbalanced datasets and the dynamic nature of software projects.
- Evolutionary and swarm intelligence algorithms have been used to optimize classifiers, but they often suffer from premature convergence and poor scalability.
- There is a growing need for adaptive and robust optimization methods that can improve prediction accuracy and efficiency in diverse data sets.

NASA Datasets

- Projects: PC1, CM1, MC2, MW1
- Characteristic number: 37
- Samples range: 125 – 735
- Defect rates: 0.0830 – 0.3520

AEEEM Datasets

- Projects: JDT, PDE, ML
- Characteristic number: 61
- Samples range: 997 – 1862
- Defect rates: 0.1321 – 0.2066

MORPH Datasets

- Projects: ant-1.3, arc, camel-1.0
- Characteristic number: 20
- Samples range: 125 – 339
- Defect rates: 0.0383 – 0.1600

JIRA Datasets

- Projects: activemq-5.0.0, hbase-0.94.0, hive-0.9.0, groovy-1.6_BETA_1, jruby-1.1
- Characteristic number: 65
- Samples range: 731 – 1884
- Defect rates: 0.0852 – 0.2058

Algorithms Used

- Extreme Learning Model (ELM): A fast learning algorithm for single-hidden layer feedforward neural networks that randomly assigns input weights and biases.
- Sparrow Search Algorithm (SSA): A swarm intelligence optimization algorithm inspired by the foraging and anti-predation behavior of sparrows.
- Adaptive Variable Sparrow Search Algorithm (AVSSA): An improved SSA that adaptively adjusts search strategies to balance exploration and exploitation.
- Sparrow Eagle Based Algorithm (SEB): A hybrid approach combining sparrow search dynamics with eagle strategy to enhance global optimization.
- Adaptive Variable Sparrow Eagle Based Algorithm (AVSEB): The proposed algorithm that adaptively tunes parameters of SEB for better convergence and prediction accuracy.

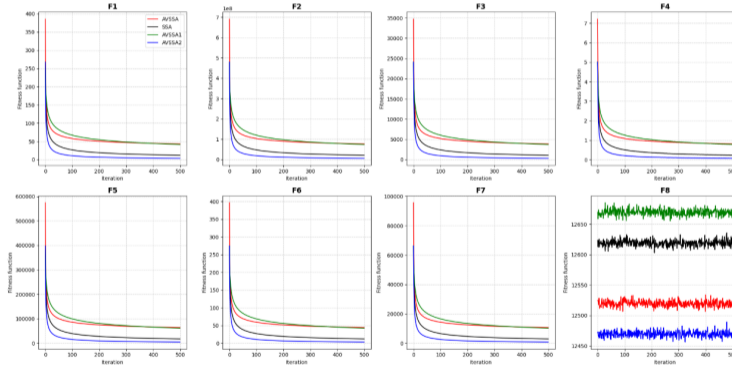
Benchmark Functions

- Standard mathematical benchmark functions, such as Sphere, Rastrigin, Ackley, Griewank, and Rosenbrock, are widely used to test optimization algorithms.
- They evaluate algorithm performance in terms of convergence speed, stability, and ability to achieve global search capability.

Type	Function	Section	Best
HDSP	$F_1(x) = \sum_{i=1}^n x_i^2$	$[-100, 100]$	0
	$F_2(x) = \sum_{i=1}^n (\sum_{i=1}^n x_i^2)$	$[-100, 100]$	0
	$F_3(x) = \max\{ x_i , 1 \leq i \leq n\}$	$[-100, 100]$	0
	$F_4(x) = \sum_i^d x_i ^{i+1}$	$[-100, 100]$	0
HDMP	$F_5(x) = ix_i^4 + \text{random}[0, 1)$	$[-1.28, 1.28]$	0
	$F_6(x) = 418.9829n - \sum_{i=1}^n \sin(\sqrt{ x_i })$	$[-32, 32]$	-418.9829n
	$F_7(x) = \sin^2(3\pi x_1) + (x_1 - 1)^2[1 + \sin^2(3\pi x_2)] + (x_2 - 1)^2[1 + \sin^2(2\pi x_2)]$	$[-50, 50]$	0
	$F_8(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2)^2 + (2.625 - x_1 + x_1x_2^3)$	$[-50, 50]$	0

Convergence Graphs

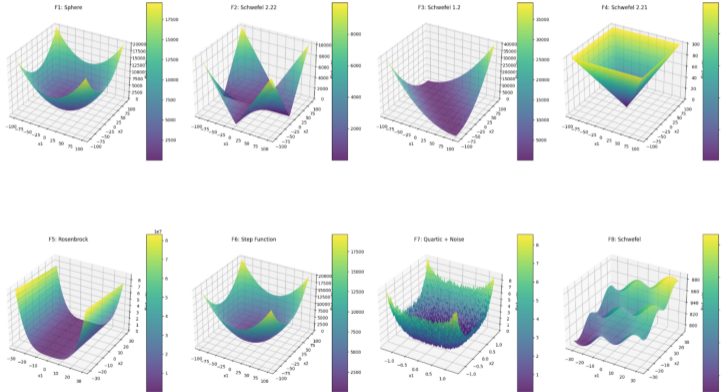
Note: Convergence graphs illustrate how quickly and effectively algorithms approach the optimal solution over iterations.



Convergence behavior of AVSEB vs other algorithms.

3D Heatmaps

Note: 3D heatmaps depict how algorithms search the solution space, highlighting regions of intensification and diversification.



Visualization of solution space exploration and exploitation.

Wilcoxon Rank Test

- A non-parametric statistical test used to compare two paired samples.
- Helps validate whether differences in algorithm performance are statistically significant.
- Applied to compare AVSEB against baseline algorithms across datasets.

Function	AVSSA-SSA	AVSSA-AVSSA1	AVSSA-AVSSA2
F1	6.1206E-08	4.3365E-08	5.7146E-08
F2	2.3108E-08	3.5080E-09	2.3108E-08
F3	9.2105E-08	1.3075E-08	2.5531E-06
F4	3.2830E-08	5.6378E-10	5.8846E-09
F5	8.6072E-08	2.0059E-08	5.3348E-08
F6	3.4648E-04	0.0029	0.0058
F7	1.6547E-09	7.8829E-09	5.8846E-09
F8	3.8113E-10	1.7990E-07	5.7146E-08

Friedman Average Ranking

- Non-parametric test for multiple algorithm comparison across multiple datasets.
- Produces average ranking of algorithms based on performance.
- Used to confirm robustness and generalization ability of AVSEB.

Algorithm	Friedman ranking	APVs
AVSEB	1.2	-
SEB	2.2	0.0833
AVSSA-ELM	3.4	0.001
SSA-ELM	3.2	0.0004
ELM	5.0	0.0

Algorithm	Friedman ranking	APVs
AVSEB	1.1333	-
SEB	2.5333	0.0153
AVSSA-ELM	3.0667	0.0008
SSA-ELM	3.2667	0.0002
ELM	5.0	0.0

Algorithm	Friedman ranking	APVs
AVSEB	1.2	-
SEB	2.2	0.0833
AVSSA-ELM	3.4667	0.0016
SSA-ELM	3.1333	0.0003
ELM	5.0	0.0

Algorithm	Friedman ranking	APVs
AVSEB	1.0333	-
SEB	2.3333	0.0243
AVSSA-ELM	3.3333	0.0002
SSA-ELM	3.3	0.0002
ELM	5.0	0.0

Evaluation Metrics

- Recall: $Recall = \frac{TP}{TP+FN}$

Measures the proportion of actual defects correctly identified by the model.

Dataset	ELM	SSA-ELM	AVSSA-ELM	SEB	AVSEB
PC1	0.1083	0.3095	0.2533	0.3737	0.9537
CM1	0.1333	0.5133	0.5433	0.5183	1.0
MC2	0.4536	0.975	0.9482	0.98	1.0
MW1	0.4593	0.6037	0.5882	0.6296	0.7636
JDT	0.4712	0.5872	0.5669	0.6585	0.7784
PDE	0.1937	0.3688	0.3321	0.3601	0.5534
ML	0.2813	0.493	0.4793	0.5363	0.7181
ant-1.3	0.3021	0.8229	1.0	0.9444	1.0
arc	0.3083	0.645	0.645	0.6537	1.0
camel-1.0	0.3143	0.4857	0.5333	0.6333	0.7333
activemq-5.0.0	0.5152	0.6126	0.6399	0.64	0.7369
hbase-0.94.0	0.4408	0.5609	0.5581	0.5881	0.7872
hive-0.9.0	0.4319	0.6242	0.5846	0.5993	0.8009
groovy-1.6_BETA_1	0.2362	0.5598	0.5416	0.5797	0.8077
ruby-1.1	0.4852	0.6337	0.6429	0.6461	0.8171
mean	0.3423	0.5824	0.5902	0.6159	0.8042
standard deviation	0.129	0.1553	0.1837	0.1772	0.1541

Evaluation Metrics

- G-Mean: $G-Mean = \sqrt{\frac{TP}{TP+FN} \times \frac{TN}{TN+FP}}$
Evaluates the balance between sensitivity (recall) and specificity.

Dataset	ELM	SSA-ELM	AVSSA-ELM	SEB	AVSEB
PCI	0.2281	0.4865	0.4098	0.6028	0.9561
CM1	0.2207	0.6955	0.7147	0.6761	0.9831
MC2	0.4938	0.9653	0.9497	0.9894	1.0
MW1	0.5617	0.7208	0.7026	0.7366	0.9786
JDT	0.6606	0.7529	0.7439	0.8045	0.8763
PDE	0.3843	0.5367	0.5416	0.5972	0.739
ML	0.4969	0.8344	0.8485	0.5744	0.7562
ant-1.3	0.3345	0.8869	0.9825	0.9674	0.9949
arc	0.3943	0.7805	0.7797	0.7444	0.9921
camel-1.0	0.3474	0.5189	0.6014	0.7788	0.97
activemq-5.0.0	0.6976	0.7741	0.7891	0.8065	0.8507
hbase-0.94.0	0.6344	0.7377	0.7313	0.7562	0.8744
hive-0.9.0	0.6391	0.7353	0.7555	0.7676	0.8264
groovy-1_6_BETA_1	0.4144	0.7373	0.7276	0.7565	0.8939
ruby-1.1	0.6781	0.7869	0.7953	0.7943	0.8994
Mean	0.4791	0.73	0.7382	0.7569	0.9061
Standard deviation	0.1561	0.127	0.1394	0.1139	0.0832

Evaluation Metrics

- F-Measure: $F = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$
Represents the harmonic mean of precision and recall.

Dataset	ELM	SSA-ELM	AVSSA-ELM	SEB	AVSEB
PC1	0.3002	0.5334	0.5025	0.5327	0.7981
CM1	0.3814	0.6342	0.6444	0.7182	0.8957
MC2	0.4698	0.9353	0.9435	0.9888	1.0
MW1	0.4619	0.7847	0.7013	0.7875	0.8438
JDT	0.5477	0.6912	0.6896	0.7713	0.8522
PDE	0.2812	0.5071	0.4796	0.5211	0.6945
ML	0.2362	0.4298	0.435	0.4903	0.713
ant-1.3	0.3515	0.8207	0.8916	0.9626	0.9857
arc	0.3865	0.7383	0.7347	0.8008	0.9408
camel-1.0	0.6667	0.8938	0.8142	0.8761	0.6979
activemq-5.0.0	0.5743	0.7191	0.7219	0.7493	0.806
hbase-0.94.0	0.503	0.6735	0.6547	0.7013	0.8317
hive-0.9.0	0.5347	0.6798	0.6996	0.7253	0.7995
groovy-1.6_BETA_1	0.3894	0.6868	0.673	0.7195	0.8796
jruby-1.1	0.5736	0.7448	0.7617	0.7411	0.8835
Mean	0.4439	0.6982	0.6899	0.7391	0.8415
Standard deviation	0.12	0.1322	0.1358	0.1397	0.0916

Evaluation Metrics

- MCC:
$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

Provides a balanced evaluation even on imbalanced datasets.

Dataset	ELM	SSA-ELM	AVSSA-ELM	SEB	AVSEB
PC1	0.1475	0.5801	0.5547	0.5867	0.7929
CM1	0.0891	0.6478	0.6499	0.7296	0.8891
MC2	0.1395	0.9173	0.901	0.9836	1.0
MW1	0.3077	0.7933	0.7093	0.7836	0.8468
JDT	0.4707	0.6519	0.6595	0.7466	0.8257
PDE	0.2659	0.5159	0.5065	0.5553	0.6951
ML	0.2249	0.4551	0.4624	0.529	0.7121
ant-1.3	0.0045	0.8094	0.8908	0.9641	0.9822
arc	0.1265	0.7506	0.7294	0.7997	0.9401
camel-1.0	0.4057	0.9005	0.8323	0.8851	0.7242
activemq-5.0.0	0.5199	0.6952	0.6915	0.7259	0.7816
hbase-0.94.0	0.4066	0.6312	0.6009	0.6682	0.7968
hive-0.9.0	0.4872	0.6516	0.6653	0.702	0.7742
groovy-1_6_BETA_1	0.3524	0.7055	0.684	0.7327	0.8791
jruby-1.1	0.5489	0.7406	0.7593	0.7195	0.877
Mean	0.2998	0.6964	0.6865	0.7408	0.8344
Standard deviation	0.1669	0.1245	0.1229	0.128	0.0907

Results and Analysis

- The proposed AVSEB algorithm was evaluated on multiple benchmark functions (F1–F8) to validate its convergence speed, stability, and global optimization ability.
- Results show that AVSEB consistently outperformed traditional SSA, AVSSA, SEB, and baseline classifiers across most benchmark functions.
- On software defect datasets (NASA, AEEEM, MORPH, and JIRA), AVSEB achieved higher recall, F-measure, G-Mean, and MCC values compared to other algorithms.
- Statistical tests (Wilcoxon rank-sum and Friedman average ranking) confirmed that the performance improvements of AVSEB are statistically significant.
- Convergence graphs indicate that AVSEB avoids premature stagnation and achieves faster convergence toward optimal solutions.
- 3D heatmap visualizations demonstrate that AVSEB balances exploration and exploitation more effectively than competing methods.

Conclusion

- The proposed AVSEB algorithm effectively improves software defect prediction by adaptively balancing exploration and exploitation in optimization.
- Experimental results on benchmark functions and real-world datasets (NASA, AEEEM, MORPH, JIRA) demonstrate superior performance over SSA, AVSSA, SEB, and baseline models.
- Statistical validation using Wilcoxon rank-sum and Friedman tests confirmed the significance and robustness of AVSEB's improvements.
- Convergence and heatmap visualizations highlighted AVSEB's capability to avoid premature convergence and maintain strong search diversity.
- Overall, AVSEB provides a reliable and efficient approach for enhancing defect prediction accuracy and can be extended to other optimization-based software engineering tasks.



The End