IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 10 December 2022, accepted 27 January 2023, date of publication 3 February 2023, date of current version 8 February 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3241924*

RESEARCH ARTICLE

# Heterogeneous Cross-Project Defect Prediction via Optimal Transport

**XING ZONG, GUIYU LI, SHANG ZHENG, HAITAO ZOU, HUALONG YU, AND SHANG GAO**
School of Computer, Jiangsu University of Science and Technology, Zhenjiang 212100, China

Corresponding author: Shang Zheng (szheng@just.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 62176107.

**ABSTRACT** Heterogeneous cross-project defect prediction (HCPDP) aims to learn a prediction model from a heterogeneous source project and then apply the model to a target project. Existing HCPDP works mapped the data of the source and target projects in a common space. However, the pre-defined forms of mapping methods often limit prediction performance and it is difficult to measure the distance between two data instances from different feature spaces. This paper introduced optimal transport (OT) theory for the first time to build the relationship between source and target data distributions, and two prediction algorithms were proposed based on OT theory. In particular, an algorithm based on the entropic Gromov-Wasserstein (EGW) discrepancy was developed to perform the HCPDP model. The proposed EGW model measures the distance between two metric spaces by learning an optimal transfer matrix with the minimum data transfer cost and avoids measuring the distance of two instances of different feature spaces. Then, to improve EGW performance, an EGW+ transport algorithm based on EGW was developed by integrating target labels. Experimental results showed the effectiveness of EGW and EGW+ methods, and proved that our methods can support developers to find the defects in the early phase of software development.

**INDEX TERMS** Software engineering, software development, software maintenance, software defect.

## I. INTRODUCTION

Software defect prediction (SDP) [1] is a hot research topic in software engineering and is considered a binary classification problem that can classify software modules into defective or non-defective. An effective SDP method can help software developers improve software quality by predicting potential defects in advance. Traditional defect prediction methods aim to build a prediction model based on some historical data in a project, and make predictions on the remaining data, called within-project defect prediction (WPDP). Researchers have proposed a number of methods to solve the WPDP problem [2], [3], [4] and these methods require that the historical data should be sufficient to construct the prediction model. However, in practice, WPDP may not work well for a new project because a new project has little or no labeled data to train a classification model.

An alternative solution is Cross-Project Defect Prediction (CPDP) [5], [6], [7], [8] based on transfer learning, which trains a prediction model by using the data collected from other projects. Unlike the WPDP methods, the training and test sets come from different projects. CPDP is called homogeneous cross-project defect prediction when the source and target projects have the same metrics. However, practitioners may use different metrics to measure the modules. Hence, heterogeneous cross-project defect prediction (HCPDP) is proposed [9] to solve this problem. Compared to CPDP with homogeneous metrics, HCPDP is more difficult to adjust the domain difference due to the different metrics. To solve the HCPDP problem, researchers have drawn lessons from transfer learning techniques [10], [11], [12], [13], [14], [15] to eliminate the heterogeneity between source and target projects.

Transfer learning used training data from the source projects to improve the prediction performance in the target projects [16]. When the feature space of the source and target projects are different, the transfer learning is called

The associate editor coordinating the review of this manuscript and approving it for publication was Aasia Khanum.

heterogeneous transfer learning, which can be used in the HCPDP works. The researchers considered transferring the data from the source and target projects to a common space using a learning mapping function. However, these functions are often pre-defined in a hypothesis space, which can limit feature transformation between two different distributions. Specially, it is very difficult to directly measure the distance between data instances with different feature representations. These limitations can make the HCPDP model unstable.

Instead of distance measurement, optimal transport (OT) theory [17] can transport samples from one distribution to another by examining the minimum cost of transmission. Recently, some studies applying OT for domain adaptation [18], [19] have been reported. To the best of our knowledge, OT theory has not been applied to the HCPDP task.

Inspired by OT theory, the entropic Gromov-Wasserstein (EGW) discrepancy [20] is introduced for the first time to perform heterogeneous transfer learning task by studying two different distributions. It is a powerful tool in OT theory for learning an optimal transport matrix that transfers data from one metric space to another metric space. For the HCPDP problem, the distributions of the target and source projects are not the same because they have different metrics. Therefore, HCPDP is one of heterogeneous tasks, and EGW can be considered solving the HCPDP problem. To maintain label consistency, we also proposed EGW+ by combining target labels during transportation so that the same label could follow a similar distribution in the target and source samples. The main contributions of this study can be summarized as follows:

- We for the first time introduced optimal transport into HCPDP task.
- An EGW-based algorithm was designed to perform the HCPDP task. It transferred the instances of the source project to the target project by learning an optimal transfer matrix.
- The labels of the target data were integrated into EGW-based algorithm, and EGW+ was proposed. It can ensure the optimal transmission scheme meets the label consistency.

To evaluate the effectiveness of the proposed approach, we conducted the experiments on 20 public datasets from AEEEM, JIRA, NASA and PROMISE projects. Compared to the state-of-the-art methods, the results indicated that the predictive models generated by two proposed algorithms could achieve a desirable performance for the HCPDP task.

The rest of this paper is organized as follows: Section II describes the related work. Section III presents the proposed approach. Section IV illustrates the experimental setup, and the results are given in Section V. Section VI provides discussions about the proposed approach and the main threats of this study. Section VII concludes this study.

## II. RELATED WORK

As HCPDP is the subject of this study, the other categories of CPDP can be found in the survey [5]. To solve the HCPDP problem, researchers designed the prediction models based on transfer learning method. For example, Nam and Kim [9] proposed the HDP_KS method that uses feature selection and feature mapping to perform domain adaption. In the work [21], transfer component analysis was proposed. In addition, the authors extended their work by using the normalization techniques to preprocess data. Liu et al. [22] proposed a two-phase transfer learning method to improve transfer component analysis. In the first phase, they designed a source project estimator to select two similar projects. In the second phase, two prediction models were built based on the two selected projects, and their prediction results were combined to improve performance. Li et al. [13] proposed a set of novel HCPDP methods based on kernel correlation alignment and ensemble learning to solve the linearly problem. Li et al. [23] proposed a cost-sensitive label and structure-consistent unilateral project (CLSUP) approach for HCPDP, which applied domain adaption by combining a limited set of target data and numerous source data. Jing et al. [24] proposed a metric representation for source and target project data, and then introduced Canonical Correlation Analysis (CCA) in HCPDP. The results proved that their CCA+ approach can achieve better performance. He et al. [25] proposed a method for distribution characteristic that analyzed 16 indicators. Cheng et al. [26] proposed a method CCT-SVM that takes into account different misclassification costs when constructing the models via SVM to solve the imbalance problem.

Recently, some researchers have investigated that the metric of defective modules in the collected SDP datasets is often higher than the metric value of non-defective modules. Zhou et al. [27] concluded the supervised and unsupervised methods for CPDP and found that different methods can achieve different performance on the same datasets. Zhang et al. [28] studied two types of unsupervised classifiers:1) distance-based classifier; and 2) connectivity-based classifier, and the results proved that the connectivity-based classifier provides a better solution for CPDP. Nam and Kim [10] proposed CLA and CLAMI to perform prediction on unlabeled datasets. The key idea of their approach was to label unlabeled datasets using the size of metric values. Chen et al. [29] compared different methods for the HCPDP task. The experimental results also confirmed that the performance of HCPDP model can depend on the defective datasets. These results may be useful to improve CPDP performance in the future. Jiang et al. [30] applied the Mahalanobis distance to deal with data imbalance, but they required that the inverse of the sample of covariance matrix must be present.

The difference between our approach and the current methods is that we for the first time introduced optimal transport theory into the field of HCPDP, and proposed two strategies based on the Gromov-Wasserstein entropic discrepancy, which can transport the source data to the target project with minimal transport cost. By learning the transportation, a classifier can be trained using the transported samples and target samples to predict the defects.

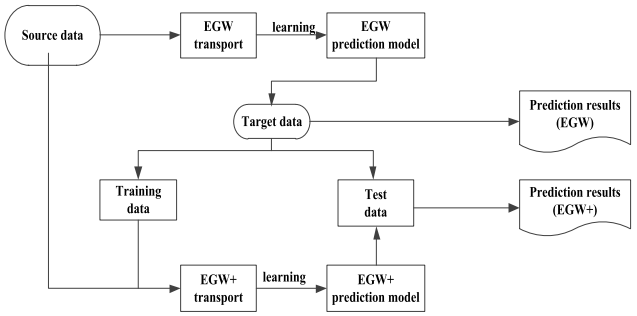



**FIGURE 1.** The framework of the proposed approach.

## III. PROPOSED APPROACH

This section presents the framework of the proposed approach including the EGW and EGW+ algorithms, and Fig. 1 demonstrates the framework.

As can be seen in Fig. 1, the source data is first transported by EGW discrepancy calculation, and the transport matrix is obtained. After learning the transportation, an EGW model is trained using the transported source data and applied to the target data for prediction. Then, when some labeled target data is known, we introduced the labels into the transportation, which can help label matching between transported source and target data. By learning the new transportation, an EGW+ model is trained using the labeled target and transported source data. In order to describe the approach in detail, the notations, the entropic Gromov-Wasserstein discrepancy, the EGW and EGW+ algorithms are presented below.

### A. NOTATIONS

For convenience of presentation, we denote the source data and the target data as $D_s = [d_1^s, \ldots, d_{n_s}^s]^{\mathrm{T}} \in \mathbb{R}^{n_s \times dm_s}$, $D_t = [d_1^t, \ldots, d_{n_t}^t]^{\mathrm{T}} \in \mathbb{R}^{n_t \times dm_t}$, where $n$ is the number of data, $d_m$ is the dimension of the metric. The simplex of histograms with N bins is $\Sigma_N \overset{def}{=} \{p \in \mathbb{R}_N^+;\ \Sigma_i p_i = 1\}$.

Finally, the entropy of transport matrix $\mathbf{T}$ is defined by $\mathrm{H}(\mathbf{T}) = -\sum_{i=1}^{n_s}\sum_{j=1}^{n_t} T_{ij}(\log T_{ij} - 1)$, the set of couplings between histogram $p \in \Sigma_{N1}$ and $q \in \Sigma_{N2}$ is $C_{p,q} \overset{def}{=} \{T \in (\mathbb{R}_+)^{N1 \times N2};\ T\mathrm{I}_{N_2} = p, T^{\top}\mathrm{I}_{N_1} = q\}$, $I_N \overset{def}{=} (1, \ldots, 1)^{\top} \in \mathbb{R}^N$. For any tensor $L = (L_{i,j,k,l})_{i,j,k,l}$ and matrix $\mathrm{T} = (T_{i,j})_{i,j}$, the sensor matrix multiplication can be defined as $L \otimes T \overset{def}{=} (\sum_{k,l} L_{i,j,k,l} T_{k,l})_{i,j}$.

### B. ENTROPIC GROMOV-WASSERSTEIN DISCREPANCY

Since the metric representations in two projects are different in HCPDP, it is necessary to build the correspondence between heterogeneous metric spaces for domain adaptation. EGW is based on the optimal transport theory that seeks for a best solution to transport a distribution to another distribution. It does not calculate the distance, but directly measures the cost of data transmission on the metric matrix of the source and target data.

Given two distributions from the source and target projects, they can be represented by two histograms $p_s$ and $q_t$. Then $\mathbf{T}$, a joint distribution of $p_s$ and $q_t$, is defined by $\mathbf{T} \in \tau = \mathbf{T} \in \mathbb{R}^{n_s \times n_t} | \mathbf{T}\mathbf{1}_{n_t} = p_s, \mathbf{T}^{\top}\mathbf{1}_{n_s} = q_t\}$. Eq. (1) show the EGW formula.

$$EGW(\mathbf{C}_s, \mathbf{C}_t, p_s, q_t) = \min_{\mathbf{T} \in \mathbf{C}_{\mathbf{p}_s, \mathbf{q}_t}} \mathrm{E}_{\mathbf{C}_s, \mathbf{C}_t}(\mathbf{T}) - \varepsilon H(\mathbf{T})$$
$$\mathrm{E}_{\mathbf{C}_s, \mathbf{C}_t}(\mathbf{T}) = \sum_{i,j,k,l} L(C_{i,k}^s, C_{j,l}^t) T_{i,j} T_{k,l}. \quad (1)$$

In Eq. (1), $C_s$ is a matrix representing a certain metric on source samples, $C_t$ is a matrix that represents metrics on target samples, and they can be constructed by the linear kernel matrix, $C_s = D_s D_s^{\top}$, $C_t = D_t D_t^{\top}$; $C_{i,k}^s$ is the $(i, k)$-th element of $C_s$, $C_{j,l}^t$ is the $(j, l)$-th element of $C_t$; $\varepsilon$ is the entropic regularization term; $L(C_{i,k}^s, C_{j,l}^t)$ is the loss function that measured the transport cost between $C_{i,k}^s$ and $C_{j,l}^t$, and is defined by $L(C_{i,k}^s, C_{j,l}^t) = (C_{i,k}^s - C_{j,l}^t)^2$.

It can be investigated that Eq. (1) is a non-convex optimization problem. By following the solution [20], the projected gradient descent is applied to solve it, where the gradient step and the project are based on Kullback-Leibler (KL) divergence [31]. Then, $\mathbf{T}$ is firstly updated by

$$\mathbf{T} \leftarrow \mathrm{Proj}_{\mathbf{C}_{\mathbf{p}_s, \mathbf{q}_t}}^{KL}(\mathbf{T} \odot e^{-\tau(\nabla \mathrm{E}_{\mathbf{C}_s, \mathbf{C}_t}(\mathbf{T}) - \varepsilon \nabla H(\mathbf{T}))}) \quad (2)$$

where $\tau > 0$ is a small enough step size, and the KL projector of the updated $\mathbf{T}$ is

$$\mathrm{Proj}_{\mathbf{C}_{\mathbf{p}_s, \mathbf{q}_t}}^{KL}(\mathbf{T}) \overset{def}{=} \underset{\mathbf{T}' \in \mathbf{C}_{\mathbf{p}_s, \mathbf{q}_t}}{\arg\min}\ KL(\mathbf{T}' \big| \mathbf{T}) \quad (3)$$

When the special case $\tau = 1/\varepsilon$, iteration Eq. (3) reads

$$\mathbf{T} \leftarrow \tau(L(\mathbf{C}_s, \mathbf{C}_t) \otimes \mathbf{T}, \mathbf{p}_s, \mathbf{q}_t) \quad (4)$$

where $\otimes$ denotes the tensor-matrix multiplication.

As described by [32], the projection is the solution to the regularized transport problem, hence

$$\mathrm{Proj}_{\mathbf{C}_{\mathbf{p}_s, \mathbf{q}_t}}^{KL}(\mathbf{T}) = \tau(-\varepsilon \log(\mathbf{T}), \mathbf{p}_s, \mathbf{q}_t) \quad (5)$$

Re-arrange the terms in Eq. (3), when $\tau = 1/\varepsilon$, the desired formula of Eq. (1) is given as follows:

$$\nabla \mathrm{E}_{\mathbf{C}_s, \mathbf{C}_t}(\mathbf{T}) - \varepsilon \nabla H(\mathbf{T}) = L(\mathbf{C}_s, \mathbf{C}_t) \otimes \mathbf{T} + \varepsilon \log(\mathbf{T}) \quad (6)$$

The above process involved the Iteration Eq. (4), in which each update $\mathbf{T}$ applies a Sinkhorn projection [33]. To obtain the corresponding convergence proof, more details can be found in some work by Peyre et al. [20] and Yan et al. [34].

After the optimal transmission matrix $\mathbf{T}$ is calculated, we can transport the source data to the target data, thereby representing the source data in the metric space of the target data. The transported source data $D_{transported}$ in the target domain can be computed by the GW barycenter [20], which is defined as

$$D_{transpoted} = n_s \mathbf{T} D_{target} \quad (7)$$

As analyzed in Eq. (7), the transported source data have the same numbers of metrics to that of target data.

**Algorithm 1** EGW-Based for HCPDP

**Input:** $D_s = [d_1^s, \ldots, d_{n_s}^s]^{\mathrm{T}} \in \mathbb{R}^{n_s \times dm_s}$, the label $Y_s$ of on $D_s$, $D_t = [d_1^t, \ldots, d_{n_t}^t]^{\mathrm{T}} \in \mathbb{R}^{n_t \times dm_t}$
**Output:** Class labels for $D_t$.
1. Initialize $\tau$, $\mathbf{T}$, $\tau = 1$, $\mathbf{T}_1 = p_s q_t^{\top}$.
2. **repeat**
 2.1. Compute $D_{transported}$ by Eq. (7)
 2.2. Iterate $\mathbf{T}$ by Eq. (4).
 2.3. Compute the gradient of the updated $\mathbf{T}$ by Eq. (5).
 2.4. Use Sinkhorn to solve Eq. (6) and update $\mathbf{T}$.
 2.5. $\tau := \tau + 1$.
 2.6. **until** Convergence.
3. Train a classifier on ($D_{transported}$, $Y_s$) with its label for HCPDP.
4. Predict the label for $D_t$.

**Algorithm 2** EGW+-Based for HCPDP

**Input:** $D_s = [d_1^s, \ldots, d_{n_s}^s]^{\mathrm{T}} \in \mathbb{R}^{n_s \times dm_s}$, $y_i^s \in Y_s$; $D_t = [d_1^t, \ldots, d_{n_t}^t]^{\mathrm{T}} \in \mathbb{R}^{n_t \times dm_t}$, $y_i^l \in Y_T$.
**Output:** Class labels for $D_t$.
1. Initialize $\tau$, $\mathbf{T}$, $\tau = 1$, $\mathbf{T}_1 = p_s q_t^{\top}$.
2. **repeat**
 2.1. Compute $D_{transported}$ by Eq. (7)
 2.2. Iterate $\mathbf{T}$ by Eq. (4).
 2.3. Compute the gradient of the updated T by Eq. (5).
 2.4. Use Sinkhorn to solve Eq. (11) and update $\mathbf{T}$.
 2.5. $\tau := \tau + 1$.
 2.6. **until** Convergence.
3. Train a classifier on ($D_{transported}$, $Y_s$) and ($D_t$, $Y_t$) for HCPDP.
4. Predict the labels for $D_t$

### C. PREDICTION-MODEL GENERATED BY EGW

By learning the transportation, a classifier can be trained using the transported source data, and then to predict the defects in the target projects. Algorithm 1 summarizes the main steps of EGW.

### D. PREDICTION-MODEL GENERATED BY EGW+

By analyzing Algorithm 1, it does not consider the target data. If there are a few labels known in the target, it is necessary to match the label condition distribution of two datasets. Therefore, EGW+ is designed to make the samples with the same label more closely distributed after transmission. A regularization term $\varphi(\mathbf{T})$, which is similar to the maximum mean difference based on label conditions using a linear kernel function [35], is added in Eq. (1). Then, Eq. (1) can be changed as follows:

$$EGW(\mathbf{C}_s, \mathbf{C}_t, p_s, q_t) = \min_{\mathbf{T} \in \mathbf{C}_{\mathbf{p_s}, \mathbf{q_t}}} \mathrm{E}_{\mathbf{C}_s, \mathbf{C}_t}(\mathbf{T}) - \varepsilon H(\mathbf{T}) + \lambda \varphi(\mathbf{T})$$

where

$$\mathrm{E}_{\mathbf{C}_s, \mathbf{C}_t}(\mathbf{T}) = \sum_{i,j,k,l} L(C_{i,k}^s, C_{j,l}^t) T_{i,j} T_{k,l}, \tag{8}$$

$\varphi(\mathbf{T})$ is the labeled information, and it is defined as follows:

$$\begin{aligned} \varphi(T) &= \sum_{k=1}^{K} \left\| \frac{1}{n_k^s} \sum_{i=1}^{n_k^s} d_{ik}^{transported} - \frac{1}{n_k^l} \sum_{i=1}^{n_k^l} d_{ik}^l \right\|_2^2 \\ &= \| n_s \mathrm{P} \mathrm{T} \mathrm{D}_{\mathrm{t}} - \mathrm{Q} \mathrm{D}_{\mathrm{t}} \|_F^2 \end{aligned} \tag{9}$$

where $n_k^s$ and $n_k^l$ were the number of labeled source and target data, $d_{ik}^{transported}$ and $d_{ik}^l$ are the data with label $k$. The label indicator matrices were $P \in \mathbb{R}^{K \times n_s}$, $Q \in \mathbb{R}^{K \times n_l}$, which are defined as follows:

$$P_{k,i} = \begin{cases} 1/n_k^s & if\ y_i^s = k, \\ 0 & otherwise; \end{cases}$$

$$Q_{k,i} = \begin{cases} 1/n_k^l & if\ y_i^l = k, \\ 0 & otherwise; \end{cases} \tag{10}$$

where the value of $k$ is {0,1}, $y$ represents the label. In HCPDP datasets, there are two labels including defective or non-defective. 0 represents non-defective and 1 represents defective.

Finally, Eq. (8) is re-arranged as follows:

$$\begin{aligned} &\nabla \mathrm{E}_{C_s, C_t}(\mathbf{T}) - \varepsilon \nabla H(\mathbf{T}) + \lambda \nabla \varphi(\mathbf{T}) \\ &= L(C_s, C_t) \otimes \mathbf{T} + \varepsilon \log(\mathbf{T}) + 2\lambda n_s P^T (n_s P T - Q) D D^T \end{aligned} \tag{11}$$

In conclusion, Algorithm 2 is given to illustrate the steps of the EGW+-based.

## IV. EXPERIMETNAL SETUP

### A. RESEARCH QUESTIONS

To investigate the effectiveness of the proposed algorithms, we designed the following two research questions.

*RQ1: How well EGW deals with HCPDP by using only source projects as training data?*

The proposed EGW algorithm did not consider the target data, and only learn the transportation between source and target distributions. This question is designed to evaluate the effectiveness of EGW without the data of target projects.

*RQ2: How well EGW+ deals with HCPDP by using a few label data of target projects as training data?*

EGW+ algorithm was designed to keep label consistency between the transmitted source data and target data. This question aims to investigate whether EGW+ can outperform the baselines, which also used the target data information.

### B. DATASETS

In the experiment, we used 20 datasets from different projects including AEEEM [36], JIRA [37], NASA [38] and PROMISE [39]. In fact, previous studies used different those methods might perform better on the selected

**TABLE 1. Defect datasets.**

| Project | Dataset | Number of instances | Number of defects | Defect% | Number of metrics |
|---|---|---|---|---|---|
| AEEEM | EQ | 324 | 129 | 39.8 | 61 |
| | JDT | 997 | 206 | 20.7 | 61 |
| | LC | 691 | 64 | 9.3 | 61 |
| | ML | 1862 | 245 | 13.2 | 61 |
| | PDE | 1497 | 209 | 14.0 | 61 |
| JIRA | activemq5.0.0 | 1884 | 293 | 15.6 | 65 |
| | derby10.5.1.1 | 2705 | 383 | 14.2 | 65 |
| | groovy1.6 | 821 | 70 | 8.5 | 65 |
| | hbase0.94.0 | 1059 | 218 | 2.6 | 65 |
| | hive0.9.0 | 1416 | 283 | 20.0 | 65 |
| NASA | KC1 | 2095 | 325 | 15.5 | 21 |
| | MC1 | 8737 | 68 | 0.8 | 38 |
| | PC1 | 735 | 61 | 8.3 | 37 |
| | PC3 | 1099 | 138 | 12.6 | 37 |
| | PC4 | 1379 | 178 | 12.9 | 37 |
| PROMISE | camel1.6 | 965 | 188 | 19.5 | 20 |
| | lucene2.4 | 340 | 203 | 59.7 | 20 |
| | poi3.0 | 442 | 281 | 63.6 | 20 |
| | synapse1.2 | 256 | 86 | 33.6 | 20 |
| | velocity1.6 | 229 | 78 | 34.1 | 20 |

datasets [9], [13], [24], [26], [54] but worse on the other datasets. In order to keep the comparison fair, the datasets used in common in all previous studies were used in this paper. Table 1 shows the details of the defect datasets. It should be noted that the metrics are not the same in every project. This paper have released the codes and datasets at https://github.com/Sevensweett/HCPDP for reproducing the experiment.

### C. EVALUATION MEASURES

Five widely used indicators to assess the predictive performance were applied in this paper, including PD, PF, F1-score, AUC and G-mean [40], [41], [42].

Because the HCPDP task is a binary classification task, the final result can be True Positive (TP) indicating the numbers of actually predicted defective modules, False Positive (FP) that denotes the numbers of incorrectly predicted non-defective modules, True Negative (TN) that denotes the numbers of correctly predicted non-defective modules, or False Negative (FN) that denotes the numbers of incorrectly predicted defective modules. Based on the four possible results, the five evaluation indicators mentioned above are specifically defined as follows:

- PD, PF: PD (aka. recall, true positive rate) and PF (aka. False positive rate) have long been widely used as evaluation indicators for software defect prediction [23], [43], [44], [45], and they analyzed that a high-performing HCPDP model should have higher PD and lower PF.

$$PD = \frac{TP}{TP + FN}; \quad PF = \frac{FP}{FP + TN} \tag{12}$$

- F1-score: Precision is used to evaluate the correctness of a prediction model, and Precision= TP/(TP+FP), and Recall is used to evaluate the possibility of correctly predicted defects, and Recall=TP/(TP+FN). Therefore, F1-score [23], [24], [46] integrates Recall and Precision in a single indicator.

$$F1 - score = \frac{2 \times recall \times precision}{recall + precision} \tag{13}$$

- AUC: AUC (Area Under Curve) [47], [48] is defined as the area enclosed by the coordinate axis under the ROC curve, PD is the X axis of the coordinate axis, and PF is the Y axis. Since AUC is rarely affected by class imbalance as well as being independent from the prediction threshold, it has been widely used for evaluation and analysis of software defect prediction [23], [49], [50].
- G-mean: It is the geometric mean of PD and (1-PF) [51].

$$G - mean = \sqrt{PD \times (1 - PF)} \tag{14}$$

**TABLE 2.** PD comparison between EGW and baselines.

| Target | EGW | ManualDown | ManualUp | CPDP_IFS | HDP_KS | EMKCA | CTKCCA | WPDP |
|---|---|---|---|---|---|---|---|---|
| EQ | 0.539 | **0.721** | 0.271 | 0.473 | 0.443 | 0.145 | 0.392 | 0.435 |
| JDT | 0.656 | **0.786** | 0.214 | 0.647 | 0.524 | 0.057 | 0.200 | 0.447 |
| LC | **0.685** | 0.672 | 0.328 | 0.617 | 0.459 | 0.091 | 0.329 | 0.371 |
| ML | 0.515 | **0.735** | 0.265 | 0.500 | 0.527 | 0.034 | 0.163 | 0.250 |
| PDE | 0.592 | **0.780** | 0.220 | 0.551 | 0.508 | 0.047 | 0.139 | 0.270 |
| activemq-5.0.0 | 0.728 | **0.802** | 0.198 | 0.670 | 0.538 | 0.020 | 0.165 | 0.434 |
| derby-10.5.1.1 | 0.680 | **0.822** | 0.178 | 0.595 | 0.535 | 0.017 | 0.147 | 0.293 |
| groovy-0.94.0 | 0.555 | **0.800** | 0.200 | 0.645 | 0.538 | 0.026 | 0.240 | 0.422 |
| hbase-0.94.0 | 0.617 | **0.812** | 0.188 | 0.584 | 0.495 | 0.018 | 0.205 | 0.446 |
| hive-0.9.0 | 0.626 | **0.820** | 0.180 | 0.511 | 0.494 | 0.017 | 0.218 | 0.415 |
| KC1 | 0.669 | **0.877** | 0.135 | 0.614 | 0.625 | 0.040 | 0.155 | 0.249 |
| MC1 | **0.890** | 0.882 | 0.118 | 0.795 | 0.747 | 0.081 | 0.778 | 0.052 |
| PC1 | 0.570 | **0.803** | 0.197 | 0.590 | 0.568 | 0.212 | 0.408 | 0.269 |
| PC3 | 0.619 | **0.754** | 0.246 | 0.575 | 0.623 | 0.094 | 0.132 | 0.224 |
| PC4 | 0.588 | **0.730** | 0.270 | 0.594 | 0.507 | 0.081 | 0.128 | 0.347 |
| camel-1.6 | 0.506 | **0.617** | 0.383 | 0.477 | 0.456 | 0.073 | 0.210 | 0.193 |
| lucene-2.4 | 0.478 | **0.576** | 0.424 | 0.472 | 0.396 | 0.093 | 0.402 | 0.612 |
| poi-3.0 | 0.410 | **0.641** | 0.359 | 0.514 | 0.392 | 0.106 | 0.358 | **0.742** |
| synapse-1.2 | 0.612 | **0.733** | 0.267 | 0.509 | 0.487 | 0.179 | 0.410 | 0.487 |
| velocity-1.6 | 0.526 | **0.679** | 0.321 | 0.414 | 0.389 | 0.223 | 0.651 | 0.516 |
| Average | 0.603 | **0.752** | 0.248 | 0.567 | 0.513 | 0.083 | 0.292 | 0.374 |

All the above evaluation measures range from 0 to 1 [52]. Obviously, a better HCPDP model should yield higher values for PD, F1-score, AUC, G-mean but lower PF.

### D. EVALUATION SETTINGS

We selected one dataset from 20 datasets as the target, and each of the remaining datasets was used as the source in turn. Because this study focuses on prediction with heterogeneous metrics, we did not perform defect prediction with the same metrics. For example, if we selected EQ in AEEEM as the target items, the other items in AEEEM would not participate in model building as source items.

To avoid the randomness, we repeated the above methods 20 times and reported the average results. For EGW, all source data was transferred to get the same dimension of the target data, and then the transferred source data was used to train the classifier that was applied to predict the target data. For EGW+, all transmitted source data and the 10% randomly labeled target data were used as training data, and the remaining data were selected as test data. The reason for selecting 10% of the target data is that the baselines used the above data splitting in their works, so this paper uses the same method to maintain a fair comparison.

### E. PARAMETER SETTINGS

In our approach, there are two parameters. The parameter $\varepsilon$ is related to the smoothness of the transport scheme. If $\varepsilon$ is too small, the transport matrix $\mathbf{T}$ would become dense. If $\varepsilon$ is too large, the transport matrix $\mathbf{T}$ would become sparse. The parameter $\lambda$ controls the effectiveness of the regular term $\varphi(T)$. If $\lambda$ is too small, the distribution alignment between the labeled source and target data may be neglected. If $\lambda$ is too large, the measurement matrix matching between the source and target data may be invalid.

## V. EXPERIMENTAL RESULTS

### A. ANSWER TO RQ1: The Effectiveness of EGW for HCPDP TASK

#### 1) METHODS

It is worth noting that there are many HCPDP methods without using any target data. Since EGW is a method for analyze the distribution between two projects, we chose those methods only from the perspective of data distributions. Additionally, ManualDown and ManualUp [27] have been reported to be easy to implement and perform better than most HCPDP works, so they have been chosen as the baselines in recent articles although the two methods completed the defect prediction based on themselves and needed a number of historical defective data of target project. Therefore, we also selected them in this paper. To demonstrate the effectiveness of cross-project defect prediction, WPDP is also selected for comparison. The logistic regression (LR) classifier [53] was chosen for all methods. The short descriptions of the compared methods are listed as follows:

- **ManualDown** [27]. It was a simple unsupervised method, which performed better than most of the existing works. ManualDown has been recognized as a new baseline method of CPDP.
- **ManualUp** [27]. It was another unsupervised method proposed by Zhou et al.
- **CPDP_IFS** [25]. It was an instance mapping method.
- **HDP_KS** [9]. In this work, metric selection and matching were conducted to construct a prediction model between projects with heterogeneous features.

**TABLE 3.** PF comparison between EGW and baselines.

| Target | EGW | ManualDown | ManualUp | CPDP_IFS | HDP_KS | EMKCA | CTKCCA | WPDP |
|---|---|---|---|---|---|---|---|---|
| EQ | 0.248 | 0.354 | 0.651 | **0.166** | 0.175 | 0.321 | 0.227 | 0.230 |
| JDT | 0.237 | 0.426 | 0.575 | 0.248 | 0.232 | **0.034** | 0.055 | 0.153 |
| LC | 0.295 | 0.483 | 0.518 | 0.282 | 0.310 | **0.053** | 0.098 | 0.218 |
| ML | 0.289 | 0.464 | 0.536 | 0.304 | 0.288 | **0.023** | 0.041 | 0.077 |
| PDE | 0.325 | 0.455 | 0.546 | 0.294 | 0.308 | **0.023** | 0.043 | 0.086 |
| activemq-5.0.0 | 0.292 | 0.444 | 0.556 | 0.446 | 0.310 | **0.021** | 0.019 | 0.092 |
| derby-10.5.1.1 | 0.274 | 0.447 | 0.553 | 0.283 | 0.280 | 0.015 | **0.011** | 0.052 |
| groovy-0.94.0 | 0.297 | 0.473 | 0.529 | 0.337 | 0.238 | **0.049** | 0.097 | 0.145 |
| hbase-0.94.0 | 0.290 | 0.420 | 0.581 | 0.445 | 0.297 | **0.042** | 0.039 | 0.173 |
| hive-0.9.0 | 0.265 | 0.420 | 0.580 | 0.320 | 0.296 | 0.030 | **0.025** | 0.106 |
| KC1 | 0.247 | 0.431 | 0.567 | 0.234 | 0.239 | 0.062 | 0.229 | **0.043** |
| MC1 | 0.283 | 0.497 | 0.503 | 0.266 | 0.254 | 0.102 | 0.295 | **0.002** |
| PC1 | 0.310 | 0.473 | 0.528 | 0.290 | 0.295 | **0.041** | 0.101 | 0.082 |
| PC3 | 0.276 | 0.464 | 0.537 | 0.286 | 0.307 | **0.034** | 0.074 | 0.069 |
| PC4 | 0.295 | 0.466 | 0.535 | 0.283 | 0.297 | **0.023** | 0.080 | 0.047 |
| camel-1.6 | 0.369 | 0.472 | 0.529 | 0.363 | 0.336 | **0.053** | 0.134 | 0.085 |
| lucene-2.4 | 0.235 | 0.387 | 0.613 | 0.225 | 0.274 | 0.139 | **0.000** | 0.464 |
| poi-3.0 | 0.215 | 0.255 | 0.745 | 0.172 | 0.252 | 0.098 | **0.064** | 0.441 |
| synapse-1.2 | 0.284 | 0.382 | 0.618 | 0.271 | 0.308 | **0.152** | 0.300 | 0.317 |
| velocity-1.6 | 0.300 | 0.411 | 0.596 | 0.266 | 0.295 | **0.193** | 0.243 | 0.427 |
| Average | 0.281 | 0.431 | 0.570 | 0.289 | 0.280 | **0.075** | 0.109 | 0.165 |

**TABLE 4.** F1-score comparison between EGW and baselines.

| Target | EGW | ManualDown | ManualUp | CPDP_IFS | HDP_KS | EMKCA | CTKCCA | WPDP |
|---|---|---|---|---|---|---|---|---|
| EQ | 0.574 | **0.639** | 0.241 | 0.538 | 0.515 | 0.177 | 0.452 | 0.484 |
| JDT | **0.532** | 0.460 | 0.125 | 0.502 | 0.447 | 0.095 | 0.284 | 0.436 |
| LC | **0.314** | 0.210 | 0.102 | 0.287 | 0.216 | 0.113 | 0.288 | 0.213 |
| ML | **0.314** | 0.306 | 0.111 | 0.291 | 0.312 | 0.057 | 0.227 | 0.282 |
| PDE | **0.346** | 0.340 | 0.096 | 0.330 | 0.300 | 0.079 | 0.198 | 0.298 |
| activemq-5.0.0 | **0.454** | 0.381 | 0.094 | 0.344 | 0.335 | 0.035 | 0.260 | 0.446 |
| derby-10.5.1.1 | **0.425** | 0.363 | 0.078 | 0.354 | 0.342 | 0.030 | 0.242 | 0.362 |
| groovy-0.94.0 | 0.251 | 0.233 | 0.058 | 0.240 | **0.267** | 0.033 | 0.211 | 0.280 |
| hbase-0.94.0 | 0.469 | **0.473** | 0.110 | 0.349 | 0.390 | 0.031 | 0.302 | 0.421 |
| hive-0.9.0 | **0.495** | 0.468 | 0.103 | 0.362 | 0.384 | 0.030 | 0.331 | 0.450 |
| KC1 | **0.444** | 0.415 | 0.064 | 0.423 | 0.426 | 0.058 | 0.134 | 0.334 |
| MC1 | 0.047 | 0.027 | 0.004 | **0.048** | 0.048 | 0.012 | 0.042 | 0.070 |
| PC1 | 0.230 | 0.228 | 0.056 | 0.250 | 0.238 | 0.254 | **0.325** | 0.241 |
| PC3 | **0.349** | 0.302 | 0.099 | 0.320 | 0.332 | 0.141 | 0.160 | 0.259 |
| PC4 | 0.329 | 0.300 | 0.111 | **0.339** | 0.293 | 0.131 | 0.155 | 0.409 |
| camel-1.6 | 0.337 | **0.346** | 0.215 | 0.321 | 0.321 | 0.113 | 0.239 | 0.242 |
| lucene-2.4 | 0.586 | 0.627 | 0.461 | 0.579 | 0.495 | 0.156 | 0.573 | **0.633** |
| poi-3.0 | 0.535 | 0.717 | 0.402 | 0.634 | 0.504 | 0.182 | 0.512 | **0.742** |
| synapse-1.2 | 0.575 | **0.589** | 0.215 | 0.493 | 0.463 | 0.242 | 0.411 | 0.459 |
| velocity-1.6 | 0.506 | 0.549 | 0.259 | 0.417 | 0.392 | 0.279 | **0.613** | 0.440 |
| Average | **0.406** | 0.399 | 0.150 | 0.371 | 0.351 | 0.112 | 0.298 | 0.375 |

- **EMKCA** [54]. It combined the advantages of multi-core learning and domain adaptive technology.
- **CTKCCA** [23]. In this work, cost sensitive learning was used to alleviate the class imbalance problem, and the transfer kernel canonical correlation analysis was used to transform the source projects and target projects.
- **WPDP** [55]. It used 10% of the label data in the project to build a defect prediction model, and the remaining 90% of the data was used as a test set to verify.

### 2) RESULTS

Tables 2-6 show the PD, PF, F1-score, AUC and G-mean values of EGW compared with baselines. The last row in the two tables represents the average values across 20 target items and the best result is in bold font. As can be seen in Tables 2-3, although the PD of ManualDown achieved the highest, PF was also very high because this method used model size in the target project to build simple prediction model. For imbalanced datasets, it would bring a high PF rate. On the contrary, EMKCA had the lowest PF, but its PD was also the lowest. To sum up, the overall performance of EGW on PD and PF was much better. In Tables 4-6, compared with ManualDown, ManualUp, CPDP_IFS, HDP_KS, EMKCA, CTKCCA and WPDP, EGW can improve the F1-score by 0.7%, 25.6%, 3.5%, 5.5%, 29.4%, 10.8%, and 3.1%. For AUC, it is improved by 0.1%, 32.2%, 2.2%, 0.5%, 11.4%, 8.1%, and 1.6% and G-mean was improved by 0.1%, 33.3%, 4.8%, 6.9%, 50.9%, 24.8% and 15.3%. Although the result difference between EGW and ManualDown was not large in F1, AUC, and G-mean, the PF of ManualDown is 15% higher

**TABLE 5.** AUC comparison between EGW and baselines.

| Target | EGW | ManualDown | ManualUp | CPDP_IFS | HDP_KS | EMKCA | CTKCCA | WPDP |
|---|---|---|---|---|---|---|---|---|
| EQ | 0.646 | 0.684 | 0.310 | 0.653 | **0.714** | 0.440 | 0.510 | 0.544 |
| JDT | **0.709** | 0.680 | 0.319 | 0.700 | 0.682 | 0.537 | 0.521 | 0.612 |
| LC | **0.695** | 0.594 | 0.405 | 0.668 | 0.588 | 0.545 | 0.601 | 0.505 |
| ML | 0.613 | 0.635 | 0.365 | 0.598 | **0.643** | 0.534 | 0.543 | 0.630 |
| PDE | 0.633 | **0.662** | 0.337 | 0.629 | 0.616 | 0.550 | 0.517 | 0.607 |
| activemq-5.0.0 | 0.718 | 0.679 | 0.321 | 0.612 | 0.636 | 0.540 | 0.574 | **0.750** |
| derby-10.5.1.1 | 0.703 | 0.688 | 0.312 | 0.656 | 0.663 | 0.570 | 0.567 | **0.709** |
| groovy-0.94.0 | 0.629 | 0.664 | 0.336 | 0.654 | **0.707** | 0.506 | 0.575 | 0.638 |
| hbase-0.94.0 | 0.663 | **0.696** | 0.303 | 0.570 | 0.639 | 0.547 | 0.581 | 0.649 |
| hive-0.9.0 | 0.681 | **0.700** | 0.300 | 0.596 | 0.632 | 0.537 | 0.600 | 0.633 |
| KC1 | 0.711 | 0.723 | 0.284 | 0.690 | **0.751** | 0.573 | 0.341 | 0.740 |
| MC1 | 0.803 | 0.693 | 0.307 | 0.765 | 0.799 | 0.522 | **0.860** | 0.744 |
| PC1 | 0.630 | 0.665 | 0.334 | 0.650 | **0.695** | 0.649 | 0.663 | 0.633 |
| PC3 | 0.672 | 0.645 | 0.355 | 0.645 | **0.697** | 0.646 | 0.487 | 0.677 |
| PC4 | 0.647 | 0.632 | 0.368 | 0.655 | 0.652 | 0.631 | 0.503 | **0.812** |
| camel-1.6 | 0.568 | 0.572 | 0.427 | 0.557 | 0.581 | 0.524 | 0.502 | **0.618** |
| lucene-2.4 | 0.621 | 0.595 | 0.405 | 0.623 | 0.596 | 0.516 | **0.700** | 0.603 |
| poi-3.0 | 0.597 | 0.693 | 0.307 | 0.671 | 0.626 | 0.487 | **0.743** | 0.673 |
| synapse-1.2 | 0.664 | **0.675** | 0.325 | 0.619 | 0.636 | 0.544 | 0.487 | 0.583 |
| velocity-1.6 | 0.613 | 0.634 | 0.362 | 0.574 | 0.565 | 0.538 | **0.718** | 0.536 |
| Average | **0.661** | 0.660 | 0.339 | 0.639 | 0.656 | 0.547 | 0.580 | 0.645 |

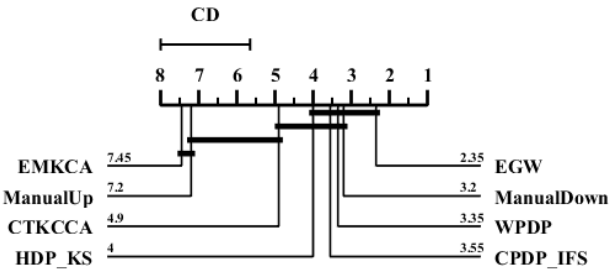



**FIGURE 2.** Comparison of average ranks in F1-score of EGW.

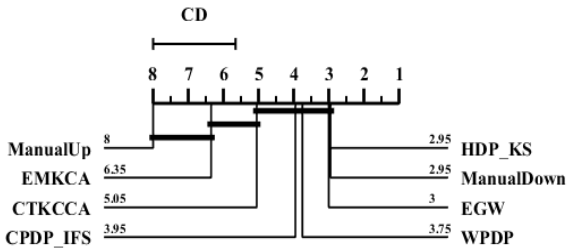



**FIGURE 3.** Comparison of average ranks in AUC of EGW.

than that of EGW, which indicated that more non-defective samples were predicted to be defective.

To statistical analyze the performance of the above methods, we conducted the Friedman test with Nemenyi test as a post hoc test with a 95% confidence level [52] used in some CPDP studies [13], [36], [56]. To visualize the differences, CD diagrams [57] can be obtained by

$$CD = q_{(\alpha,L)}\sqrt{\frac{L(L+1)}{6M}} \tag{15}$$

where $L$ represents the number of the method, $M$ is the cross-project pair's number, and $q$ $(\alpha, L)$ is a critical value based on $L$. In the CD diagrams, the average rank of each method is marked along the axis (higher ranks to the left). We connected them with a thick line if these approaches are not significantly different under the Nemenyi test.

Most previous works selected F1-score, G-mean and AUC to perform the statistical analysis. Therefore, as shown in Figs 2-4, we have also provided the analysis of the three indicators. It is shown that EGW performs better than all other methods for F1-score and G-mean. For AUC, EGW was also in the top three. In general, EGW outperformed other methods in HCPDP task.

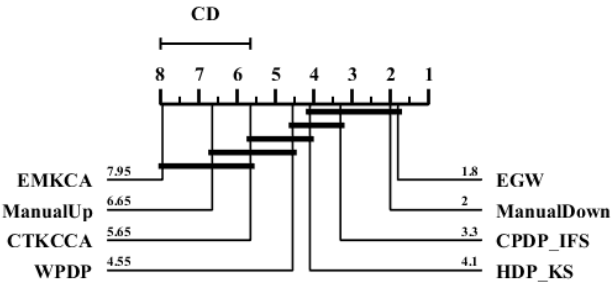



**FIGURE 4.** Comparison of average ranks in G-mean of EGW.

Then, Cliff's delta [58], [59] was applied to calculate the effect size of F1-score, AUC and G-mean between EGW and baselines values. The mappings between $\delta$ and its level are given in Table 7.

By analyzing the effect size of the three indicators in Table 8 compared to other methods, there are four negligible differences and seventeen non-negligible differences, suggesting that EGW performs better than most methods. It is worth noting that EGW and ManualDown do not have a significant difference in performance. Therefore, EGW+ was

**TABLE 6.** G-mean comparison between EGW and baselines.

| Target | EGW | ManualDown | ManualUp | CPDP_IFS | HDP_KS | EMKCA | CTKCCA | WPDP |
|---|---|---|---|---|---|---|---|---|
| EQ | 0.630 | **0.683** | 0.308 | 0.584 | 0.568 | 0.238 | 0.519 | 0.573 |
| JDT | **0.705** | 0.672 | 0.301 | 0.689 | 0.617 | 0.107 | 0.330 | 0.610 |
| LC | **0.695** | 0.589 | 0.398 | 0.643 | 0.536 | 0.165 | 0.481 | 0.532 |
| ML | 0.603 | **0.627** | 0.351 | 0.574 | 0.597 | 0.066 | 0.278 | 0.477 |
| PDE | 0.631 | **0.652** | 0.316 | 0.605 | 0.574 | 0.090 | 0.242 | 0.494 |
| activemq-5.0.0 | **0.717** | 0.668 | 0.297 | 0.582 | 0.594 | 0.039 | 0.282 | 0.626 |
| derby-10.5.1.1 | **0.702** | 0.674 | 0.282 | 0.620 | 0.610 | 0.033 | 0.255 | 0.525 |
| groovy-0.94.0 | 0.622 | **0.649** | 0.307 | 0.625 | 0.622 | 0.050 | 0.376 | 0.594 |
| hbase-0.94.0 | 0.659 | **0.686** | 0.281 | 0.532 | 0.569 | 0.035 | 0.337 | 0.605 |
| hive-0.9.0 | 0.676 | **0.689** | 0.275 | 0.554 | 0.561 | 0.033 | 0.355 | 0.607 |
| KC1 | **0.708** | 0.706 | 0.242 | 0.660 | 0.681 | 0.077 | 0.258 | 0.486 |
| MC1 | **0.798** | 0.666 | 0.242 | 0.733 | 0.737 | 0.149 | 0.738 | 0.181 |
| PC1 | 0.627 | **0.650** | 0.305 | 0.626 | 0.621 | 0.347 | 0.561 | 0.487 |
| PC3 | **0.667** | 0.636 | 0.338 | 0.615 | 0.647 | 0.171 | 0.230 | 0.453 |
| PC4 | **0.644** | 0.624 | 0.354 | 0.631 | 0.585 | 0.150 | 0.224 | 0.568 |
| camel-1.6 | 0.563 | **0.571** | 0.425 | 0.542 | 0.528 | 0.136 | 0.337 | 0.410 |
| lucene-2.4 | **0.597** | 0.594 | 0.405 | 0.580 | 0.494 | 0.168 | 0.573 | 0.569 |
| poi-3.0 | 0.550 | **0.691** | 0.303 | 0.628 | 0.492 | 0.190 | 0.516 | 0.636 |
| synapse-1.2 | 0.660 | **0.673** | 0.320 | 0.584 | 0.559 | 0.295 | 0.516 | 0.569 |
| velocity-1.6 | 0.605 | 0.633 | 0.360 | 0.506 | 0.483 | 0.349 | **0.698** | 0.538 |
| Average | **0.653** | 0.652 | 0.320 | 0.605 | 0.584 | 0.144 | 0.405 | 0.527 |

**TABLE 7.** Mappings $\delta$ between and its level.

| Cliff's Delta ($\delta$) | Level |
|---|---|
| $\lvert\delta\rvert < 0.147$ | Negligible (N) |
| $0.147 \le \lvert\delta\rvert < 0.333$ | Small (S) |
| $0.333 \le \lvert\delta\rvert < 0.474$ | Medium (M) |
| $\lvert\delta\rvert \ge 0.474$ | Large (L) |

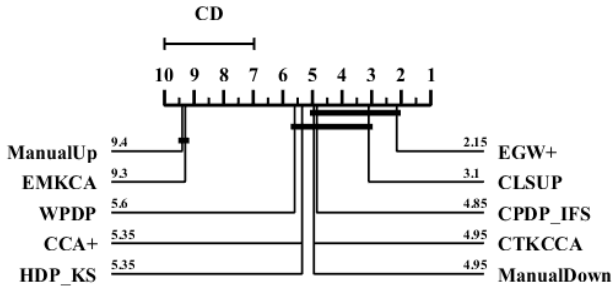



**FIGURE 5.** Comparison of average ranks in F1-score of EGW+.

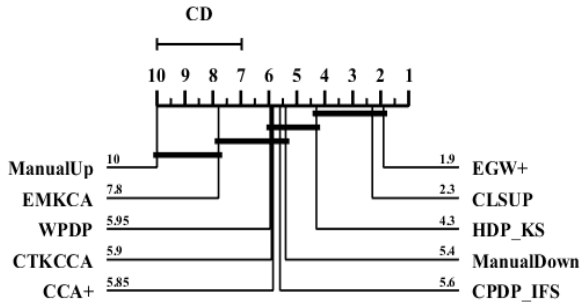



**FIGURE 6.** Comparison of average ranks in AUC of EGW+.

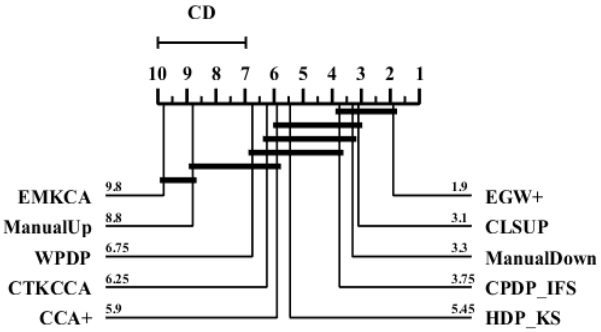



**FIGURE 7.** Comparison of average ranks in AUC of EGW+.

developed and the concrete results are shown in the next section.

### *B. ANSWER To RQ2: The Effectiveness of EGW+*

#### 1) METHODS

There are several methods that used target data to construct HCPDP model, but we found that those methods used different datasets. For a fair comparison, we selected the methods that have common datasets. For CCA+ [24] and CLSUP [23], the two methods also used 10% of the target data during the experiments. Meanwhile, we randomly added 10% of the target data for CPDP_IFS, HDP_LS, EMKCA, and CTKCCA to ensure all the baselines use the same datasets. Similarly, ManualDown and ManualUp, and WPDP were still considered, and logistic regression (LR) was used as the basic classifier. Since some baselines were introduced. Here, we have only given the brief descriptions of CCA+ and CLSUP.

- **CCA+**. It introduced canonical correlation analysis into HCPDP model.
- **CLSUP**. A cost-sensitive label and structure-consistent unilateral projection approach was proposed to solve

**TABLE 8.** Cliff's delta of EGW versus baselines.

| | ManualDown | ManualUp | CPDP_IFS | HDP_KS | EMKCA | CTKCCA | WPDP |
|---|---|---|---|---|---|---|---|
| F1-score | 0.060(N) | 0.795(L) | 0.155(S) | 0.285(S) | 0.905(L) | 0.475(L) | 0.210(S) |
| AUC | -0.055(N) | 1.000(L) | 0.245(S) | 0.070(N) | 0.860(L) | 0.550(L) | 0.150(S) |
| G-mean | -0.020(N) | 1.000(L) | 0.460(M) | 0.615(L) | 1.000(L) | 0.815(L) | 0.800(L) |

**TABLE 9.** PD comparison between EGW+ and baselines.

| Target | EGW+ (10%) | ManualDown | ManualUp | CPDP_IFS (10%) | HDP_KS (10%) | EMKCA (10%) | CTKCCA (10%) | WPDP | CCA+ (10%) | CLSUP (10%) |
|---|---|---|---|---|---|---|---|---|---|---|
| EQ | 0.646 | **0.721** | 0.271 | 0.508 | 0.425 | 0.205 | 0.306 | 0.435 | 0.507 | 0.546 |
| JDT | 0.672 | **0.786** | 0.214 | 0.647 | 0.562 | 0.069 | 0.192 | 0.447 | 0.618 | 0.623 |
| LC | 0.606 | 0.672 | 0.328 | 0.636 | 0.488 | 0.145 | 0.617 | 0.371 | 0.647 | **0.678** |
| ML | 0.583 | **0.735** | 0.265 | 0.527 | 0.538 | 0.047 | 0.204 | 0.250 | 0.500 | 0.529 |
| PDE | 0.597 | **0.780** | 0.220 | 0.560 | 0.536 | 0.065 | 0.189 | 0.270 | 0.544 | 0.593 |
| activemq-5.0.0 | 0.792 | **0.802** | 0.198 | 0.722 | 0.554 | 0.045 | 0.135 | 0.434 | 0.642 | 0.695 |
| derby-10.5.1.1 | 0.666 | **0.822** | 0.178 | 0.719 | 0.584 | 0.027 | 0.103 | 0.293 | 0.542 | 0.594 |
| groovy-0.94.0 | 0.733 | **0.800** | 0.200 | 0.720 | 0.549 | 0.051 | 0.564 | 0.422 | 0.482 | 0.590 |
| hbase-0.94.0 | 0.716 | **0.812** | 0.188 | 0.646 | 0.540 | 0.032 | 0.181 | 0.446 | 0.404 | 0.558 |
| hive-0.9.0 | 0.661 | **0.820** | 0.180 | 0.608 | 0.545 | 0.020 | 0.143 | 0.415 | 0.456 | 0.536 |
| KC1 | 0.651 | **0.877** | 0.135 | 0.663 | 0.622 | 0.051 | 0.178 | 0.249 | 0.444 | 0.554 |
| MC1 | 0.591 | 0.882 | 0.118 | 0.785 | 0.646 | 0.088 | **0.939** | 0.052 | 0.753 | 0.732 |
| PC1 | 0.742 | **0.803** | 0.197 | 0.595 | 0.580 | 0.240 | 0.671 | 0.269 | 0.448 | 0.515 |
| PC3 | 0.732 | **0.754** | 0.246 | 0.621 | 0.652 | 0.137 | 0.304 | 0.224 | 0.367 | 0.595 |
| PC4 | **0.819** | 0.730 | 0.270 | 0.681 | 0.563 | 0.117 | 0.230 | 0.347 | 0.393 | 0.562 |
| camel-1.6 | 0.598 | **0.617** | 0.383 | 0.466 | 0.418 | 0.082 | 0.226 | 0.193 | 0.403 | 0.463 |
| lucene-2.4 | **0.620** | 0.576 | 0.424 | 0.483 | 0.417 | 0.156 | 0.209 | 0.612 | 0.415 | 0.490 |
| poi-3.0 | 0.569 | 0.641 | 0.359 | 0.541 | 0.408 | 0.150 | 0.260 | **0.742** | 0.327 | 0.493 |
| synapse-1.2 | 0.709 | **0.733** | 0.267 | 0.535 | 0.512 | 0.235 | 0.459 | 0.487 | 0.551 | 0.574 |
| velocity-1.6 | 0.660 | **0.679** | 0.321 | 0.428 | 0.457 | 0.294 | 0.556 | 0.516 | 0.401 | 0.495 |
| Average | 0.668 | **0.752** | 0.248 | 0.604 | 0.530 | 0.113 | 0.333 | 0.374 | 0.492 | 0.571 |

**TABLE 10.** PF comparison between EGW+ and baselines.

| Target | EGW+ (10%) | ManualDown | ManualUp | CPDP_IFS (10%) | HDP_KS (10%) | EMKCA (10%) | CTKCCA (10%) | WPDP | CCA+ (10%) | CLSUP (10%) |
|---|---|---|---|---|---|---|---|---|---|---|
| EQ | 0.566 | **0.639** | 0.241 | 0.566 | 0.507 | 0.267 | 0.468 | 0.484 | 0.554 | 0.606 |
| JDT | 0.525 | 0.460 | 0.125 | 0.532 | 0.493 | 0.109 | 0.322 | 0.436 | **0.534** | 0.533 |
| LC | 0.268 | 0.210 | 0.102 | 0.295 | 0.237 | 0.156 | **0.763** | 0.213 | 0.314 | 0.312 |
| ML | **0.356** | 0.306 | 0.111 | 0.318 | 0.338 | 0.073 | 0.330 | 0.282 | 0.326 | 0.354 |
| PDE | 0.359 | 0.340 | 0.096 | 0.340 | 0.321 | 0.101 | 0.318 | 0.298 | 0.346 | **0.367** |
| activemq-5.0.0 | **0.519** | 0.381 | 0.094 | 0.435 | 0.358 | 0.074 | 0.237 | 0.446 | 0.449 | 0.476 |
| derby-10.5.1.1 | **0.452** | 0.363 | 0.078 | 0.392 | 0.382 | 0.046 | 0.187 | 0.362 | 0.394 | 0.413 |
| groovy-0.94.0 | 0.295 | 0.233 | 0.058 | 0.271 | 0.273 | 0.057 | **0.721** | 0.280 | 0.269 | 0.283 |
| hbase-0.94.0 | **0.514** | 0.473 | 0.110 | 0.434 | 0.441 | 0.051 | 0.307 | 0.421 | 0.347 | 0.451 |
| hive-0.9.0 | **0.518** | 0.468 | 0.103 | 0.448 | 0.428 | 0.033 | 0.250 | 0.450 | 0.410 | 0.445 |
| KC1 | **0.443** | 0.415 | 0.064 | 0.435 | 0.427 | 0.071 | 0.217 | 0.334 | 0.400 | 0.443 |
| MC1 | 0.109 | 0.027 | 0.004 | 0.043 | 0.043 | 0.010 | **0.598** | 0.070 | 0.076 | 0.081 |
| PC1 | 0.279 | 0.228 | 0.056 | 0.246 | 0.246 | 0.242 | **0.780** | 0.241 | 0.241 | 0.241 |
| PC3 | 0.381 | 0.302 | 0.099 | 0.327 | 0.352 | 0.185 | **0.466** | 0.259 | 0.258 | 0.356 |
| PC4 | **0.436** | 0.300 | 0.111 | 0.353 | 0.328 | 0.174 | 0.367 | 0.409 | 0.255 | 0.355 |
| camel-1.6 | **0.391** | 0.346 | 0.215 | 0.318 | 0.326 | 0.119 | 0.354 | 0.242 | 0.327 | 0.355 |
| lucene-2.4 | 0.540 | 0.627 | 0.461 | 0.587 | 0.519 | 0.242 | 0.346 | **0.633** | 0.531 | 0.611 |
| poi-3.0 | 0.653 | 0.717 | 0.402 | 0.657 | 0.532 | 0.241 | 0.398 | **0.742** | 0.424 | 0.624 |
| synapse-1.2 | **0.776** | 0.589 | 0.215 | 0.511 | 0.488 | 0.278 | 0.629 | 0.459 | 0.537 | 0.551 |
| velocity-1.6 | 0.584 | 0.549 | 0.259 | 0.429 | 0.457 | 0.328 | **0.715** | 0.440 | 0.409 | 0.481 |
| Average | **0.448** | 0.399 | 0.150 | 0.397 | 0.375 | 0.143 | 0.439 | 0.375 | 0.370 | 0.417 |

HCPDP. It alleviated the class imbalance problem by cost-sensitive analysis.

### 2) RESULTS

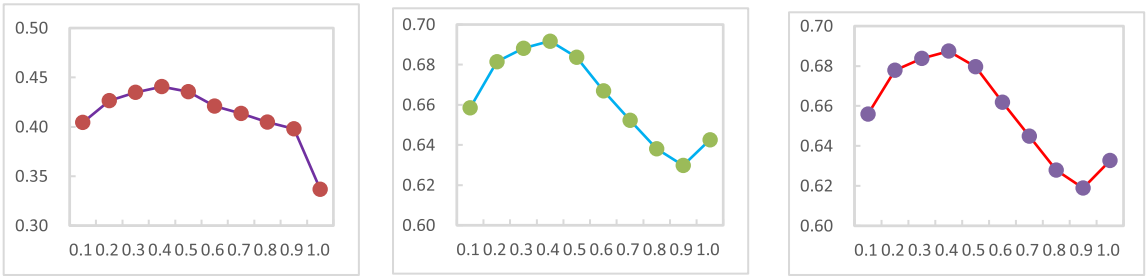
In Tables 9-10, it can be seen that the PD of ManualDown was still higher than EGW+ and PF of CTKCCA was lower than EGW+, but PF of ManualDown was higher than EGW+ and PD of CTKCCA was lower than EGW+. Therefore, considering PD and PF comprehensively, EGW+ had the more stable performance. Furthermore, Tables 11-13 shows the F1-score, AUC and G-mean results of EGW+ and baseline methods. We found that EGW+ performs better than

**TABLE 11.** F1-score comparison between EGW+ and baselines.

| Target | EGW+ (10%) | ManualDown | ManualUp | CPDP_IFS (10%) | HDP_KS (10%) | EMKCA (10%) | CTKCCA (10%) | WPDP | CCA+ (10%) | CLSUP (10%) |
|---|---|---|---|---|---|---|---|---|---|---|
| EQ | 0.607 | 0.684 | 0.310 | 0.670 | 0.718 | 0.521 | 0.531 | 0.544 | 0.671 | **0.808** |
| JDT | 0.756 | 0.680 | 0.319 | 0.720 | 0.729 | 0.587 | 0.568 | 0.612 | 0.730 | **0.763** |
| LC | 0.666 | 0.594 | 0.405 | 0.678 | 0.630 | 0.581 | **0.809** | 0.505 | 0.721 | 0.777 |
| ML | **0.713** | 0.635 | 0.365 | 0.628 | 0.679 | 0.577 | 0.597 | 0.630 | 0.640 | 0.712 |
| PDE | 0.697 | 0.662 | 0.337 | 0.640 | 0.652 | 0.591 | 0.592 | 0.607 | 0.659 | **0.719** |
| activemq-5.0.0 | **0.844** | 0.679 | 0.321 | 0.712 | 0.676 | 0.584 | 0.566 | 0.750 | 0.733 | 0.804 |
| derby-10.5.1.1 | **0.788** | 0.688 | 0.312 | 0.699 | 0.726 | 0.614 | 0.550 | 0.709 | 0.683 | 0.750 |
| groovy-0.94.0 | 0.760 | 0.664 | 0.336 | 0.694 | 0.716 | 0.555 | **0.806** | 0.638 | 0.653 | 0.766 |
| hbase-0.94.0 | **0.768** | 0.696 | 0.303 | 0.651 | 0.711 | 0.592 | 0.589 | 0.649 | 0.618 | 0.717 |
| hive-0.9.0 | **0.760** | 0.700 | 0.300 | 0.665 | 0.695 | 0.587 | 0.587 | 0.633 | 0.664 | 0.715 |
| KC1 | **0.789** | 0.723 | 0.284 | 0.704 | 0.757 | 0.557 | 0.510 | 0.740 | 0.678 | 0.705 |
| MC1 | 0.897 | 0.693 | 0.307 | 0.737 | 0.697 | 0.678 | **0.975** | 0.744 | 0.813 | 0.827 |
| PC1 | 0.784 | 0.665 | 0.334 | 0.648 | 0.709 | 0.767 | **0.869** | 0.633 | 0.605 | 0.684 |
| PC3 | **0.773** | 0.645 | 0.355 | 0.655 | 0.725 | 0.718 | 0.655 | 0.677 | 0.606 | 0.731 |
| PC4 | **0.848** | 0.632 | 0.368 | 0.678 | 0.698 | 0.685 | 0.628 | 0.812 | 0.589 | 0.705 |
| camel-1.6 | **0.679** | 0.572 | 0.427 | 0.558 | 0.601 | 0.518 | 0.590 | 0.618 | 0.583 | 0.608 |
| lucene-2.4 | 0.688 | 0.595 | 0.405 | 0.626 | 0.619 | 0.540 | 0.609 | 0.603 | 0.633 | **0.721** |
| poi-3.0 | 0.722 | 0.693 | 0.307 | 0.687 | 0.673 | 0.538 | 0.656 | 0.673 | 0.488 | **0.742** |
| synapse-1.2 | **0.801** | 0.675 | 0.325 | 0.632 | 0.660 | 0.548 | 0.707 | 0.583 | 0.670 | 0.700 |
| velocity-1.6 | 0.730 | 0.634 | 0.362 | 0.577 | 0.622 | 0.558 | **0.807** | 0.536 | 0.576 | 0.649 |
| Average | **0.754** | 0.660 | 0.339 | 0.663 | 0.685 | 0.595 | 0.660 | 0.645 | 0.651 | 0.730 |

**TABLE 12.** AUC comparison between EGW+ and baselines.

| Target | EGW+ (10%) | ManualDown | ManualUp | CPDP_IFS (10%) | HDP_KS (10%) | EMKCA (10%) | CTKCCA (10%) | WPDP | CCA+ (10%) | CLSUP (10%) |
|---|---|---|---|---|---|---|---|---|---|---|
| EQ | 0.609 | **0.683** | 0.308 | 0.612 | 0.558 | 0.323 | 0.468 | 0.573 | 0.588 | 0.659 |
| JDT | **0.717** | 0.672 | 0.301 | 0.708 | 0.656 | 0.129 | 0.322 | 0.610 | 0.684 | 0.704 |
| LC | 0.647 | 0.589 | 0.398 | 0.660 | 0.567 | 0.249 | **0.763** | 0.532 | 0.679 | 0.700 |
| ML | **0.653** | 0.627 | 0.351 | 0.605 | 0.618 | 0.089 | 0.339 | 0.477 | 0.582 | 0.629 |
| PDE | 0.651 | 0.652 | 0.316 | 0.617 | 0.598 | 0.122 | 0.318 | 0.494 | 0.612 | **0.656** |
| activemq-5.0.0 | **0.778** | 0.668 | 0.297 | 0.704 | 0.613 | 0.085 | 0.237 | 0.626 | 0.685 | 0.731 |
| derby-10.5.1.1 | **0.722** | 0.674 | 0.282 | 0.690 | 0.654 | 0.052 | 0.187 | 0.525 | 0.623 | 0.677 |
| groovy-0.94.0 | 0.711 | 0.649 | 0.307 | 0.673 | 0.629 | 0.097 | **0.721** | 0.594 | 0.573 | 0.660 |
| hbase-0.94.0 | **0.718** | 0.686 | 0.281 | 0.629 | 0.623 | 0.062 | 0.307 | 0.605 | 0.474 | 0.638 |
| hive-0.9.0 | **0.715** | 0.689 | 0.275 | 0.645 | 0.619 | 0.039 | 0.250 | 0.607 | 0.544 | 0.634 |
| KC1 | 0.701 | **0.706** | 0.242 | 0.690 | 0.681 | 0.096 | 0.298 | 0.486 | 0.555 | 0.661 |
| MC1 | 0.674 | 0.666 | 0.242 | 0.720 | 0.661 | 0.160 | **0.964** | 0.181 | 0.771 | 0.791 |
| PC1 | 0.702 | 0.650 | 0.305 | 0.628 | 0.633 | 0.381 | **0.802** | 0.487 | 0.533 | 0.606 |
| PC3 | **0.712** | 0.636 | 0.338 | 0.643 | 0.668 | 0.240 | 0.466 | 0.453 | 0.447 | 0.658 |
| PC4 | **0.759** | 0.624 | 0.354 | 0.663 | 0.626 | 0.208 | 0.374 | 0.568 | 0.487 | 0.644 |
| camel-1.6 | **0.618** | 0.571 | 0.425 | 0.531 | 0.520 | 0.151 | 0.368 | 0.410 | 0.505 | 0.564 |
| lucene-2.4 | **0.632** | 0.594 | 0.405 | 0.585 | 0.522 | 0.260 | 0.346 | 0.569 | 0.524 | 0.615 |
| poi-3.0 | 0.651 | **0.691** | 0.303 | 0.647 | 0.531 | 0.253 | 0.404 | 0.636 | 0.417 | 0.623 |
| synapse-1.2 | **0.750** | 0.673 | 0.320 | 0.601 | 0.581 | 0.358 | 0.629 | 0.569 | 0.622 | 0.645 |
| velocity-1.6 | 0.674 | 0.633 | 0.360 | 0.520 | 0.551 | 0.418 | **0.715** | 0.538 | 0.487 | 0.579 |
| Average | **0.690** | 0.652 | 0.320 | 0.639 | 0.605 | 0.189 | 0.464 | 0.527 | 0.570 | 0.654 |




(a) F1-score (b) AUC (c) G-mean

**FIGURE 8.** The variances of $\varepsilon$ with three indicators in EGW.

the other methods. Compared with ManualDown, ManualUp, CPDP_IFS, HDP_KS, EMKCA, CTKCCA, WPDP, CCA+ and CLSUP, EGW+ can improve F1-score by 4.9%, 29.8%, 5.1%, 7.3%, 30.5%, 0.9%, 7.3%, 7.8%, and 3.1%, AUC by 9.4%, 41.5%, 9.1%, 6.9%, 15.9%, 9.4%, 10.9%, 10.3%, and 2.4%. For G-mean, it was improved by

**TABLE 13.** G-mean comparison between EGW+ and baselines.

| Target | EGW+ (10%) | ManualDown | ManualUp | CPDP_IFS (10%) | HDP_KS (10%) | EMKCA (10%) | CTKCCA (10%) | WPDP | CCA+ (10%) | CLSUP (10%) |
|---|---|---|---|---|---|---|---|---|---|---|
| EQ | 0.609 | **0.683** | 0.308 | 0.612 | 0.558 | 0.323 | 0.468 | 0.573 | 0.588 | 0.659 |
| JDT | **0.717** | 0.672 | 0.301 | 0.708 | 0.656 | 0.129 | 0.322 | 0.610 | 0.684 | 0.704 |
| LC | 0.647 | 0.589 | 0.398 | 0.660 | 0.567 | 0.249 | **0.763** | 0.532 | 0.679 | 0.700 |
| ML | **0.653** | 0.627 | 0.351 | 0.605 | 0.618 | 0.089 | 0.339 | 0.477 | 0.582 | 0.629 |
| PDE | 0.651 | 0.652 | 0.316 | 0.617 | 0.598 | 0.122 | 0.318 | 0.494 | 0.612 | **0.656** |
| activemq-5.0.0 | **0.778** | 0.668 | 0.297 | 0.704 | 0.613 | 0.085 | 0.237 | 0.626 | 0.685 | 0.731 |
| derby-10.5.1.1 | **0.722** | 0.674 | 0.282 | 0.690 | 0.654 | 0.052 | 0.187 | 0.525 | 0.623 | 0.677 |
| groovy-0.94.0 | 0.711 | 0.649 | 0.307 | 0.673 | 0.629 | 0.097 | **0.721** | 0.594 | 0.573 | 0.660 |
| hbase-0.94.0 | **0.718** | 0.686 | 0.281 | 0.629 | 0.623 | 0.062 | 0.307 | 0.605 | 0.474 | 0.638 |
| hive-0.9.0 | **0.715** | 0.689 | 0.275 | 0.645 | 0.619 | 0.039 | 0.250 | 0.607 | 0.544 | 0.634 |
| KC1 | 0.701 | **0.706** | 0.242 | 0.690 | 0.681 | 0.096 | 0.298 | 0.486 | 0.555 | 0.661 |
| MC1 | 0.674 | 0.666 | 0.242 | 0.720 | 0.661 | 0.160 | **0.964** | 0.181 | 0.771 | 0.791 |
| PC1 | 0.702 | 0.650 | 0.305 | 0.628 | 0.633 | 0.381 | **0.802** | 0.487 | 0.533 | 0.606 |
| PC3 | **0.712** | 0.636 | 0.338 | 0.643 | 0.668 | 0.240 | 0.466 | 0.453 | 0.447 | 0.658 |
| PC4 | **0.759** | 0.624 | 0.354 | 0.663 | 0.626 | 0.208 | 0.374 | 0.568 | 0.487 | 0.644 |
| camel-1.6 | **0.618** | 0.571 | 0.425 | 0.531 | 0.520 | 0.151 | 0.368 | 0.410 | 0.505 | 0.564 |
| lucene-2.4 | **0.632** | 0.594 | 0.405 | 0.585 | 0.522 | 0.260 | 0.346 | 0.569 | 0.524 | 0.615 |
| poi-3.0 | 0.651 | **0.691** | 0.303 | 0.647 | 0.531 | 0.253 | 0.404 | 0.636 | 0.417 | 0.623 |
| synapse-1.2 | **0.750** | 0.673 | 0.320 | 0.601 | 0.581 | 0.358 | 0.629 | 0.569 | 0.622 | 0.645 |
| velocity-1.6 | 0.674 | 0.633 | 0.360 | 0.520 | 0.551 | 0.418 | **0.715** | 0.538 | 0.487 | 0.579 |
| Average | **0.690** | 0.652 | 0.320 | 0.639 | 0.605 | 0.189 | 0.464 | 0.527 | 0.570 | 0.654 |

**TABLE 14.** Cliff's delta of EGW+ versus baselines.

| | ManualDown | ManualUp | CPDP_IFS | HDP_KS | EMKCA | CTKCCA | WPDP | CCA+ | CLSUP |
|---|---|---|---|---|---|---|---|---|---|
| F1-score | 0.865(L) | 0.865(L) | 0.220(S) | 0.330(S) | 0.930(L) | 0.115(N) | 0.320(S) | 0.320(S) | 0.120(N) |
| AUC | 0.780(L) | 1.000(L) | 0.750(L) | 0.610(L) | 0.885(L) | 0.505(L) | 0.710(L) | 0.735(L) | 0.205(S) |
| G-mean | 0.460(M) | 1.000(L) | 0.535(L) | 0.750(L) | 1.000(L) | 0.550(L) | 0.970(L) | 0.745(L) | 0.410(M) |

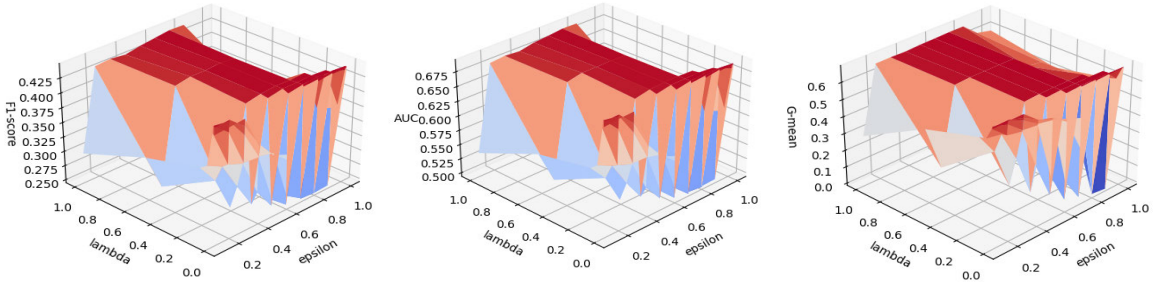
**TABLE 15.** Comparison results between EGW+ and EGW.

| Target | PD | | PF | | F1-score | | AUC | | G-mean | |
|---|---|---|---|---|---|---|---|---|---|---|
| | EGW+ | EGW | EGW+ | EGW | EGW+ | EGW | EGW | EGW | EGW+ | EGW |
| EQ | **0.598** | 0.506 | **0.353** | 0.369 | **0.391** | 0.337 | **0.679** | 0.568 | **0.618** | 0.563 |
| JDT | **0.620** | 0.478 | 0.341 | **0.235** | 0.540 | **0.586** | **0.688** | 0.621 | **0.632** | 0.597 |
| LC | **0.569** | 0.410 | 0.245 | **0.215** | **0.653** | 0.535 | **0.722** | 0.597 | **0.651** | 0.550 |
| ML | **0.709** | 0.612 | **0.202** | 0.284 | **0.776** | 0.575 | **0.801** | 0.664 | **0.750** | 0.660 |
| PDE | **0.660** | 0.526 | 0.302 | **0.300** | **0.584** | 0.506 | **0.730** | 0.613 | **0.674** | 0.605 |
| activemq-5.0.0 | 0.651 | **0.669** | **0.237** | 0.247 | 0.443 | **0.444** | **0.789** | 0.711 | 0.701 | **0.708** |
| derby-10.5.1.1 | 0.591 | **0.890** | **0.110** | 0.283 | **0.109** | 0.047 | **0.897** | 0.803 | 0.674 | **0.798** |
| groovy-0.94.0 | **0.742** | 0.570 | 0.325 | **0.310** | **0.279** | 0.230 | **0.784** | 0.630 | **0.702** | 0.627 |
| hbase-0.94.0 | **0.732** | 0.619 | 0.302 | **0.276** | **0.381** | 0.349 | **0.773** | 0.672 | **0.712** | 0.667 |
| hive-0.9.0 | **0.819** | 0.588 | **0.290** | 0.295 | **0.436** | 0.329 | **0.848** | 0.647 | **0.759** | 0.644 |
| KC1 | **0.792** | 0.728 | **0.233** | 0.292 | **0.519** | 0.454 | **0.844** | 0.718 | **0.778** | 0.717 |
| MC1 | 0.666 | **0.680** | **0.211** | 0.274 | **0.452** | 0.425 | **0.788** | 0.703 | **0.722** | 0.702 |
| PC1 | **0.733** | 0.555 | 0.305 | **0.297** | **0.295** | 0.251 | **0.760** | 0.629 | **0.711** | 0.622 |
| PC3 | **0.716** | 0.617 | **0.278** | 0.290 | **0.514** | 0.469 | **0.768** | 0.663 | **0.718** | 0.659 |
| PC4 | **0.661** | 0.626 | **0.223** | 0.265 | **0.518** | 0.495 | **0.760** | 0.681 | **0.715** | 0.676 |
| camel-1.6 | **0.646** | 0.539 | 0.422 | **0.248** | 0.566 | **0.574** | 0.607 | **0.646** | 0.609 | **0.630** |
| lucene-2.4 | **0.672** | 0.656 | **0.233** | 0.237 | 0.525 | **0.532** | **0.756** | 0.709 | **0.717** | 0.705 |
| poi-3.0 | 0.606 | **0.685** | 0.305 | **0.295** | 0.268 | **0.314** | 0.666 | **0.695** | 0.647 | **0.695** |
| synapse-1.2 | **0.583** | 0.515 | **0.260** | 0.289 | **0.356** | 0.314 | **0.713** | 0.613 | **0.653** | 0.603 |
| velocity-1.6 | **0.597** | 0.592 | **0.285** | 0.325 | **0.359** | 0.346 | **0.697** | 0.633 | **0.651** | 0.631 |
| Average | **0.668** | 0.603 | **0.273** | 0.281 | **0.448** | 0.406 | **0.754** | 0.661 | **0.690** | 0.653 |

3.8%, 37.0%, 5.1%, 8.5%, 50.1%, 22.6%, 16.3%, 12.0% and 3.6%.

Similarly, we also provided the analysis of CD diagrams and effect size. Figs 5-7 show that EGW+ performed better against other compared methods in F1-score, AUC and G-mean. In Table 14, compared with other methods, there are two negligible differences and twenty-five non-negligible differences, which indicated that EGW+ is different from most methods and the research results still make sense.

It is obviously that EGW+ performed better than other methods, which can prove the optimal transport can be used




(a) F1-score (b) AUC (c) G-mean

**FIGURE 9.** The variances of $\varepsilon$ and $\lambda$ with three indicators in EGW+.

**TABLE 16.** Results of five indicators with the best parameter $\varepsilon$ in EGW.

| Source | Target | ε | F1-score | AUC | G-mean |
|---|---|---|---|---|---|
| AEEEM | JIRA | 0.9 | 0.452 | 0.720 | 0.718 |
| | NASA | 0.5 | 0.283 | 0.693 | 0.689 |
| | PROMISE | 0.8 | 0.526 | 0.644 | 0.626 |
| JIRA | AEEEM | 0.4 | 0.441 | 0.692 | 0.687 |
| | NASA | 0.2 | 0.275 | 0.694 | 0.692 |
| | PROMISE | 0.4 | 0.500 | 0.608 | 0.593 |
| NASA | AEEEM | 0.9 | 0.364 | 0.590 | 0.582 |
| | JIRA | 0.9 | 0.352 | 0.597 | 0.594 |
| | PROMISE | 0.9 | 0.498 | 0.587 | 0.566 |
| PROMISE | AEEEM | 0.9 | 0.443 | 0.696 | 0.689 |
| | JIRA | 0.9 | 0.453 | 0.719 | 0.714 |
| | NASA | 0.8 | 0.282 | 0.690 | 0.685 |

to match the data distribution between two heterogeneous domains. Possible reasons that they can achieve better results are: 1) most of the baselines did not select all the metrics, but only used the common metrics shared by the source and target projects, which would limit HCPDP performance; 2) the pre-defined mapping functions of the compared methods may limit the performance.

### C. COMPARISON RESULTS BETWEEN EGW AND EGW+

Although we cannot compare EGW with EGW+ directly because they used different training data, we wanted to know the adaptation of the two algorithms. Table 15 lists the comparison results of EGW and EGW+ in PD, PF, F1-score, AUC and G-mean. Apparently, EGW+ can outperform better than EGW, which indicated that the label information of the training was useful to advance the classification performance.

As a whole, EGW was suitable when there was no defect data in target project, and EGW+ performed well after a few defects are labeled in target project.

## VI. DISCUSSION

### A. EFFECT OF PARAMETER SETTINGS

To explore the parameter variance for each dataset, we set $\varepsilon$ in the search place {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 10.0} and $\lambda$ in the search place {0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0}. Since each project is different, two algorithms cannot use the same parameters. Grid search was used to find the best parameters for EGW and EGW+ on each project. For example, we took JIRA as source project and AEEEM as target project, Fig.8 shows the variance of $\varepsilon$ with three indicators in EGW. Obviously, when $\varepsilon = 0.4$, F1-score, AUC and G-mean can achieve the best. Similarly, Fig. 9 shows the variances of $\varepsilon$ and $\lambda$ with three indicators in EGW+, and it can be found that the three indicators can obtain the best values when =0.5 and =0.001. By following the above process, Tables 16 and 17 list the best parameters of EGW and EGW+ on each project.

### B. TIME-EFFICIENCY

To illustrate whether EGW and EGW+ have the acceptable training time. Table 18 shows the average training time of our methods and other baselines on all datasets. Compared to CPDP_IFS, EMKCA, CCA+, WPDP, ManualDown and ManualUp, EGW and EGW+ have more training time because the transfer matrix is time-consuming. In particular, although WPDP, ManualDown and ManualUp ran faster, they required enough defective data of target project. Hence, they are not suitable for a new project. In conclusion, EGW and EGW+ are still acceptable under the premise of the resulting high forecast accuracy.

**TABLE 17.** Results of five indicators with the parameter $\varepsilon$ and $\lambda$ in EGW+.

| Source | Target | $\varepsilon$ | $\lambda$ | F1-score | AUC | G-mean |
|---|---|---|---|---|---|---|
| AEEEM | JIRA | 0.3 | 0.001 | 0.458 | 0.782 | 0.727 |
| | NASA | 0.1 | 0.0001 | 0.322 | 0.824 | 0.729 |
| | PROMISE | 0.4 | 0.005 | 0.589 | 0.724 | 0.665 |
| JIRA | AEEEM | 0.5 | 0.001 | 0.429 | 0.705 | 0.668 |
| | NASA | 0.3 | 0.0001 | 0.324 | 0.829 | 0.742 |
| | PROMISE | 0.4 | 0.001 | 0.585 | 0.718 | 0.663 |
| NASA | AEEEM | 0.1 | 0.005 | 0.398 | 0.674 | 0.646 |
| | JIRA | 0.3 | 0.001 | 0.462 | 0.797 | 0.739 |
| | PROMISE | 0.4 | 0.005 | 0.592 | 0.730 | 0.666 |
| PROMISE | AEEEM | 0.4 | 0.005 | 0.417 | 0.685 | 0.653 |
| | JIRA | 0.1 | 0.001 | 0.458 | 0.774 | 0.721 |
| | NASA | 0.2 | 0.0005 | 0.344 | 0.802 | 0.658 |

**TABLE 18.** Average running time of each method on all datasets.

| | EGW | EGW+ | CLSUP | CPDP_IFS | CTKCCA | EMKCA | HDP_KS | CCA+ | WPDP | ManualDown | ManualUp |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Time(s) | 3.061 | 3.097 | 3.262, | 2.730 | 3.248 | 31.991 | 2.028 | 260.230 | 0.230 | 1.000 | 0.068 |

### C. THREATS TO VALIDITY

Several potential threats to the validity are described in the followings.

1) Bias of comparison. Most of the compared works do not provide the program codes of their methods. We implemented their methods by following their papers. In addition, many methods have been proposed to solve the HCPDP problems. But EGW and EGW+ were designed from data distribution perspective, we chose only some representative unsupervised and semi-supervised methods for comparison.
2) Bias of evaluation measures and settings. In this work, the widely used measures of PD, PF, F1-score, AUC and G-mean were selected to evaluate the results. Other measures, such as matthews correlation coefficient is left for future work.
3) Bias of classifier. Classification is a large research topic and many learning methods can be used to construct the classifiers. As investigated in previous studies, Logistic regression has been used widely. Therefore, this work also applied Logistic regression to build the classifiers.
4) Bias of datasets. There are several benchmark datasets used in cross-project defect prediction, such as Relink, and SOFTLAB, and some datasets contain different versions of each project such as tomcat of PROMISE, jruby of JIRA. More datasets will be used in the future.

## VII. CONCLUSION

For HCPDP task, the source and target data have different features, which impelled researchers considered applying transfer learning to solve HCPDP problems. Most previous works studied how to learn feature mapping functions in order to put the source and target data in a common space. However, these methods proposed the pre-defined mapping functions, which often affect the effect of feature transformation.

To avoid learning specific mapping function for HCPDP model, optimal transport theory can implement how to transfer data instances from one distribution to another by calculating transmission cost. We designed an algorithm based the entropic Gromove-Wassertein (EGW) distance to learn the data transmission scheme between two different metric spaces. The EGW model considers the transmission cost between metric spaces without directly measuring the distance between two data instances from different feature spaces. Then, by combining the label information of the target data, an EGW+ algorithm was designed to improve EGW. The experimental results showed that the HCPDP modes based on EGW and EGW+ could achieve competitive predictive performance.

For the future work, more datasets and baselines will be used to verify the proposed approach. In addition, we will extend the optimal theory to improve HCPDP performance by introducing the unlabeled target data.

## REFERENCES

[1] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A systematic literature review on fault prediction performance in software engineering," *IEEE Trans. Softw. Eng.*, vol. 38, no. 6, pp. 1276–1304, Nov. 2012.
[2] T. Zimmermann and N. Nagappan, "Predicting defects using network analysis on dependency graphs," in *Proc. 13th Int. Conf. Softw. Eng. (ICSE)*, Jan. 2008, pp. 531–540.

[3] T. Menzies, A. Dekhtyar, J. Distefano, and J. Greenwald, "Data mining static code attributes to learn defect predictors," *IEEE T. Softw. Eng.*, vol. 33, no. 9, pp. 637–640, 2007.

[4] S. Zheng, J. Gai, H. Yu, H. Zou, and S. Gao, "Software defect prediction based on fuzzy weighted extreme learning machine with relative density information," *Sci. Program.*, vol. 2020, pp. 1–18, Nov. 2020.

[5] S. Hosseini, B. Turhan, and D. Gunarathna, "A systematic literature review and meta-analysis on cross project defect prediction," *IEEE Trans. Softw. Eng.*, vol. 45, no. 2, pp. 111–147, Feb. 2019.

[6] Y. Liu, T. M. Khoshgoftaar, and N. Seliya, "Evolutionary optimization of software quality modeling with multiple repositories," *IEEE Trans. Softw. Eng.*, vol. 36, no. 6, pp. 852–864, Nov. 2010.

[7] F. Peters, T. Menzies, and A. Marcus, "Better cross company defect prediction," in *Proc. 10th Work. Conf. Mining Softw. Repositories (MSR)*, May 2013, pp. 409–418.

[8] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, "Cross-project defect prediction: A large scale experiment on data vs. domain vs. process," in *Proc. 7th Joint Meeting Eur. Softw. Eng. Conf. ACM SIGSOFT Symp. Found. Softw. Eng.*, Aug. 2009, pp. 91–100.

[9] J. Nam and S. Kim, "Heterogeneous defect prediction," in *Proc. 10th Joint Meeting Found. Softw. Eng.*, Aug. 2015, pp. 508–519.

[10] J. Nam and S. Kim, "CLAMI: Defect prediction on unlabeled datasets (T)," in *Proc. 30th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Nov. 2015, pp. 452–463.

[11] C. Liu, D. Yang, X. Xia, M. Yan, and X. Zhang, "A two-phase transfer learning model for cross-project defect prediction," *Inf. Softw. Technol.*, vol. 107, pp. 125–136, Mar. 2019.

[12] Q. Yu, S. Jiang, and Y. Zhang, "A feature matching and transfer approach for cross-company defect prediction," *J. Syst. Softw.*, vol. 132, pp. 366–378, Oct. 2017.

[13] Z. Q. Li, X. Y. Jing, X. K. Zhu, H. Y. Zhang, B. W. Xu, and S. Ying, "Heterogeneous defect prediction with two-stage ensemble learning," *Automated Softw. Eng.*, vol. 26, no. 3, pp. 599–651, Jun. 2019.

[14] H. Chen, X.-Y. Jing, Z. Li, D. Wu, Y. Peng, and Z. Huang, "An empirical study on heterogeneous defect prediction approaches," *IEEE Trans. Softw. Eng.*, vol. 47, no. 12, pp. 2803–2822, Dec. 2021.

[15] X. Chen, Y. Mu, K. Liu, Z. Cui, and C. Ni, "Revisiting heterogeneous defect prediction methods: How far are we?" *Inf. Softw. Technol.*, vol. 130, Feb. 2021, Art. no. 106441.

[16] S. J. Pan and Y. Qiang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[17] J. Lott and C. Villani, "Ricci curvature for metric-measure spaces via optimal transport," *Ann. Math.*, vol. 169, no. 3, pp. 903–991, May 2009.

[18] M. Perrot, N. Courty, R. Flamary, and A. Habrard, "Mapping estimation for discrete optimal transport," in *Proc. Adv. Neural Inform. Process. Sys. 29 (NIPS)*, 2016, pp. 4197–4205.

[19] N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy, "Optimal transport for domain adaptation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 9, pp. 1853–1865, Sep. 2017.

[20] G. Peyre, M. Cuturi, and J. Solomon, "Gromov–Wasserstein averaging of kernel and distance matrices," in *Proc. 33rd Int. Conf. Int. Conf. Mach. Learn. (ICML)*, vol. 2016, pp. 2664–2672.

[21] J. Nam, S. J. Pan, and S. Kim, "Transfer defect learning," in *Proc. Int. Conf. Softw. Eng.*, May 2013, pp. 382–391.

[22] C. Liu, D. Yang, X. Xia, M. Yan, and X. Zhang, "A two-phase transfer learning model for cross-project defect prediction," *Inf. Softw. Technol.*, vol. 107, pp. 125–136, Mar. 2019.

[23] Z. Li, X. Y. Jing, and X. Zhu, "Heterogeneous fault prediction with cost-sensitive domain adaptation: heterogeneous fault prediction with cost sensitive domain adaptation," *Softw. Test. Verif. Rel.*, vol. 28, no. 1, 2018, Art. no. e1658.

[24] X. Jing, F. Wu, X. Dong, F. Qi, and B. Xu, "Heterogeneous cross-company defect prediction by unified metric representation and CCA-based transfer learning," in *Proc. 10th Joint Meeting Found. Softw. Eng.*, Aug. 2015, pp. 496–507.

[25] P. He, B. Li, and Y. T. Ma, "Towards cross-project defect prediction with imbalanced feature sets," 2014, *arXiv:1411.4228*.

[26] M. Cheng, G. Wu, M. Jiang, H. Wan, G. You, and M. Yuan, "Heterogeneous defect prediction via exploiting correlation subspace," in *Proc. Int. Conf. Softw. Eng. Knowl. Eng.*, Jul. 2016, pp. 171–176.

[27] Y. Zhou, Y. Yang, H. Lu, L. Chen, Y. Li, Y. Zhao, J. Qian, and B. Xu, "How far we have progressed in the journey? An examination of cross-project defect prediction," *ACM Trans. Softw. Eng. Methodol.*, vol. 27, no. 1, pp. 1–51, Jan. 2018.

[28] F. Zhang, Q. Zheng, Y. Zou, and A. E. Hassan, "Cross-project defect prediction using a connectivity-based unsupervised classifier," in *Proc. 38th Int. Conf. Softw. Eng.*, May 2016, pp. 309–320.

[29] H. Chen, X.-Y. Jing, Z. Li, D. Wu, Y. Peng, and Z. Huang, "An empirical study on heterogeneous defect prediction approaches," *IEEE Trans. Softw. Eng.*, vol. 47, no. 12, pp. 2803–2822, Dec. 2021.

[30] K. Jiang, Y. Zhang, H. Wu, A. Wang, and Y. Iwahori, "Heterogeneous defect prediction based on transfer learning to handle extreme imbalance," *Appl. Sci.*, vol. 10, no. 1, p. 396, Jan. 2020.

[31] J. R. Hershey and P. A. Olsen, "Approximating the Kullback Leibler divergence between Gaussian mixture models," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2007, pp. 317–320.

[32] J.-D. Benamou, G. Carlier, M. Cuturi, L. Nenna, and G. Peyré, "Iterative Bregman projections for regularized transportation problems," *SIAM J. Sci. Comput.*, vol. 37, no. 2, pp. A1111–A1138, Jan. 2015.

[33] M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transportation distances," in *Proc. Int. Conf. Neural Inform. Process. Sys (NIPS)*, 2013, pp. 2292–2300.

[34] Y. Yan, W. Li, H. Wu, H. Min, M. Tan, and Q. Wu, "Semi-supervised optimal transport for heterogeneous domain adaptation," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 1–7.

[35] W. Li, L. Duan, D. Xu, and I. W. Tsang, "Learning with augmented features for supervised and semi-supervised heterogeneous domain adaptation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 6, pp. 1134–1148, Jun. 2014.

[36] M. D'Ambros, M. Lanza, and R. Robbes, "Evaluating defect prediction approaches: A benchmark and an extensive comparison," *Empir. Softw. Eng.*, vol. 17, nos. 4–5, pp. 531–577, 2011.

[37] S. Yatish, J. Jiarpakdee, P. Thongtanunam, and C. Tantithamthavorn, "Mining software defects: Should we consider affected releases?" in *Proc. IEEE/ACM 41st Int. Conf. Softw. Eng. (ICSE)*, May 2019, pp. 654–665.

[38] M. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data quality: Some comments on the NASA software defect datasets," *IEEE Trans. Softw. Eng.*, vol. 39, no. 9, pp. 1208–1215, Sep. 2013.

[39] M. Jureczko and L. Madeyski, "Towards identifying software project clusters with regard to defect prediction," in *Proc. 6th Int. Conf. Predictive Models Softw. Eng.*, Sep. 2010, pp. 1–10.

[40] F. Peters, T. Menzies, L. Gong, and H. Zhang, "Balancing privacy and utility in cross-company defect prediction," *IEEE Trans. Softw. Eng.*, vol. 39, no. 8, pp. 1054–1068, Aug. 2013.

[41] F. Zhang, A. Mockus, I. Keivanloo, and Y. Zou, "Towards building a universal defect prediction model with rank transformed predictors," in *Proc. Work. Conf. Mining Softw. Repositories (MSR)*, 2014, pp. 1–39.

[42] F. Peters, T. Menzies, and L. Layman, "LACE2: Better privacy-preserving data sharing for cross project defect prediction," in *Proc. IEEE/ACM 37th IEEE Int. Conf. Softw. Eng.*, May 2015, pp. 801–811.

[43] 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 Softw. Eng.*, vol. 14, no. 5, pp. 540–578, 2009.

[44] D. Ryu, O. Choi, and J. Baik, "Value-cognitive boosting with a support vector machine for cross-project defect prediction," *Empirical Softw. Eng.*, vol. 21, no. 1, pp. 43–71, 2016.

[45] L. Chen, B. Fang, Z. Shang, and Y. Tang, "Negative samples reduction in cross-company software defects prediction," *Inf. Softw. Technol.*, vol. 62, pp. 67–77, Jun. 2015.

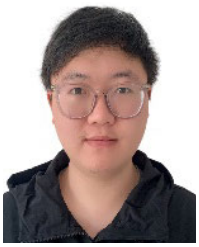
[46] X. Chen, Y. Mu, Y. Qu, C. Ni, M. Liu, T. He, and S. Liu, "Do different cross-project defect prediction methods identify the same defective modules?" *J. Softw., Evol. Process*, vol. 32, no. 5, May 2020, Art. no. e2234.

[47] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognit.*, vol. 30, no. 7, pp. 1145–1159, 1997.

[48] C. Tantithamthavorn, A. E. Hassan, and K. Matsumoto, "The impact of class rebalancing techniques on the performance and interpretation of defect prediction models," *IEEE Trans. Softw. Eng.*, vol. 46, no. 11, pp. 1200–1219, Nov. 2020.

[49] Z. Li, X.-Y. Jing, F. Wu, X. Zhu, B. Xu, and S. Ying, "Cost-sensitive transfer kernel canonical correlation analysis for heterogeneous defect prediction," *Automated Softw. Eng.*, vol. 25, no. 2, pp. 201–245, Jun. 2018.

[50] H. Tong, B. Liu, and S. Wang, "Kernel spectral embedding transfer ensemble for heterogeneous defect prediction," *IEEE Trans. Softw. Eng.*, vol. 47, no. 9, pp. 1886–1906, Sep. 2021.
[51] Y. Jiang, B. Cukic, and Y. Ma, "Techniques for evaluating fault prediction models," *Empirical Softw. Eng.*, vol. 13, no. 5, pp. 561–595, 2008.
[52] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Dec. 2006.
[53] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Jun. 2008.
[54] Z. Li, X.-Y. Jing, X. Zhu, and H. Zhang, "Heterogeneous defect prediction through multiple kernel learning and ensemble learning," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Sep. 2017, pp. 91–102.
[55] C. Catal and B. Diri, "A systematic review of software fault prediction studies," *Expert Syst. Appl.*, vol. 36, no. 4, pp. 7346–7354, May 2009.
[56] S. Zheng, J. Gai, H. Yu, H. Zou, and S. Gao, "Training data selection for imbalanced cross-project defect prediction," *Comput. Electr. Eng.*, vol. 94, Sep. 2021, Art. no. 107370.
[57] A. E. C. Cruz and K. Ochimizu, "Towards logistic regression models for predicting fault-prone code across software projects," in *Proc. 3rd Int. Symp. Empirical Softw. Eng. Meas.*, Oct. 2009, pp. 460–463.
[58] Q. Huang, E. Shihab, X. Xia, D. Lo, and S. Li, "Identifying self-admitted technical debt in open source projects using text mining," *Empirical Softw. Eng.*, vol. 23, no. 1, pp. 418–451, 2018.
[59] J. Ren, C. Peng, S. Zheng, H. Zou, and S. Gao, "An approach to improving homogeneous cross-project defect prediction by Jensen–Shannon divergence and relative density," *Sci. Program.*, vol. 2022, pp. 1–16, Oct. 2022.



**XING ZONG** received the B.S. degree in electronic information engineering from the Southwest University of Science and Technology, Sichuan, China. He is currently pursuing the M.S. degree with the Jiangsu University of Science and Technology. His research interests include intelligent software engineering and data mining.

**GUIYU LI** received the B.S. degree in computer science and technology from the Sichuan University of Science and Engineering. She is currently pursuing the M.S. degree with the Jiangsu University of Science and Technology. Her research interests include intelligent software engineering and data mining.

**SHANG ZHENG** received the B.S. degree from the College of Computer Science and Technology, Northeast Normal University, Changchun, China, in 2006, the M.S. degree from the College of Computer Science and Technology, Jilin University, Changchun, in 2009, and the Ph.D. degree in computer science from De Montfort University, Leicester, U.K., in 2014.

Since 2015, he has been a Lecturer with the School of Computer Science and Engineering, Jiangsu University of Science and Technology, Zhenjiang, China. He has authored or coauthored more than ten top journal and conference papers. His research interests include software repository mining, software maintenance, software evolution, and intelligent computation.

**HAITAO ZOU** received the B.S. degree in information security from Nankai University, Tianjin, China, in 2007, and the M.S. and Ph.D. degrees in software engineering from the University of Macau, Macau, China, in 2010 and 2015, respectively.

He is currently a Lecturer with the School of Computer, Jiangsu University of Science and Technology, Zhenjiang, China. He has authored and coauthored several top journal articles. His research interests include databases, web information retrieval, and web mining. He is a reviewer of a couple of web mining conferences.

**HUALONG YU** was born in Harbin, China, in 1982. He received the B.S. degree in computer science from Heilongjiang University, Harbin, in 2005, and the M.S. and Ph.D. degrees in computer science from Harbin Engineering University, Harbin, in 2008 and 2010, respectively.

From 2013 to 2017, he was a Postdoctoral Fellow at the School of Automation, Southeast University, Nanjing, China. From 2017 to 2018, he was a Senior Visiting Scholar at the Faculty of Information Technology, Monash University, Melbourne, Australia. Since 2020, he has been a Professor at the School of Computer Science and Engineering, Jiangsu University of Science and Technology, Zhenjiang, China. He has authored or coauthored more than 90 journals and conference papers, and four monographs. His research interests include machine learning, data mining, and bioinformatics.

Dr. Yu is a member in the organizing committee of several international conferences. He is also a member of ACM, the China Computer Federation (CCF), and the Youth Committee of the Chinese Association of Automation (CAA). He is an active Reviewer of more than 20 high-quality international journals, including IEEE Transactions on Neural Networks and Learning Systems, IEEE Transactions on Cybernetics, IEEE Transactions on Knowledge and Data Engineering, IEEE Transactions on Computational Biology and Bioinformatics, and *ACM Transactions on Knowledge Discovery from Data* (TKDD). He is an Associate Editor of IEEE Access.

**SHANG GAO** received the B.S. degree in system engineering and the M.S. degree in military equipment from Air Force Engineering University, Xi'an, China, in 1993 and 1996, respectively, and the Ph.D. degree in pattern recognition from the Nanjing University of Science and Technology, Nanjing, China, in 2006. Since 2009, he has been a Professor at the School of Computer, Jiangsu University of Science and Technology, Zhenjiang, China, where he has been the Dean, since 2017. He has published more than 80 research papers on the professional journals and conferences. His research interests include optimization theory, swarm intelligence, and machine learning.