*International Journal of Engineering &Technology*, 7 (4) (2018) 7068-7071

**International Journal of Engineering & Technology**

*Website: www.sciencepubco.com/index.php/IJET*
*doi: 10.14419/ijet.v7i4.22264*
***Research paper***

# Cross-project defect prediction using ant colony optimization

**Kiran Kumar B. [1] *, Dr. Jayadev Gyani [2], Dr. Narsimha G. [3]**

*[1] Dept. of IT, KITS, Warangal, India*
*[2] Dept. of CS, CCIS, Majmaah University, Saudi Arabia*
*[3] Dept. of CSE, JNTUH College of Engineering, Sultanpur, India*
**Corresponding author E-mail: kiran_b_kumar@yahoo.com*

## Abstract

Software defect prediction techniques applied on single project are showcasing good results because of availability voluminous data to train the model. But newly developed software projects may not have sufficient amount data to train the model. In cross-project defect prediction model (CPDP), training model is constructed by using defect dataset of one project (which contains sufficient amount of data) and tested on another project (which contains less amount of data). In this paper, we selected similar features from eight open source defect datasets from PROMISE repository and applied meta-heuristic Ant Colony Optimization (ACO) algorithm for Cross-Project defect Prediction.

***Keywords***: *Ant Colony Optimization; Classification; Cross-Project Defect Prediction; Data Mining; Meta-heuristic.*

## 1. Introduction

Software quality is measured by the reliability of the final product. Reliability is inversely proportion to the number of defects in the product. Deploying the poor quality software, leads to more issues in project maintenance. One way to improve the software reliability is by reducing number of defects (bugs) in the software. So, the software testing team plays a vital role in finding the defects and the cost spent on testing phase takes about half of the total project cost.

To reduce this cost, researchers are focusing on using of defect prediction techniques to predict the defects in early stages of the project. Defects can be detected by training the model with existing defect data and the trained model can be used to test the new data.

Quality of the prediction models depends on dataset size used in training the model. A model trained on large volumes of data gives more accurate prediction [1]. In reality, for most of the projects, the training dataset size is less or may not be available. Building the models for prediction is not possible. Hence, engineers are using the datasets available in other projects to build the model and use these models to test the defects in their own projects [2-4]. This method is called as cross-project defect prediction.

The major challenge in cross-project defect prediction is selecting relevant features (metrics) from different projects. To address these challenges, we used TDselector method [7] which considers the similarity between training instance and testing instance and also the number of defects of each training instance. The selected (relevant) features are normalized using z-score normalization. The normalized features are used in training and testing process.

The rest of the paper is organized as follows. Related work is given in Section II. Section III describes process used. Experimentation and results are shown in Section IV and Section V gives conclusion and future scope.

## 2. Related work

In last few years CPDP become thrust area in the research of software engineering. Comparison between cross-project defect prediction and with-in project defect prediction is presented by He et al. [8]. Different techniques of feature selection are used in the comparison. Results shows that with-in project defect prediction gives high precision where as cross-project defect prediction gives better performance in recall and F-measure. Defect prediction across the companies proposed by Turhan et al. [5]. To build predictors, they used defect dataset of other company's project to test the target projects. Ni et al. [6] proposed three ranking strategies named FeSCH to choose relevant features. By considering class-imbalance contexts under CPDP environments, Ryu et al. [9] proposed a multi objective naive Bayes learning technique. This approach achieved better performance over single objective models.

Li et al. [10] proposed hierarchical select-based filter method by comparing some well known data filters to improve CPDP performance. It shows that the choosing of data filter strategy improves the CPDP performance. Zhang et al. [11] experimented on the projects collected from both SourceForge and Google Code and proposed a universal CPDP model. They concluded that CPDP is viable for different projects which have metric sets of heterogeneous type. Nam and Kim [12] introduced metric selection and matching technique to build a predictor and proposed a method HDP. They experimented on twenty eight different projects and results shows that 68% of predictions are giving better performance compared to WPDP. A Unified Metric Representation (UMR) is proposed by Jing et al. [24] for heterogeneous defect data. They considered fourteen publicly available heterogeneous datasets from 4 different companies for the experimentation. Christian Blum [13] discussed outline and applications of Ant Colony Optimization (ACO) by using more techniques from operations research and artificial intelligence. Ramakanta Mohanthy, Venkatshwarlu Naik, Azmath Mubeen [14] used ACO for prediction of software reliability and to optimize the accuracy of software reliability predictive models. D Martens [15] used AntMiner+ to identify credit risk of customers by building an internal rating system. Bo Liu, Hussein A. Abbass, and Bob McKay [16] proposed an improvement for ACO called Ant_Miner3 to discover the classification rules. Wei Gao [17] presented ant colony algorithm for clustering for improving the accuracy and computational efficiency. Christian Blum [18] presented a review and recent trends of using ACO. M. Dorigo et al [19] introduced A novel Ant Colony based classifier, PolyACO that utilizes ray casting to operate in two dimensional space. David Martens et. al.[20] proposed ant miner+ for classification by using ACO.

## 3. Process

In CPDP, the main challenge is selection of relavant attributes from different defect datasets of various projects. Defect datasets contain different metrics pertaining to each project. For the combination of these metrics, there is a class label attrinute which shows the presence of defect. Some datasets store YES or NO, some may store TRUE or FALSE and some may have 0 or 1. The metric values of each project depends on complexity of the project. For example LOC metric, number of inheritance level etc. is different from one project to another. To address these issues the datasets should be preprocessed.

### 3.1. Pre-processing

The steps involved in preprocessing are given below.

1) Choose the predictor attributes (metrics) which are common in all the datasets.
2) Change the value of class label attribute to uniform value.
3) Normalize the predictor attribute values to map in the range between 0 and 1.

### 3.2. Ant colony optimization

ACO is a meta-heuristic bio-inspired optimization technique motivated by the behavior of real ants in the process of finding their food. The ants can communicate through their environment by depositing a chemical called pheromone. The paths chosen by ants contain high volume of pheromone and the paths that are not chosen having less pheromone level due to evaporation.
ACO uses artificial agents (ants) that cooperate to find good solutions for discrete optimization problems. These artificial agents simulate the foraging behavior of their counterparts in finding the shortest-path to the food source from their nest. The first algorithm following the principles of the ACO meta-heuristic is the Ant System [21], [22], where ants repeatedly construct solutions and add pheromone to the paths corresponding to these solutions.
Path selection is a random procedure based on two parameters, the pheromone and heuristic values. The pheromone value gives an indication of the number of ants that chose the trail recently, while the heuristic value is a problem dependent quality measure.
An ant will move from node i to node j with probability

$$P_{i,j} = \frac{(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}{\sum (\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}$$

Where

$\tau_{i,j}$ Is the amount of pheromone on edge i, j

$\alpha$ is a parameter to control the influence of $\tau_{i,j}^{\alpha}$
$\eta_{i,j}$ is the desirability of edge i, j (typically $1/d_{i,j}$ )
$\beta$ is a parameter to control the influence of $\eta_{i,j}$
Amount of pheromone is updated according to the equation

$$\tau_{i,j} = (1 - \rho)\tau_{i,j} + \Delta\tau_{i,j}$$

Where
$\tau_{i,j}$ is the amount of pheromone on a given edge i, j
$\rho$ is the rate of pheromone evaporation
$\Delta\tau i,j$ is the amount of pheromone deposited, typically given by

$$\Delta\tau_{i,j}^{k} = \begin{cases} \frac{1}{L_k} & \text{if ant } k \text{ travels on edge } i, j \\ 0 & \text{otherwise} \end{cases}$$

Where $L_k$ is the cost of the kth ant’s tour (typically length).

## 4. Experimentation and results

We considered Jureczko datasets [23] obtained from PROMISE repository [24] shown in Table 1 for experimentation.

**Table 1:** Sample Datasets

| Name of the Dataset | Number of attributes | Number of records |
|---|---|---|
| CM1 | 38 | 369 |
| KC1 | 95 | 145 |
| KC2 | 22 | 522 |
| KC3 | 40 | 194 |
| MC2 | 40 | 125 |
| PC1 | 38 | 705 |
| PC3 | 38 | 1077 |
| PC4 | 38 | 1458 |

From these datasets, we selected nine common attributes as predictor attributes and one class label attribute. The class label attribute in different datasets contain different values like YES or NO, TRUE or FALSE, 0 OR 1 etc. We replaced all YES and TRUE values to 1 and all NO and FALSE values to 0. The metrics chosen are then normalized by using the following min-max normalization equation which maps each metric value in to the range of 0 and 1.

$$Z_i = \frac{X_i - \min(X)}{\max(X) - \min(X)}$$

This normalized dataset is used for training and testing the model. The dataset is tested using the models trained by different datasets. For example, the dataset CM1 is tested with the models trained by KC1, KC2, KC3, MC2, PC1, PC3, and PC4. After testing, we create a confusion or error matrix shown in Table 2 to know the values for True-Positives (TP), False-Positives (FP), True-Negatives (TN) and False-Negatives (FN).

**Table 2:** Confusion Matrix

| | | Real Class | |
|---|---|---|---|
| | | Defective | Non-Defective |
| Predicted Class | Defective | TP | FP |
| | Non-Defective | FN | TN |

These values are used in the calculation of geometric mean, sensitivity, specificity, precision, F-measure and accuracy by using the following formulas.

$$Geometric\ Mean = \sqrt{Sencitivity \times Specificity}$$

$$Sencitivity = \mathrm{Recall} = \frac{TP}{TP + FN}$$

$$Specificity = \frac{TN}{TN + FP}\ ;\ \mathrm{Precision} = \frac{TP}{TP + FP}$$

$$F - Measure = \frac{2 \times \mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$$

$$Accuracy = \frac{(TP + TN)}{(FP + FN + TP + TN)}$$

The results are depicted in table 3 to table 10 shows that cross project data can be used to predict the defects.

**Table 3:**

| Testing:CM1 | | | | | | |
|---|---|---|---|---|---|---|
| Training | Geometric Mean | Sensitivity | Specificity | Precision | Fmeasure | Accuracy |
| KC1 | 0.955533 | 1 | 0.913043 | 0.684211 | 0.8125 | 0.926829 |
| KC2 | 0.955533 | 1 | 0.913043 | 0.684211 | 0.8125 | 0.926829 |
| KC3 | 0.940244 | 1 | 0.884058 | 0.619048 | 0.764706 | 0.902439 |
| MC2 | 0.932505 | 1 | 0.869565 | 0.590909 | 0.742857 | 0.890244 |
| PC1 | 0.963087 | 1 | 0.927536 | 0.722222 | 0.83871 | 0.939024 |
| PC3 | 0.932505 | 1 | 0.869565 | 0.590909 | 0.742857 | 0.890244 |
| PC4 | 0.947919 | 1 | 0.898551 | 0.65 | 0.787879 | 0.914634 |

**Table 4:**

| Testing:KC1 | | | | | | |
|---|---|---|---|---|---|---|
| Training | Geometric Mean | Sensitivity | Specificity | Precision | Fmeasure | Accuracy |
| CM1 | 0.946864 | 1 | 0.896552 | 0.727273 | 0.842105 | 0.918919 |
| KC2 | 0.964901 | 1 | 0.931034 | 0.8 | 0.888889 | 0.945946 |
| KC3 | 0.946864 | 1 | 0.896552 | 0.727273 | 0.842105 | 0.918919 |
| MC2 | 0.946864 | 1 | 0.896552 | 0.727273 | 0.842105 | 0.918919 |
| PC1 | 0.909718 | 1 | 0.827586 | 0.615385 | 0.761905 | 0.864865 |
| PC3 | 0.982607 | 1 | 0.965517 | 0.888889 | 0.941176 | 0.972973 |
| PC4 | 0.946864 | 1 | 0.896552 | 0.727273 | 0.842105 | 0.918919 |

**Table 5:**

| Testing:KC2 | | | | | | |
|---|---|---|---|---|---|---|
| Training | Geometric Mean | Sensitivity | Specificity | Precision | Fmeasure | Accuracy |
| CM1 | 0.966092 | 1 | 0.933333 | 0.980583 | 0.990196 | 0.984733 |
| KC1 | 0.966092 | 1 | 0.933333 | 0.980583 | 0.990196 | 0.984733 |
| KC3 | 0.926329 | 0.990099 | 0.866667 | 0.961538 | 0.97561 | 0.961832 |
| MC2 | 0.930949 | 1 | 0.866667 | 0.961905 | 0.980583 | 0.969466 |
| PC1 | 0.912871 | 1 | 0.833333 | 0.95283 | 0.975845 | 0.961832 |
| PC3 | 0.926329 | 0.990099 | 0.866667 | 0.961538 | 0.97561 | 0.961832 |
| PC4 | 0.930949 | 1 | 0.866667 | 0.961905 | 0.980583 | 0.969466 |

**Table 6:**

| Testing:KC3 | | | | | | |
|---|---|---|---|---|---|---|
| Training | Geometric Mean | Sensitivity | Specificity | Precision | Fmeasure | Accuracy |
| CM1 | 0.952353 | 1 | 0.906977 | 0.6 | 0.75 | 0.918367 |
| KC1 | 0.869376 | 0.833333 | 0.906977 | 0.555556 | 0.666667 | 0.897959 |
| KC2 | 0.880451 | 0.833333 | 0.930233 | 0.625 | 0.714286 | 0.918367 |
| MC2 | 0.964486 | 1 | 0.930233 | 0.666667 | 0.8 | 0.938776 |
| PC1 | 0.952353 | 1 | 0.906977 | 0.6 | 0.75 | 0.918367 |
| PC3 | 0.964486 | 1 | 0.930233 | 0.666667 | 0.8 | 0.938776 |
| PC4 | 0.952353 | 1 | 0.906977 | 0.6 | 0.75 | 0.918367 |

**Table 7:**

| Testing:MC2 | | | | | | |
|---|---|---|---|---|---|---|
| Training | Geometric Mean | Sensitivity | Specificity | Precision | Fmeasure | Accuracy |
| CM1 | 0.930484 | 0.909091 | 0.952381 | 0.909091 | 0.909091 | 0.9375 |
| KC1 | 0.9759 | 1 | 0.952381 | 0.916667 | 0.956522 | 0.96875 |
| KC2 | 0.95119 | 1 | 0.904762 | 0.846154 | 0.916667 | 0.9375 |
| KC3 | 0.95119 | 1 | 0.904762 | 0.846154 | 0.916667 | 0.9375 |
| PC1 | 0.930484 | 0.909091 | 0.952381 | 0.909091 | 0.909091 | 0.9375 |
| PC3 | 0.9759 | 1 | 0.952381 | 0.916667 | 0.956522 | 0.96875 |
| PC4 | 0.906924 | 0.909091 | 0.904762 | 0.833333 | 0.869565 | 0.90625 |

**Table 8:**

| Testing:PC1 | | | | | | |
|---|---|---|---|---|---|---|
| Training | Geometric Mean | Sensitivity | Specificity | Precision | Fmeasure | Accuracy |
| CM1 | 0.923404 | 0.933333 | 0.91358 | 0.5 | 0.651163 | 0.915254 |
| KC1 | 0.907672 | 0.933333 | 0.882716 | 0.424242 | 0.583333 | 0.887006 |
| KC2 | 0.942809 | 1 | 0.888889 | 0.454545 | 0.625 | 0.898305 |
| KC3 | 0.942809 | 1 | 0.888889 | 0.454545 | 0.625 | 0.898305 |
| MC2 | 0.894887 | 0.933333 | 0.858025 | 0.378378 | 0.538462 | 0.864407 |
| PC3 | 0.904493 | 0.933333 | 0.876543 | 0.411765 | 0.571429 | 0.881356 |
| PC4 | 0.949334 | 1 | 0.901235 | 0.483871 | 0.652174 | 0.909605 |

**Table 9:**

| Testing:PC3 | | | | | | |
|---|---|---|---|---|---|---|
| Training | Geometric Mean | Sensitivity | Specificity | Precision | Fmeasure | Accuracy |
| CM1 | 0.921266 | 0.964286 | 0.880165 | 0.482143 | 0.642857 | 0.888889 |
| KC1 | 0.940371 | 1 | 0.884298 | 0.5 | 0.666667 | 0.896296 |
| KC2 | 0.935966 | 1 | 0.876033 | 0.482759 | 0.651163 | 0.888889 |
| KC3 | 0.951293 | 1 | 0.904959 | 0.54902 | 0.708861 | 0.914815 |
| MC2 | 0.89271 | 0.892857 | 0.892562 | 0.490196 | 0.632911 | 0.892593 |
| PC1 | 0.944755 | 1 | 0.892562 | 0.518519 | 0.682927 | 0.903704 |
| PC4 | 0.964237 | 1 | 0.929752 | 0.622222 | 0.767123 | 0.937037 |

**Table 10:**

| Testing:PC4 | | | | | | |
|---|---|---|---|---|---|---|
| Training | Geometric Mean | Sensitivity | Specificity | Precision | Fmeasure | Accuracy |
| CM1 | 0.953162 | 1 | 0.908517 | 0.623377 | 0.768 | 0.920548 |
| KC1 | 0.923322 | 0.958333 | 0.88959 | 0.567901 | 0.713178 | 0.89863 |
| KC2 | 0.9399 | 0.979167 | 0.902208 | 0.602564 | 0.746032 | 0.912329 |
| KC3 | 0.921684 | 0.958333 | 0.886435 | 0.560976 | 0.707692 | 0.89589 |
| MC2 | 0.943181 | 1 | 0.88959 | 0.578313 | 0.732824 | 0.90411 |
| PC1 | 0.926661 | 0.979167 | 0.876972 | 0.546512 | 0.701493 | 0.890411 |
| PC3 | 0.931401 | 1 | 0.867508 | 0.533333 | 0.695652 | 0.884932 |

## 5. Conclusions and future work

In this paper, we tested the each dataset against the models trained using different datasets. We applied novel bio-inspired meta-heuristic Ant Colony Optimization (ACO) technique to train and test the models. The results show that cross project defect prediction with ACO giving improved performance. In future, we implement various meta-heuristic algorithms in defect prediction.

## References

[1] N. Nagappan, T. Ball, and A. Zeller, "Mining metrics to predict component failures," in International Conference on Software Engineering, 2006, pp. 452-461. https://doi.org/10.1145/1134285.1134349.

[2] B. Kitchenham, E. Mendes, and G. H. Travassos, "Cross- vs. within company cost estimation studies: A systematic review," IEEE Transactions in Software Engineering, vol. 33, pp. 316-329, 2007. https://doi.org/10.1109/TSE.2007.1001.

[3] F. Porto and A. Simao, "Feature Subset Selection and Instance Filtering for Cross-project Defect Prediction - Classification and Ranking", CLEI electronic journal, vol. 19, no. 3, pp. 4:1-4:17, 2016. https://doi.org/10.19153/cleiej.19.3.4.

[4] X. Yang, D. Lo, X. Xia and J. Sun, "TLEL: A two-layer ensemble learning approach for just-in-time defect prediction", Information and Software Technology, vol. 87, pp. 206-220, 2017. https://doi.org/10.1016/j.infsof.2017.03.007.

[5] B. Turhan, T. Menzies, A. B. Bener, and J. Di Stefano, "On the relative value of cross-company and within-company data for defect prediction," Empirical Software Engineering, vol. 14, no. 5, pp.540–578, 2009. https://doi.org/10.1007/s10664-008-9103-7.

[6] C. Ni, W. Liu, Q. Gu, X. Chen, and D. Chen, "FeSCH: A Feature Selection Method using Clusters of Hybrid-data for Cross-Project Defect Prediction," in Proceedings of the 41st IEEE Annual Computer Software and Applications Conference, COMPSAC2017, pp.51–56, ita, July2017. https://doi.org/10.1109/COMPSAC.2017.127.

[7] Peng He, Yao He, Lvjun Yu, and Bing Li, "An Improved Method for Cross-Project Defect Prediction by Simplifying Training Data," Mathematical Problems in Engineering, vol. 2018, Article ID 2650415, 18 pages, 2018. https://doi.org/10.1155/2018/2650415.

[8] P. He, B. Li, X. Liu, J. Chen, and Y. Ma, "An empirical study on software defect prediction with a simplified metric set," Information and Software Technology, vol.59, pp.170–190,2015. https://doi.org/10.1016/j.infsof.2014.11.006.

[9] D.Ryu and J.Baik, "Effective multi-objective naive Bayes learning for cross-project defect prediction", Applied Soft Computing, vol.49, pp.1062–1077, 2016. https://doi.org/10.1016/j.asoc.2016.04.009.

[10] Y. Li, Z.Huang, Y. Wang, and B. Fang, "Evaluating Data Filter on Cross-Project Defect Prediction: Comparison and Improvements," IEEEAccess, vol.5, pp.25646–25656, 2017. https://doi.org/10.1109/ACCESS.2017.2771460.

[11] F. Zhang, A. Mockus, I. Keivanloo, and Y. Zou, "Towards building a universal defect prediction model," in Proceedings of the 11th International Working Conference on Mining Software Repositories, MSR2014, pp.182–191, ind, June2014. https://doi.org/10.1145/2597073.2597078.

[12] J. Nam and S. Kim, "Heterogeneous defect prediction," in Proceedings of the 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2015, pp. 508–519, September 2015. https://doi.org/10.1145/2786805.2786814.

[13] Christian Blum, "Ant colony optimization: Introduction and recent trends", Elsevier, Physics of Life Reviews 2 pp. 353–373, 2005 https://doi.org/10.1016/j.plrev.2005.10.001.

[14] Ramakanta Mohanthy, Venkateshwarlu Naik, Azmath Mubeen, "Predicting Software Reliability Using Ant Colony Optimization Technique" 2014 Fourth International Conference on Communication Systems and Network Technologies, pp. 496-500, 2014. https://doi.org/10.1109/CSNT.2014.105.

[15] D Martens, T Van Gestel, M De Backer, R Haesen, J Vanthienen and B Baesens, "Credit rating prediction using Ant Colony Optimization" Journal of the Operational Research Society (2010) 61, pp. 561-573, 2010. https://doi.org/10.1057/jors.2008.164.

[16] Bo Liu, Hussein A. Abbass, and Bob McKay, "Classification Rule Discovery with Ant Colony Optimization", IEEE Computational Intelligence Bulletin, Vol.3, No.1, pp. 31-35, February 2004.

[17] Wei Gao, "Improved Ant Colony Clustering Algorithm and Its Performance Study", Computational Intelligence and Neuroscience, Volume 2016, Article ID 4835932, 14 pages https://doi.org/10.1155/2016/4835932.

[18] Christian Blum, "Ant colony optimization: Introduction and recent trends", Physics of Life Reviews 2 (2005) 353–373. https://doi.org/10.1016/j.plrev.2005.10.001.

[19] M. Dorigo et al., "Ant Colony Optimisation-Based Classification Using Two-Dimensional Polygons", Springer International Publishing Switzerland, pp. 53–64, 2016. https://doi.org/10.1007/978-3-319-44427-7_5.

[20] David Martens et. al., "Classification with Ant Colony Optimization", IEEE transactions on evolutionary computation, vol. 11, no. 5, pp. 651-664, October 2007. https://doi.org/10.1109/TEVC.2006.890229.

[21] M. Dorigo, V. Maniezzo, and A. Colorni, "Positive feedback as a search strategy" Dipartimento di Elettronica e Informatica, Politecnico di Milano, Milano, Italy, Tech. Rep. 91016, 1991.

[22] M. Dorigo ; V. Maniezzo ; A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern. Part B*, vol. 26, no. 1, pp. 29–41, Feb.1996. https://doi.org/10.1109/3477.484436.

[23] M. Jureczko and L. Madeyski, "Towards identifying software project clusters with regard to defect prediction", Proceedings of the 6th International Conference on Predictive Models in Software Engineering - PROMISE '10, 2010. https://doi.org/10.1145/1868328.1868342.

[24] T. Menzies, B. Cagayan, Z. He, E. Kocaguneli, J. Krall, F. Peters, et al., The PROMISE Repository of empirical software engineering data, 2012 http:// openscience.us/repo/.