Hybrid Machine Learning-Based CPU Performance Prediction for Heterogeneous Scheduling Using LASSO, SVR, and Metaheuristic Optimization

Chunxia Liu*, Yuquan Zhou
Guangxi Technological College of Machinery and Electricity, Nanning 530007, China
E-mail: tremere1010@163.com
*Corresponding author

Keywords: CPU performance, support vector regression, lasso regression, walrus optimization algorithm, crocodile hunting strategy

The increasing prevalence of heterogeneous computing systems necessitates the development of advanced CPU scheduling strategies capable of leveraging diverse computational resources effectively. Accurate performance prediction of applications across heterogeneous platforms is a key enabler of efficient scheduling. In this study, machine learning (ML) models—Lasso Regression (LAS) and Support Vector Regression (SVR)—were employed to predict CPU performance. To enhance their accuracy, two metaheuristic optimization algorithms, the Crocodiles Hunting Strategy (CHS) and the Walrus Optimization Algorithm (WOA), were integrated with the base models, resulting in four hybrid schemes: SVCH (SVR + CHS), SVWO (SVR + WOA), LACH (LAS + CHS), and LAWO (LAS + WOA). The models were trained and validated on a dataset comprising 205 samples from published papers. Model performance was evaluated using five metrics: Root Mean Square Error (RMSE), Coefficient of Determination (R^2), Mean Squared Error (MSE), Symmetric Mean Absolute Percentage Error (SMAPE), and Coefficient of Variation of RMSE (CV_RMSE). Among all models, the LACH scheme achieved the best validation performance with an R^2 of 0.993, RMSE of 8.778, and MSE of 77.1, significantly outperforming the base SVR model, which achieved an R^2 of only 0.915 and RMSE of 32.785. Compared to existing ML-based CPU prediction approaches, the proposed hybrid models—particularly LACH and LAWO—demonstrate superior accuracy and robustness, offering valuable potential for enhancing scheduling decisions in heterogeneous systems.

Povzetek: Analizirana je napoved CPU-zmogljivosti v heterogenih sistemih v zvezi z nelinearnimi delovnimi obremenitvami in s slabimi izbirami hiperparametrov. Predlagani so hibridni modeli (LAS/SVR + CHS/WOA), kjer metahevristiki optimizirata LASSO in SVR.

1 Introduction

These days, technology has advanced to a point where use cases and applications are more complicated and denser than ever before. Improving the efficiency of complex systems is a perennial challenge at the center of most research in contemporary computing. Heterogeneous clusters based on multicore CPUs and high-performing GPUs have revolutionized current-performance computing for demanding and complicated applications [1]. To meet the demands of contemporary necessity, these high-performance systems must be continuously developed. Determining the linkages and underlying interdependence in hybrid systems requires intelligent analysis and application execution monitoring. This supports the goal of improving this heterogeneous system's efficiency [2] to make the most use of the resources at hand. Numerous uses for artificial intelligence (AI) and ML have emerged as a result of advancements in these fields. Making use of these technologies, the ability to precisely evaluate the cluster using various data types. Estimated from the past data used to train the model, regression schemes [3] predict a cycle's value. Among the most accurate ML techniques are regression schemes and numerical prediction schemes when compared to classical categorization procedures [4], [5].

Fink used the linear regression model in [6] to make predictions for 3 different planning methods. 3 planning methods were created by Fink based on different issue sizes. The algorithm that is performed for the longest duration is chosen based on these 3 predictions [7]. In the study, a prediction model for CPU-GPU heterogeneous clusters was developed by considering previous data. Qilin et al. created a method that uses the offline regression model to provide runtime prediction [1]. After doing extensive profiling with various input factors, a linear model is developed. This approach works well with the map-reduce programming paradigm, although it is not appropriate for many applications. In this study, the calculation is done entirely on GPUs rather than a CPU to get better speed [8], [9].

Boyer et al.'s work [10] mainly focused on teaching the scheduler to choose adequate hardware resources according to job size. After dividing the incoming data into manageable portions, the author explains how to run

a data block on the designated processors. Boyer even brought up external performance-influencing factors like CPU load and clock rate scaling [11], [12]. The scheduler assigns the remaining chunk's data to the appropriate processing units in a linear fashion, drawing on the operation period of the preceding chunk. The author of this research did not create a unique analytical prediction model, nor did they employ heterogeneous systems with CPUs and GPUs and a hybrid programming paradigm. However, a portion of the application was also considered during the implementation of the prediction model on a heterogeneous system of CPUs and GPUs [13].

Wang et al.'s attempt [14] offered a straightforward approach that splits tasks into 2 categories, such as computation and communication. Some operating nodes must handle data flow computation activities, while communication responsibilities are handled by other transmitting nodes. The processor tasks that need to be mapped are determined by an algorithm that uses a combination of task portioning and runtime task execution. This is an easy, intricate process, yet the writer contrasted it with conventional techniques and obtained a performance increase of 23%. In this work, the author is not regarded as a resource prediction model and heterogeneous cluster application.

A runtime scheduling mechanism for the IBM processor was proposed by author Pieter in [15], which aids in dividing up workloads among several processors. The author of this study took into consideration a heterogeneous architecture system in which the CPU is used to operate the GPU, and the GPUs are used to compute difficult calculation sections. Even the author considers a pool of processors to schedule work for any processor and employs a heterogeneous architecture for practical implementation. FF (first free), SJF, RR, and $FCFS$ are examples of common scheduling algorithms that the author originally utilized because of their effectiveness and simplicity. The $CPU + GPU$ cluster scheduling approach was not given an analytical prediction model by the study's author [16]. The successful utilization of increasing transistor densities has been made feasible by heterogeneous computing resources, which mix incredibly fast and powerful cores with cores that use less energy, integrated GPUs, and other accelerators. The industry's interest in heterogeneous processors has lately led to several useful implementations, like ARM's large, by TINY [17]. However, to fully take use of and maximize the advantages that heterogeneous architectures provide, multi-program and parallel applications must be properly controlled by a CPU scheduler [18]. Table 1 shows the literature review for the CPU performance prediction.

Table 1: Literature review for the CPU performance prediction.

Study	Objective	Results
Liu [19]	Create a standardized dataset for CPU performance prediction. Develop a deep learning model for accurate predictions.	PerfCastDB dataset enhances CPU performance prediction tasks. The NCPP model shows superior performance over traditional approaches.
Daraghmeh et al. [20]	Improve CPU usage prediction in cloud data centers. Enhance resilience to cloud environment volatility.	Improved CPU usage prediction accuracy in cloud data centers. Enhanced resilience to cloud environment volatility.
Wei-Wei et al. [21]	Improve the accuracy and rationality of processor performance predictions. Enhance model interpretability and result traceability.	The proposed method generates more reliable and accurate processor performance predictions. The method is superior to traditional models in terms of prediction accuracy.
Foots et al. [22]	Classify vendors based on estimated CPU performance. Predict CPU performance using hardware components.	Neural networks predict CPU performance accurately with low error. Cubic regression outperforms neural networks with more training data.

The rapid advancement of heterogeneous computing environments, where CPUs with varying architectural designs are integrated, poses significant challenges for effective task scheduling. These systems demand intelligent prediction mechanisms to estimate application performance accurately across diverse resources, enabling optimal CPU scheduling and improved system throughput. Traditional models used for this purpose, such as standalone Support Vector Regression (SVR) and Lasso Regression (LAS), often lack the adaptability required to capture the complex and non-linear characteristics of dynamic workloads. Furthermore, without sophisticated parameter tuning or adaptive learning, their predictive performance may plateau, limiting their practical deployment in real-time systems. To overcome these limitations, this study proposes a hybrid machine learning framework that enhances the predictive capacity of SVR and LAS models by integrating two recent metaheuristic optimization algorithms: the Crocodiles Hunting Strategy (CHS) and the Walrus Optimization Algorithm (WOA). These algorithms are designed to perform global optimization of model hyperparameters, which plays a crucial role in improving prediction accuracy, reducing overfitting, and enhancing generalization on unseen data. Accordingly, the objective of this research is fourfold. First, to construct

baseline predictive models using SVR and LAS for estimating CPU workload performance. Second, to develop hybrid versions of these models, SVR + CHS (SVCH), SVR + WOA (SVWO), LAS + CHS (LACH), and LAS + WOA (LAWO), by integrating the optimizers into the training pipeline. Third, to rigorously evaluate and compare the performance of all models using standard metrics, including Root Mean Square Error (RMSE), Mean Squared Error (MSE), Coefficient of Determination (R^2), Symmetric Mean Absolute Percentage Error (SMAPE), and Coefficient of Variation of RMSE (CV_RMSE). Finally, the study aims to analyze the practical implications of the hybrid models in real-world CPU scheduling tasks by examining their predictive power, computational cost, and potential for real-time implementation. Through this hybridization approach, the study seeks to demonstrate measurable improvements in prediction accuracy while offering models that are robust, generalizable, and interpretable, especially important for resource-aware decision-making in complex, heterogeneous computing systems. SVR and LAS were selected as the foundational models in this study due to their complementary strengths in handling high-dimensional and complex datasets. SVR is well-known for its robustness in modeling non-linear relationships through kernel functions, making it effective in capturing intricate patterns in data without overfitting. Its ability to generalize well in small to medium-sized datasets makes it suitable for the present CPU performance prediction task. On the other hand, LAS offers a robust feature selection mechanism by enforcing sparsity in the model coefficients through L1 regularization. This helps reduce model complexity and improves interpretability, particularly when dealing with datasets that may contain redundant or irrelevant features.

To guide the study, the following research questions are posed:

RQ1: Can the integration of metaheuristic optimization algorithms (CHS and WOA) significantly improve the predictive accuracy of ML models for CPU workload estimation?

RQ2: How does each hybrid model (SVCH, SVWO, LAWO, LACH) compare to its standalone counterpart in terms of RMSE and R^2?

RQ3: What are the trade-offs between computational cost and prediction accuracy in hybrid vs. non-hybrid models?

RQ4: Does the LACH model offer meaningful interpretability advantages due to Lasso's inherent feature selection properties?

RQ5: How can accurate CPU performance prediction models contribute to improved system throughput and resource utilization in heterogeneous scheduling scenarios?

Based on the objectives and prior work, the following hypotheses are proposed:

H1: Hybrid ML models optimized with CHS or WOA will outperform standalone SVR and LAS models in terms of RMSE, R^2, and MSE.

H2: The LACH model will yield the highest prediction accuracy among all proposed schemes due to the synergistic effect of LAS's regularization and CHS's adaptive exploration.

H3: Lasso-based hybrid models will provide better interpretability than SVR-based models due to the sparsity and feature reduction properties of Lasso Regression.

H4: While hybrid models incur higher training-time computational costs, they will generalize better on unseen workloads, making them suitable for deployment in real-time heterogeneous CPU schedulers.

2 Mathematical schemes

2.1 Support vector regression (SVR)

The SVM was first developed by Vapnik VN to tackle categorization issues [23]. Nevertheless, it was later changed to handle regression issues as well. SVM regression is based on the structural risk minimization (SRM) idea, which is far more enhanced than the conventional empirical risk minimization (ERM) methodology [24]. Because it attempts to reduce the upper bound of generalization error, a critical component of the learning process, the SRM principle is essential to statistical learning. An expansion of SVM created especially for regression problems is called SVR [25]. While SVR and SVM use comparable methods, their purposes for parameter estimation are distinct. The usage of slack variables is where the 2 differ most from one another. For classification tasks, SVMs provide a single slack variable to each training data point; for regression tasks, SVR assigns 2 slack variables to each training data point. One unique feature of SVR is that, in contrast to other schemes, it can improve generalization performance and hasten the attainment of the best global resolution [26].

2.1.1 Linear SVR

The local linear regression version of SVR maybe described by Eq. (1) given a training database $\{y_i, x_i, i = 1, 2, 3 \ ... \ n\}$, where y_i signifies the output vector, x_i displays the input vector, and n indicates the size of the database.

$$f(x,k) = k \times x + b \tag{1}$$

The given equation illustrates the dot product as (x,k), where k symbolizes the weight vector, x displays the normalized test pattern, and b denotes the bias. SRM theory aims to minimize the empirical risk, as represented by $R_{emp}\ (k,b)$. Eq. (3) illustrates how an ε-insensitive loss function, known as $L_\varepsilon(y_i, f(x_i,k))$, is used to calculate the empirical risk.

$$R_{emp}(k,b) = \frac{1}{n}\sum_{i=1}^{n} L_\varepsilon\left(y_i, f(x_i,k)\right) \tag{2}$$

$$L_\varepsilon\big(y_i, f(x_i,k)\big) = \begin{cases} \varepsilon, & if \quad |y_i - f(x_i,k)| \le \varepsilon \\ |y_i - f(x_i,k)| - \varepsilon, & otherwise \end{cases} \tag{3}$$

During optimization, the loss function $L(\varepsilon)$, where x_i is the training pattern, gauges the tolerance error between the objective output (y_i) and the estimated output values $f(x_i; w)$. By reducing $||k||^2$, the ε-insensitive loss function may be utilized to resolve SVR model complexity in linear regression problems. Furthermore, it is feasible to estimate the training data deviation outside of the ε-zone by introducing a non-negative slack variable, φ_i.

$$\underset{k,b,\varphi,\varphi^*}{\text{Lim}}\left[\frac{1}{2}k.k + c\left(\sum_{i=1}^{n}\varphi_i^* + \sum_{1=1}^{n}\varphi_i\right)\right]$$
$$\text{Subjected to,}\begin{cases} y_i - k.x_i - b \le \varepsilon + \varphi_i^* \\ k.x_i + b - y_i \le \varepsilon + \varphi_i \ , i \\ \varphi_i^*, \varphi_i \ge 0 \end{cases} \quad (4)$$
$$= 1, \ldots, n$$

Finding the Lagrange function's saddle point is pivotal to resolving the earlier issue.

$$L(k, \varphi^*, \varphi, \alpha^*, \alpha, c, \gamma^*, \gamma = \frac{1}{2}k.k + c\left(\sum_{i=1}^{n}\varphi_i^* + \sum_{1=1}^{n}\varphi_i\right) - \sum_{i=1}^{n}\alpha_i[y_i - k.x_i - b + \varepsilon + \varphi_i\ ] \quad (5)$$
$$- \sum_{i=1}^{n}\alpha_i^*[k.x_i + b - y_i + \varepsilon + \varphi_i^*\ ] - \sum_{1}^{n}(\gamma_i^*\varphi_i^* + \gamma_i\varphi_i)$$

By using the KKT conditions, the Lagrange function may be reduced. This requires partial differentiation of Eq. (5) concerning k, b, φ_i^*, and φ_i.

$$\frac{\delta \mathrm{L}}{\delta \mathrm{k}} = \mathrm{k} + \sum_{i=1}^{n}\alpha_i x_i \quad (6)$$
$$-\sum_{i=1}^{n}\alpha_i^* x_i = 0, k = \sum_{i=1}^{n}(\alpha_i^* - \alpha_i)x_i$$

$$\frac{\delta \mathrm{L}}{\delta \mathrm{b}} = \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\alpha_i^* = 0, \sum_{i=1}^{n}\alpha_i = \sum_{i=1}^{n}\alpha_i^* \quad (7)$$

$$\frac{\delta \mathrm{L}}{\delta \varphi^*} = \mathrm{c} - \sum_{i=1}^{n}\gamma_i^* - \sum_{i=1}^{n}\alpha_i^* = 0, \sum_{i=1}^{n}\gamma_i^* = c - \sum_{i=1}^{n}\alpha_i^* \quad (8)$$

$$\frac{\delta \mathrm{L}}{\delta \varphi} = \mathrm{c} - \sum_{i=1}^{n}\gamma_i - \sum_{i=1}^{n}\alpha_i = 0, \sum_{i=1}^{n}\gamma_i = c - \sum_{i=1}^{n}\alpha_i \quad (9)$$

Eqs. (6) and (1) have a relationship with the parameter k. The dual optimization function is stated as follows when Eq. (6) is substituted into the Lagrange function of Eq. (5):

$$max_{\alpha,\alpha^*}[k(\alpha, \alpha^*)] = max_{\alpha,\alpha^*}\left[\sum_{i=1}^{n}\gamma_i(\alpha_i^* - \alpha_i) - \varepsilon\sum_{i=1}^{n}(\alpha_i^* - \alpha_i) - \frac{1}{2}\sum_{ij=1}^{n}(\alpha_i^* - \alpha_i)(\alpha_i^* - \alpha_i)(x_i.x_j)\right] \quad (10)$$
$$subjected\ to\begin{cases} \sum_{i=1}^{n}(\alpha_i^* - \alpha_i) = 0 \\ 0 \le \alpha_i^*, \alpha_i \le 0 \end{cases} i$$
$$= 1, \ldots, n$$

The optimization problem is defined by utilizing the Lagrange multipliers α_i^* and α_i [27]. Upon solving Eq. (10) concerning the restrictions in Eq. (11), the final linear regression function is displayed as:

$$f(x, \alpha^*, \alpha) = \sum_{i=1}^{n}(\alpha_i^* - \alpha_i)(x_i, x) + b \quad (11)$$

Fig. 1 presents the SVR's flowchart.

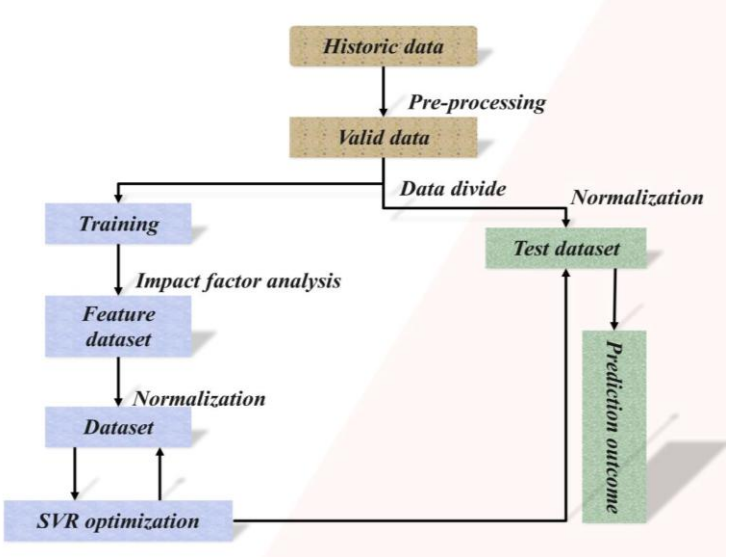


Figure 1: The SVR's flowchart

2.2 Lasso regression (LAS)

LAS aims to minimize prediction error by identifying relevant variables and their associated regression coefficients. This is achieved by shrinking the regression coefficients toward zero by restricting the model factors [28]. Making sure that the total of the regression coefficients' absolute values stays below a preset threshold (λ) is the responsibility of this constraint. This constraint practically reduces the model's complexity. As a result, variables with regression coefficients that approach 0 are eliminated from the model. A common automated technique for selecting λ is the k-fold cross-validation approach. Using this approach, the database is separated into k equal-sized subsets at random [29]. The other $k-1$ subsets are used to build a prediction model, with one subset set aside for validation. Each of the k subsets is alternatively allocated for validation during the k cycles of this procedure, with the remaining subsets being used for model building. The culmination of these cycles involves consolidating the individual validation results across various λ values, ultimately selecting the optimal λ to define the final model. One important benefit of this approach is that it can reduce overfitting without restricting any subset of the database to be used for internal validation [30].

Although LAS has outperformed standard techniques in some situations, optimism bias and overfitting cannot be eliminated by using it. It does not remove the need to use an external database for model validation. Furthermore, for a better overall prediction expectancy, the LASSO technique trades off potential bias in projecting individual factors. One associated disadvantage is that since the focus is on optimizing combined prediction rather than the accuracy of projecting and interpreting the contribution of individual variables, the regression coefficients may not be reliably interpretable regarding independent risk factors. There are variations on the classic LASSO method, like elastic net and ridge regression, and research on the relative benefits of penalized regression methods is still ongoing.

2.3 Walrus optimization algorithm (WOA)

The natural behaviors of walruses dictate 3 distinct stages in which their positioning within the WOA is simulated.

Stage 1: Feeding approach (exploration).

A wide variety of marine invertebrates, including sea cucumbers, shrimp, soft corals, tunicates, tube worms, and various mollusks, are consumed by walruses. They consume over 60 distinct kinds of marine invertebrates. Still, they seem to choose benthic bivalve mollusks, particularly clams. They graze along the seafloor in search of these, using their sensitive vibrissae and vigorous flipper movements to find and detect food. During this excursion, the prominently tusked walrus takes on the position of leader, leading the group on their food search. There exists a correlation between the walrus tusk length and the quality of the objective function values of possible solutions. The possible resolution that has the best objective function value is therefore recognized as the dominant walrus in the group. Because of their foraging habits, walruses can explore a wider range of regions in the search domain, which improves the WOA's overall search capabilities. Eqs. (12) and (13) describe the mathematical modeling of the walruses' position update process, which is based on their feeding mechanism and is guided by the most influential group member. In this process, the walrus's initial position is first calculated using Eq. (12). As stated in Eq. (13), the newly found location takes the place of the previously determined one if it increases the value of the goal function.

$$x_{i,j}^{p1} = x_{i,j} + rand_{i,j}.(sw_j - I_{i,j}.x_{ij}) \tag{12}$$

$$X_i = \begin{cases} x_{i,j}^{p1}, F_{i,j}^{p1} < F_i \\ x_i, \ else \end{cases} \tag{13}$$

X_i^{P1} pertains to the newly calculated position of the $i-th$ walrus in the context of the first stage. Within this, F_i^{P1} indicates the objective function value associated with this specific walrus, and $x_{i,j}^{P1}$ displays its $j-th$ dimension. The variables $rand_{ij}$ represent random numbers falling within the [0, 1] range. SW corresponds to the leading possible resolution, symbolizing the most influential walrus. Furthermore, I_{ij} involves the selection of random integers within the range of 1 to 2. The role of I_{ij} is to enhance the algorithm's exploration capabilities. When it is set to 2, it results in more significant and extensive adjustments to the walruses' positions, as opposed to the default value of 1, which displays the standard degree of displacement. Enhancing the algorithm's global search, these conditions enable it to escape local optima and reveal the primary optimal region within the problem-solving space.

Stage 2: Migration

Walruses naturally go toward rocky outcrops and beaches as the temperature increases in the late summer. The WOA makes use of this migratory tendency to direct the walruses across the search domain and assist them in finding appropriate locations for exploration. Each walrus that is arbitrarily chosen from a distinct area of the search domain is thought to go to a separate location in this modeling. As a result, the new position was originally established using Eq. (14). Then, in line with Eq. (15), the walrus's previous location is replaced if this new position increases the value of the goal function. Eqs. (14) and (15) provide a mathematical representation of this behavioral mechanism.

$$x_{i,j}^{p2} = \begin{cases} x_{i,j} + rand_{i,j}.\left(x_{kj} - I_{ij}.x_{ij}\right), F_k < F_i \\ x_{i,j} + rand_{i,j}.\left(x_{i,j} - x_{kj}\right), else, \end{cases} \tag{14}$$

$$x_i = \begin{cases} x_i^{p2}, F_i^{p2} < F_i, \\ x_i, \ else, \end{cases} \tag{15}$$

Within this context, X_i^{P2} symbolizes the freshly calculated position for the $i-th$ walrus during the second stage. Here, x_{ij}^{P2} pertains to its $j-th$ dimension, while

F_i^{P2} corresponds to its objective function value. Additionally, X_k, where $k \in \{1, 2, \ldots, N\}$ and $k \neq i$, designates the location of the chosen walrus to which the $i-th$ walrus migrates. x_{kj} displays the $j-th$ dimension of this selected walrus, and F_k displays its respective objective function value.

Stage 3: Evasion and confrontation with predators (Exploitation)

The WOA improves its exploitation capabilities by imitating this natural behavior, which also helps it better explore the local search domain around potential solutions. Walruses are often in danger of coming into conflict with killer whales and polar bears. The walruses' positions change as a result of the tactics they use to avoid and defend against these predators; typically, these changes occur in the vicinity of their present locations. The WOA's design assumes that, as this process plays out close to each walrus's position, the range of walrus position changes takes place inside a neighborhood with a defined radius centered around each walrus. To emphasize global search in the early algorithm cycles and identify the ideal location in the search domain, the radius of this neighborhood is changeable. As the algorithm iterates, it starts with the highest value and gradually gets smaller. To solve this, the WOA includes local lower and upper boundaries at this stage, which causes the radius to change with each algorithm cycle. A neighborhood is established around every walrus in the WOA to simulate this behavior. First, using Eqs. (16) and (17), a new location is created at random inside this neighborhood. According to Eq. (18), this new position replaces the previous one if the objective function's value grows later.

$$x_{i,j}^{p3} = x_{ij} + (lb_{localj}^{t} + (ub_{localj}^{t} - rand.lb_{localj}^{t}) \tag{16}$$

$$local\ bounds = \begin{cases} lb_{localj}^{t} = \dfrac{lb_j}{t} \\ ub_{localj}^{t} = \dfrac{ub_j}{t} \end{cases} \tag{17}$$

$$x_i = \begin{cases} x_i^{p3}, F_i^{p3} < F_i \\ x_i, \quad else \end{cases} \tag{18}$$

In this context, X_i^{P3} signifies the freshly determined position for the $i-th$ walrus in accordance with the third stage. Here, x_{ij}^{P3} denotes its $j-th$ dimension, while F_i^{P3} corresponds to its objective function value. The variable t displays the cycle step, while lb_j and ub_j denote the lower and upper thresholds for the $j-th$ variable. Additionally, lb_{localj}^{t} and ub_{localj}^{t} are the local lower and local upper thresholds specifically applicable to the $j-th$ variable. These local thresholds simulate a localized search within the vicinity of the possible resolutions. Algorithm 1 shows the pseudocode of WOA.

Algorithm 1: Pseudocode of WOA.

```
Start WOA
  Input all optimization problem information.
  Set walrusses' number (N) and the iterations' total number (T).
  Initialization locations of walruses' process.
  For t = 1: N
  Phase 1: Exploration
    Calculate new location of the j^th walrus
    Update the i^th walrus location
  Phase 2: Migration
    Choose an immigration destination for the i^th walrus
    Calculate new location of the j^th walrus
    Update the i^th walrus location
  Phase 3: Exploitation
    Calculate new position in the neighborhood of the j^th walrus
    Update the i^th walrus location
  end
  Save the best candidate solution so far.
End WOA.
```

2.4 Crocodiles hunting strategy (CHS)

This section examines the optimization strategy known as CHS, which was first used as one of the feature clustering procedures [31].

2.4.1 Initializing

The initializing step must be finished before proceeding to the main stages, much like with other metaheuristic algorithms [32]. In the initializing process, a number of initial solutions are generated at random. The initial population of crocodiles consists of these responses that were chosen at random. These solutions have a homogeneous random distribution generated between the lower and upper thresholds. These solutions are obtained by using the following equation:

$$x = LB + r \times (UB - LB) \tag{19}$$

After basic factors such as population size, maximum number of cycles, and lower and upper thresholds of variables have been defined, random solutions (x) are

generated based on (2), where LB and UB are the problem's lower and upper limits, respectively. Moreover, r has a generation interval of $0\ to\ 1$ and is a uniform random variable. These solutions are then evaluated using the objective function. CHS operators evaluate the solutions using the objective function. The ideal solution gets replaced when a better one comes along. The optimal cure (x_{prey}) displays the smallest objective function value of the superior resolution.

Chasing the Prey

This section mimics what the chasers would do. As previously said, the population is divided in half. Thus, each sector displays half of the population. Included in the group of chasers are the first 50% of the answers. Therefore, ambushers make up the latter half of fifty percent of the population. These 2 groups are arbitrarily divided into 2 ambushers and 2 chaser groups. The reasoning behind replicating chaser behavior is based on the distance between prey and crocodiles that resembles that of chasers. The prey is led to the shore and other shallow regions by another set of hunters called the chasers, who pursue it without actually catching it. The following are the recommended equations:

$$d^{i,t} = \left|x_{prey}^{t} - x_{chaser}^{i,t}\right|,\ \ for\ all\ i\ , \tag{20}$$

$$\begin{cases} x_{chaser}^{i,t+1} = \left(x_{chaser}^{i,t} - \beta.d\right)\ \ for\ all\ i(\dfrac{\ln(it)}{\ln\left(\max it\right)} \times r) \geq 0.5 \\ x_{chaser}^{i,t+1} = |\beta - (\beta.d)|\ \ for\ all\ i(\dfrac{\ln(it)}{\ln(maxit)} \times r < 0.5 \end{cases} \tag{21}$$

In this instance, the prey location at cycle t is represented by x_{prey}^{t}, while the chaser's position at cycle t is reflected by $x_{chaser}^{i,t}$. The coefficient β has a uniform distribution that ranges from $[-3,3]$. This r is an arbitrary number in the interval $0-1$. It also refers to the local and global cycle numbers in inner loops. After running the program several times, these intervals were measured empirically. Eq. (21) provides the distance d between the prey and its chase. 2 requirements are satisfied by Eq. (21): first, if it$\frac{it}{maxit} \times r \geq 0.5$, which indicates that every chaser. A random search is carried out if the movement of the first group toward the prey had a positive coefficient and it reached its maximum when r was less than 0.5. To sum up, Eq. (21) shows that by employing an intelligent technique, the second is more fully implemented, and a random search is carried out in the initial cycles. In the most recent versions, it is more fully implemented, and the chasers approach the target at random.

2.4.2 Attacking the prey

The ambushers are poised to grab their victim at the final destination of their objective. While the pursuers try to guide the victim to this spot or the attack area, the ambushers hide in the final position. Attackers are stated to be forced to change positions in order to replicate the attacking stage by using the following equations:

$$d^{i,t} = \left|x_{prey}^{t} - x_{ambusher}^{i,t}\right|, for\ all\ i, \tag{22}$$

$$A = \frac{apc + apa + x_{prey}}{3} \tag{23}$$

$$\begin{cases} x_{ambusher}^{i,t+1} = (d.\cos(2\pi r) + x_{prey}^{t}\ \ for\ all\ i\ (p) \geq 0.5 \\ x_{ambusher}^{i,t+1} = (x_{ambusher}^{i,t} - \beta\left(A - x_{ambusher}^{i,t}\right))\ \ for\ all\ i\ (p) < 0.5 \end{cases} \tag{24}$$

where $p = \frac{\ln\left(it\right)}{\ln\left(maxit\right)} \times r$, in this case, d indicates the separation between the ambushers and the prey, and $x_{ambusher}^{i,t}$ displays the position of ith ambusher at cycle t. Additionally, x_{prey} is the best position or prey position, apa is the average position of ambushers, and apc is the average position of chasers. Crocodiles adjust their posture based on average position or the locations of their prey in every category. Thus, if$(\frac{\ln(it)}{\ln(,axit)} \times r < 0.5$, meaning that every ambusher rotates around the prey; if $\frac{\ln(it)}{\ln(,axit)} \times r \geq 0.5$ ambushers adjust their position based on the average positions of all groups of chasers, ambushers, and prey. The flowchart of the CHS is presented in Fig. 2.

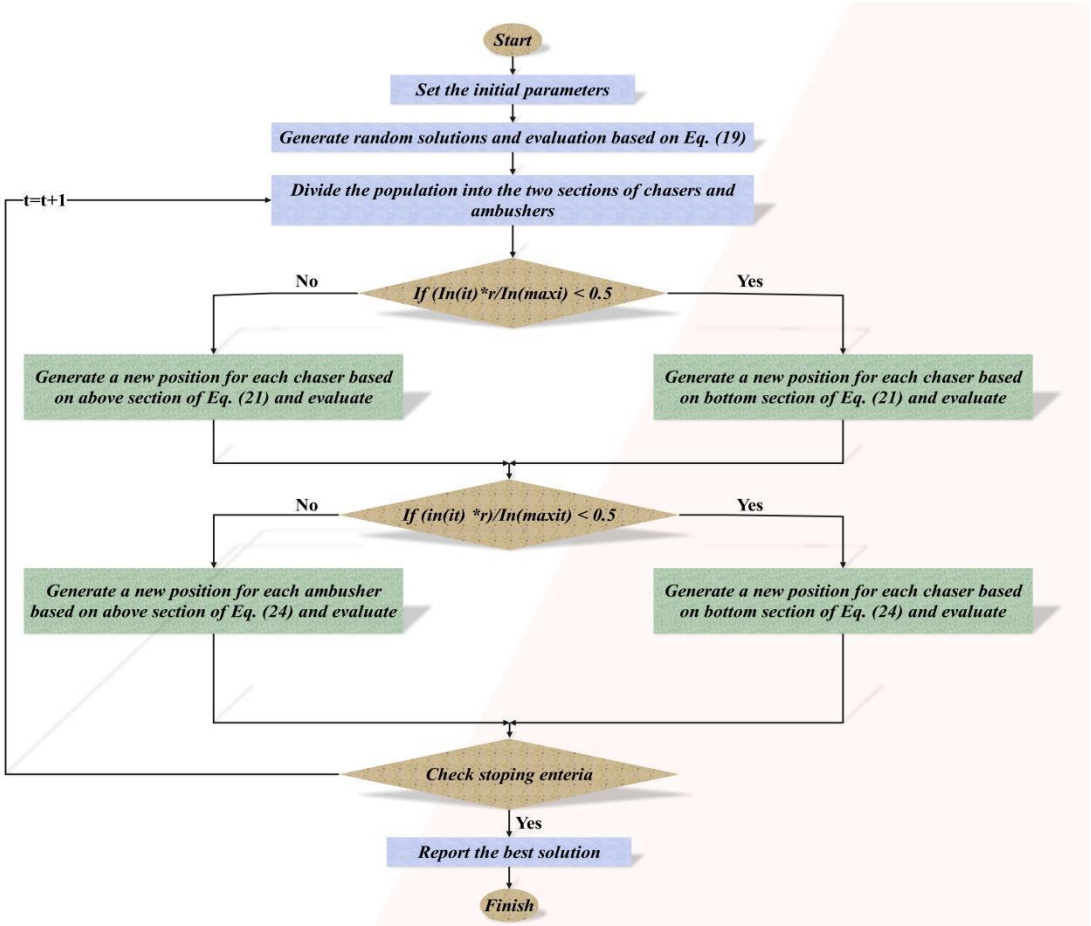


Figure 2: The flowchart of the CHS.

2.5 Performance evaluators

Several performance metrics, such as the coefficient correlation (R^2), symmetric mean absolute percentage error (SMAPE), MSE, RMSE, and coefficient variance of RMSE (CV_{RMSE}), have been used in this study to evaluate the efficacy of the model. Below are the formulas for these performance standards:

$$SMAPE = \frac{100}{n}\sum_{i}^{n}\frac{2\times|m_i - b_i|}{|m_i| + |b_i|} \tag{25}$$

$$R^2 = \left(\frac{\sum_{i=1}^{n}(b_i - \bar{b})(m_i - \bar{m})}{\sqrt{\left[\sum_{i=1}^{n}(b_i - \bar{b})^2\right]\left[\sum_{i=1}^{n}(m_i - \bar{m})^2\right]}}\right)^2 \tag{26}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(m_i - b_i)^2} \tag{27}$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(m_i - b_i)^2 \tag{28}$$

$$CV_{RMSE} = \left(\frac{RMSE}{\bar{t}}\right)\times 100 \tag{29}$$

$$CI = \left(\sqrt{\frac{n\times metric^2}{X^2_{a/2,n}}}, \sqrt{\frac{n\times metric^2}{X^2_{1-a/2,n}}}\right) \tag{30}$$

It is also possible to express the variables as follows:

- n is a symbol for sample size.
- b_i is the expected value.
- The measured and mean predicted values are denoted by $\bar{m}$ and $\bar{b}$, respectively.
- m_i displays the gauged value.
- t displays the degrees of freedom, and the crucial value of the t-distribution depends on the desired level of confidence.
- $X^2_{a/2,n}$ and $X^2_{1-a/2,n}$ are critical values from the chi-squared distribution with n degrees of freedom.
- a shows significance level.
- $metric$ represents a component that can be placed on each evaluation to determine the CI.

3 Data collection

The variables are presented as factors to describe and analyze computer schemes or systems. “Model Name” serves as a unique identifier for a specific computer model, while “MYCT” displays the machine cycle time, indicating the duration of one instruction cycle in nanoseconds. “MMIN” and “MMAX” denote the minimum and maximum main memory capacities, respectively, measured in kilobytes. “CACH” stands for cache memory size in kilobytes, storing frequently

accessed data for faster CPU access. “CHMIN” and “CHMAX” signify the minimum and maximum numbers of channels or connections for I/O devices. Finally, “PRP” displays the published relative performance, a benchmark score comparing the performance of one model against others. These variables collectively provide insights into the specifications, performance, and capabilities of various computer schemes, aiding in performance analysis, system optimization, and comparative evaluations.

The dataset contains 209 samples, which were gathered from Kaggle (https://www.kaggle.com/datasets/abdelazizsami/computer-hardware).

Before modeling, a basic preprocessing pipeline was applied:

- No missing values were found in the dataset, so no imputation was required.
- All feature values were numerical, eliminating the need for encoding.
- Features were normalized using Min-Max scaling to ensure that the optimization algorithms and learning models operate within the same scale, improving convergence and stability.

The dataset was randomly split into 70% for training (145 samples) and 30% for testing (64 samples) to evaluate generalization performance. Additionally, during the model development phase, a 5-fold cross-validation strategy was employed on the training set to fine-tune model hyperparameters and assess performance consistency. This ensures that the models are robust and not overfitted to any specific subset of the data. The dataset was split into 70% for train and 30% for testing. Fig. 3 displays the 3D bar for the correlation of the input and output variables.

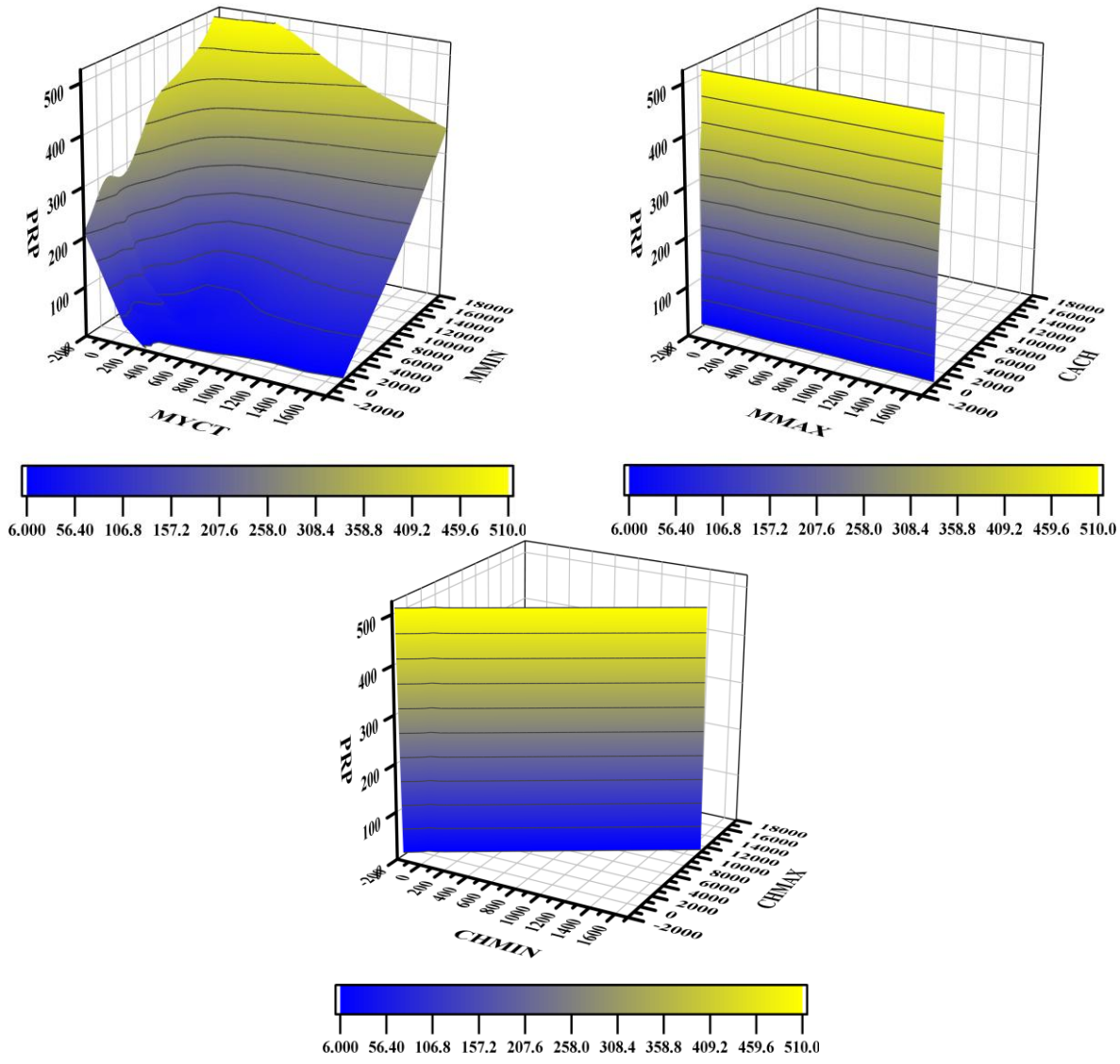


Figure 3: 3D surface plot for the correlation of input and output

4 Results

4.1 Convergence and hyperparameter

Fig. 4 shows the convergence curve for the hybrid schemes. It provides information about how the algorithm is performing as it approaches the optimal solution by visually representing the dynamic variations in the objective function value for successive cycles. The objective function frequently exhibits a constant trend of growing or lowering as convergence gets closer until it reaches a point where doing more cycles marginally improves the situation. Significant information about the algorithm's convergence behavior and best-answer path is provided by this graphical depiction. The graph makes evident the differences in operation between the LACH and LAWO schemes. The LACH model shows great promise, achieving an accuracy of 12.134 after around 135 cycles. Conversely, the LAWO model requires around 110 cycles to reach its highest accuracy of 17.139. After 140 cycles, the SVCH model achieves its maximum accuracy of 27.429, whereas the SVWO model gets its accuracy of 30.313 after 125 iterations.

As expected, the single models (SVR and Lasso) demonstrate significantly lower computational overhead. SVR completes in approximately 0.5 seconds, and Lasso runs even faster, requiring only 0.3 seconds. This is due to their deterministic nature and lack of iterative optimization beyond the internal model fitting. In contrast, the hybrid models, which involve repeated evaluation and optimization of hyperparameters using CHS or WOA, require considerably more time. Among these, SVR + CHS is the most time-consuming, with a runtime of 40 seconds, likely due to both the complexity of SVR and the iterative search process of CHS. SVR + WOA is slightly more efficient, requiring 32 seconds, indicating that WOA may converge faster or be computationally lighter than CHS in this context. On the Lasso side, the Lasso + CHS hybrid consumes 26 seconds, while Lasso + WOA is the fastest among hybrids, completing in 20 seconds. This performance difference may stem from Lasso's inherently simpler structure and faster convergence when compared to SVR, combined with WOA's relatively lower computational footprint.

In addition, Table 2 presents the optimized hyperparameters for both SVR-based and Lasso-based models. For SVR variants, the metaheuristic-optimized models (SVCH and SVWO) show significant improvements over the baseline SVR, with higher penalty values (C) and tuned epsilon (ε) settings. SVCH uses a larger C and moderate ε, promoting lower training error, while SVWO opts for tighter tolerance (ε = 0.001) for precision. Both hybrids use γ = 1 for kernel influence. For Lasso-based models, the regularization parameter α varies: LAS uses a default value of 1, while LAWO and LACH use higher values (50.0 and 13.5, respectively) optimized by WOA and CHS. LACH's moderate α suggests a balanced trade-off between sparsity and model flexibility, aligning with its top performance. Overall, the metaheuristic algorithms effectively fine-tune hyperparameters to enhance prediction accuracy beyond the baseline configurations. Notably, the parameters are selected based on the random search method.

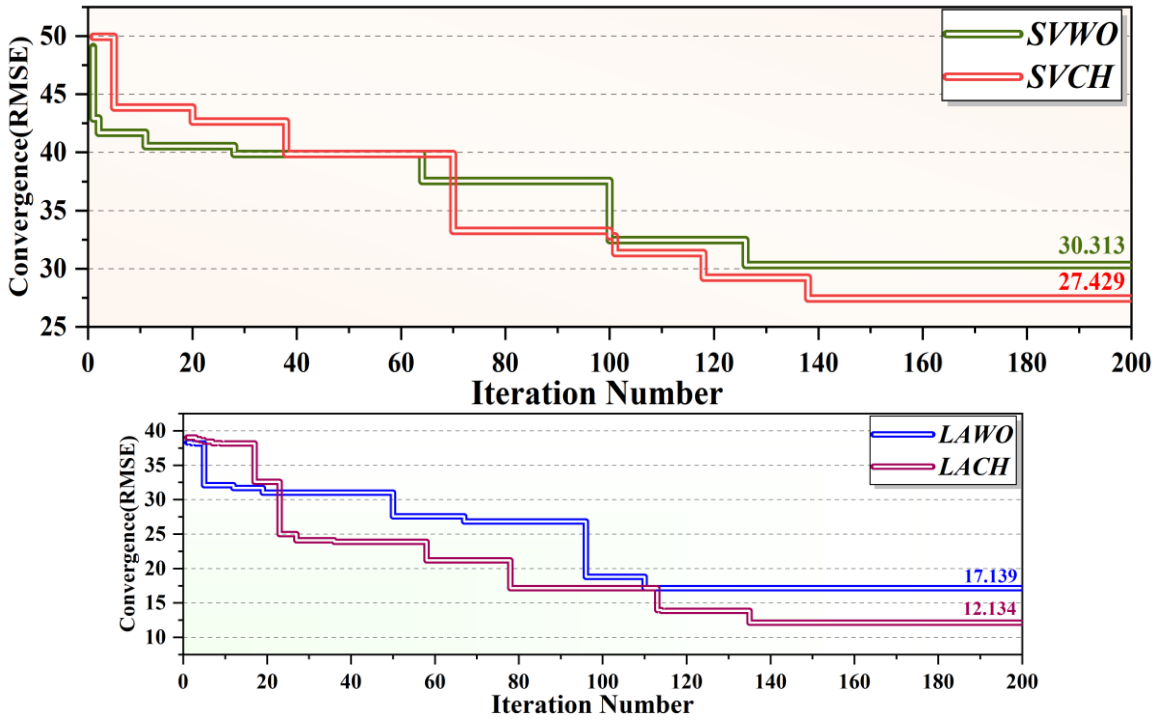


Figure 4: Convergence plot of hybrid schemes

Table 2: Result of the hyperprameters

Models	Hyperprameter		
	C	Epsilon	Gama
SVCH	415.5	0.339	1
SVWO	169.2	0.001	1
SVR	1	0.1	scale
	alpha		
LACH	13.50		
LAWO	50.00		
LAS	1		

4.2 Schemes comparison

The results of the generated schemes, which were evaluated using 5 metrics and 3 different stages, are shown in Table 3. The stages encompass training, validation, and testing, while the metrics comprise RMSE, R^2, MSE, and SMAPE. During the training stage, LACH demonstrated superior performance regarding RMSE, achieving a value of 13.228, followed closely by the LAWO scheme with a performance metric of 18.780, ranking second in performance.

Transitioning to the validation stage, the LACH model showcased exceptional performance with an R^2 value of 0.993, securing the top position. Conversely, the LAS model emerged as the third-best performer with an R^2 value of 0.985. Upon entering the testing stage, the SVCH scheme exhibited the 4th-best performance, attaining an MSE value of 638.8. Subsequently, the SVWO model trailed closely behind with the fifth-best performance, registering an MSE value of 809.3. In the Validation stage, the LACH model excelled in SMAPE, achieving a value of 0.002, while the LAWO model attained the second-highest performance with a value of 0.003. Furthermore, regarding MSE during the validation stage, the LAS model ranked third with a value of 234.6, while the SVR model demonstrated the weakest performance, recording an MSE value of 1074.8.

Notably, the code of models and optimizers provided in GitHub (https://github.com/Chliu32/Model-and-optimizer).

Table 3: The result of developed schemes for LAS and SVR

Model	Stage	Index values						
		RMSE	R^2	MSE	SMAPE	CV_{RMSE}	CI_{R^2}	CI_{RMSE}
SVR	Train	35.492	0.898	1259.7	0.002	0.377	0.8618	23.145
	Validation	32.785	0.915	1074.8	0.009	0.318		
	Test	31.431	0.660	987.9	0.018	0.620		
SVWO	Train	31.551	0.928	995.5	0.002	0.335	0.8899	20.655
	Validation	25.985	0.946	675.2	0.007	0.252		
	Test	28.449	0.737	809.3	0.017	0.561		
SVCH	Train	27.377	0.940	749.5	0.002	0.291	0.7453	31.416
	Validation	28.880	0.948	834.0	0.008	0.280		
	Test	26.150	0.812	683.8	0.014	0.516		
LAS	Train	21.710	0.963	471.3	0.001	0.231	0.9798	8.854
	Validation	15.317	0.985	234.6	0.004	0.149		
	Test	13.859	0.976	192.1	0.006	0.273		
LAWO	Train	18.780	0.973	352.7	0.001	0.199	0.9560	13.059
	Validation	12.407	0.987	153.9	0.003	0.120		
	Test	12.716	0.975	161.7	0.006	0.251		
LACH	Train	13.288	0.987	176.6	0.001	0.141	0.9411	15.110
	Validation	8.778	0.993	77.1	0.002	0.085		
	Test	9.068	0.990	82.2	0.004	0.004		

Fig. 5 shows a plot of the dispersion of emerging hybrid schemes. The measured values are shown on the X axis, while the predicted values are displayed on the Y axis. The optimal performance of the model is indicated by the colored zone surrounding the R^2 value, often known as the center line. To provide clarity, Fig. 5 shows this shading in several tones. Data points that are below the center line are shown as underestimation, while those that are above it are shown as overestimation. The model is considered to be functioning correctly when there is a minor angle between the linear line and the center line. The productivity of the LACH scheme exceeds that of the LAWO, LAS, SVCH, SVWO, and SVR schemes, as shown in Fig. 5. The second-best performance is exhibited by the LAWO model, which trails the LAS model by a narrow margin.

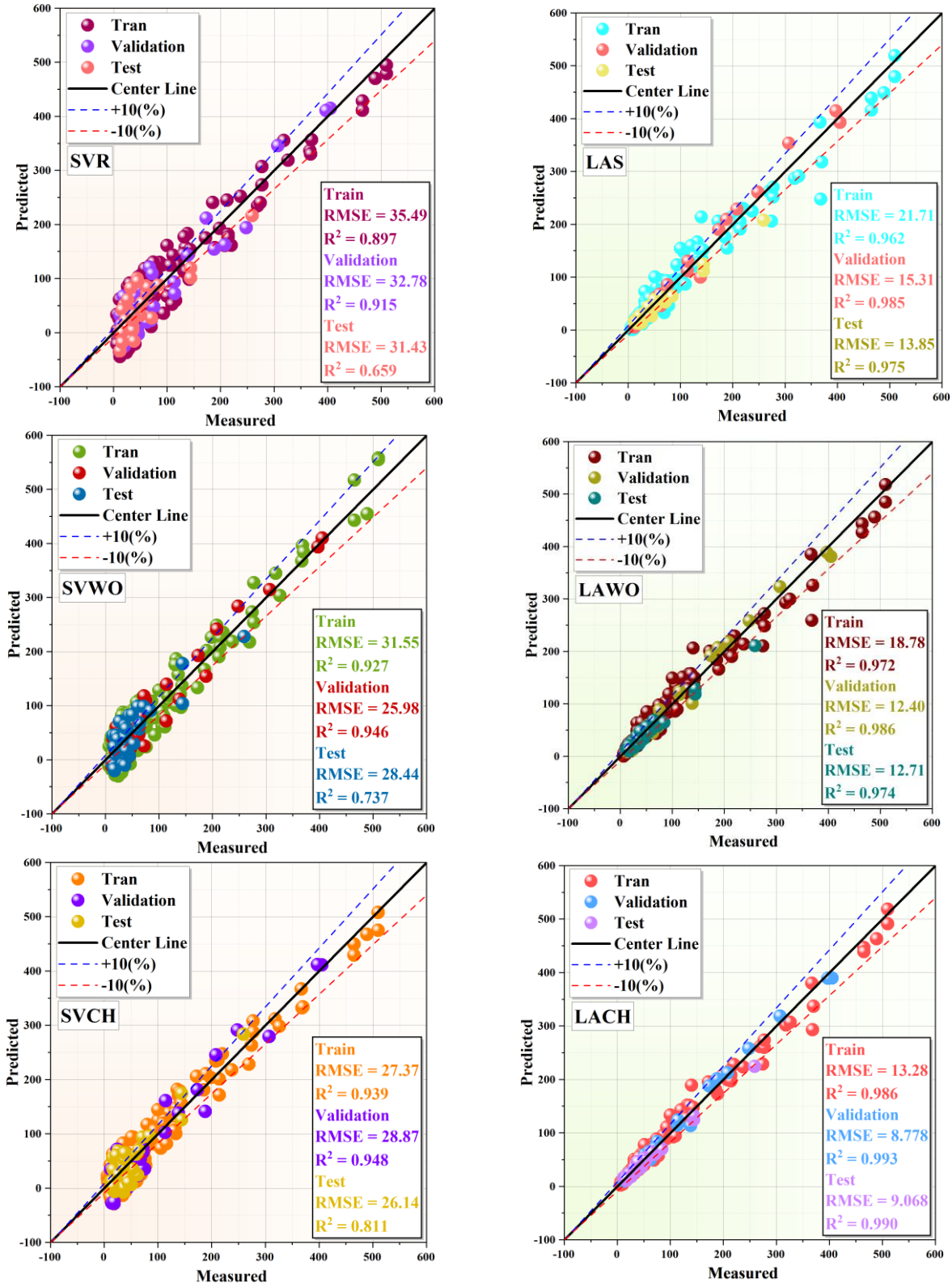
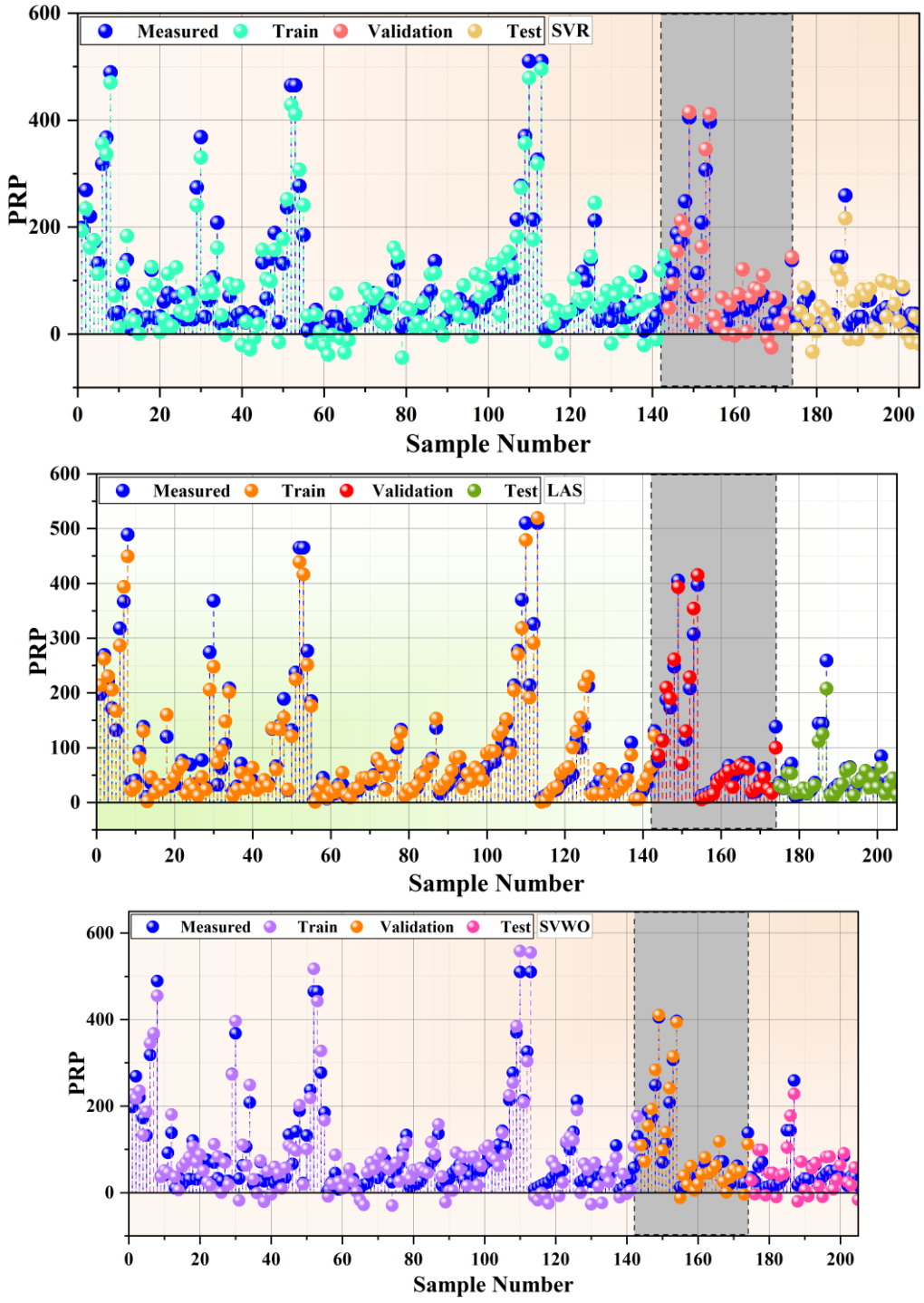


Figure 5: Plotting the dispersion of evolved hybrid schemes

A comparison between estimated and actual values is presented in Fig. 6. The line that should match the measured line is the visual cue that shows how accurate the forecast made by the model is. When the measured line and the anticipated column are almost in line, there is a strong sign of accuracy; when not, there may be inefficiencies. The LACH model demonstrated strong performance during training, with very few measured lines displaying lengths larger than the predicted column. This model performed better in the validation stage than in the test stage, and its predictions were rather close to the actual data. In contrast to the LACH model, the LAWO model exhibited weaker performance during the training stage, as evidenced by the fewer predicted columns

deviating from observed lines. Nevertheless, the LAWO model excelled in both the test and validation stages, demonstrating strong performance.



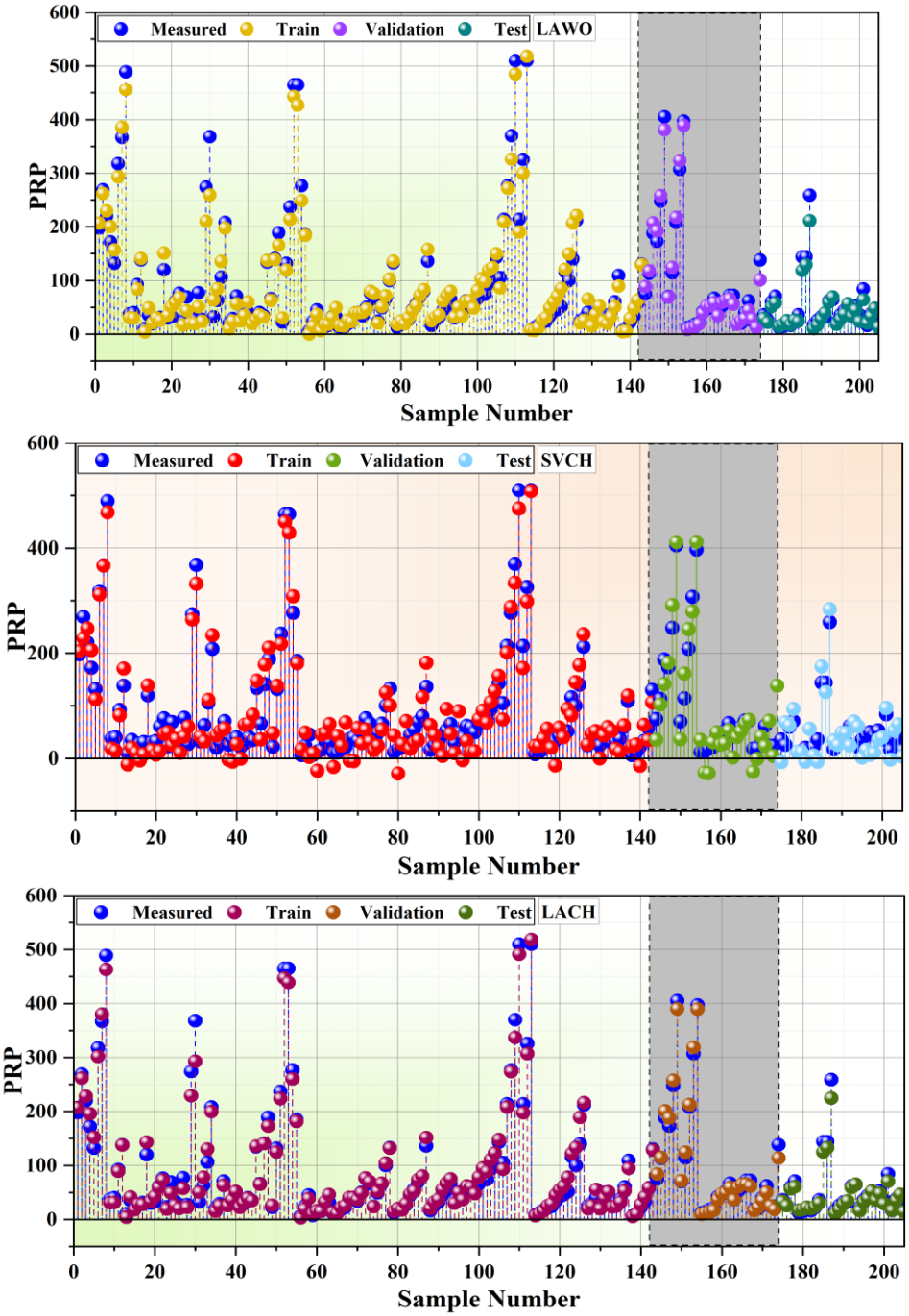
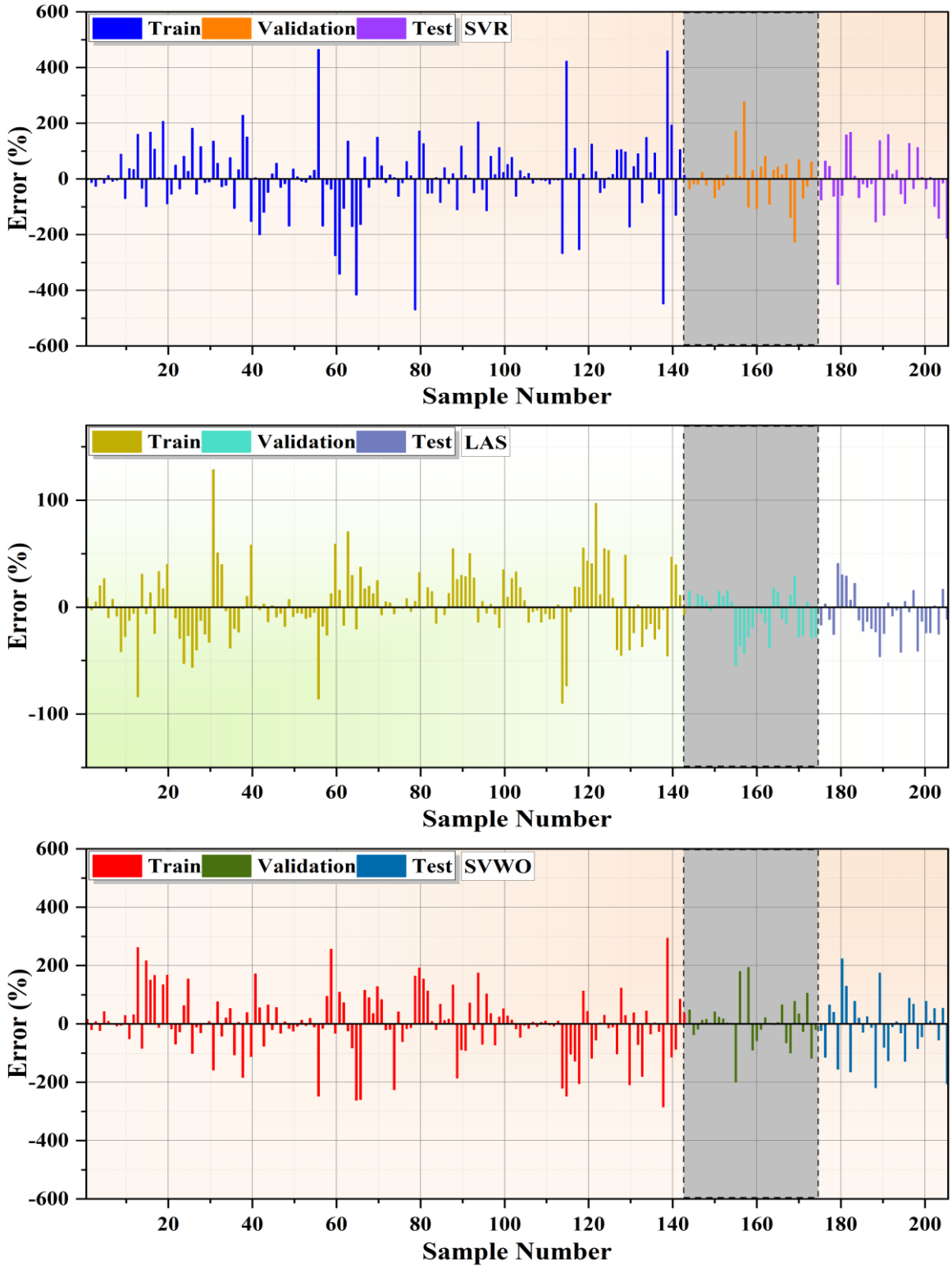


Figure 6: Comparison between estimated and actual values

The scatter plot-based error percentage of the schemes is displayed in Fig. 7. When the error rate is almost zero, the model is doing exceptionally well. The LACH performs well during training, as seen by an error rate that varies from (-60) to (52). What stands out in particular is the maximum error rate in the test stage, which is represented as (60) in Fig. 7. This shows how much better the LACH model is than the other 5 schemes. Conversely, compared to the LACH scheme, the LAS model, as shown in Fig. 7, exhibits noticeably more variation in error rate throughout the training period. It is noteworthy that at the validation stage, the LAS model attains an approximate minimal error rate of 120, suggesting a markedly better performance than the SVWO model. Among all schemes, the SVR model has the largest error rate during the training stage, indicating comparatively poorer performance during this period. Notably, the SVR model outperforms the GPCS and GPAR schemes regarding error rate, displaying the greatest maximum error rate of roughly 450 across all sections.


Train
Validation
Test
SVR
Error (%)
Sample Number
Train
Validation
Test
LAS
Error (%)
Sample Number
Train
Validation
Test
SVWO
Error (%)
Sample Number

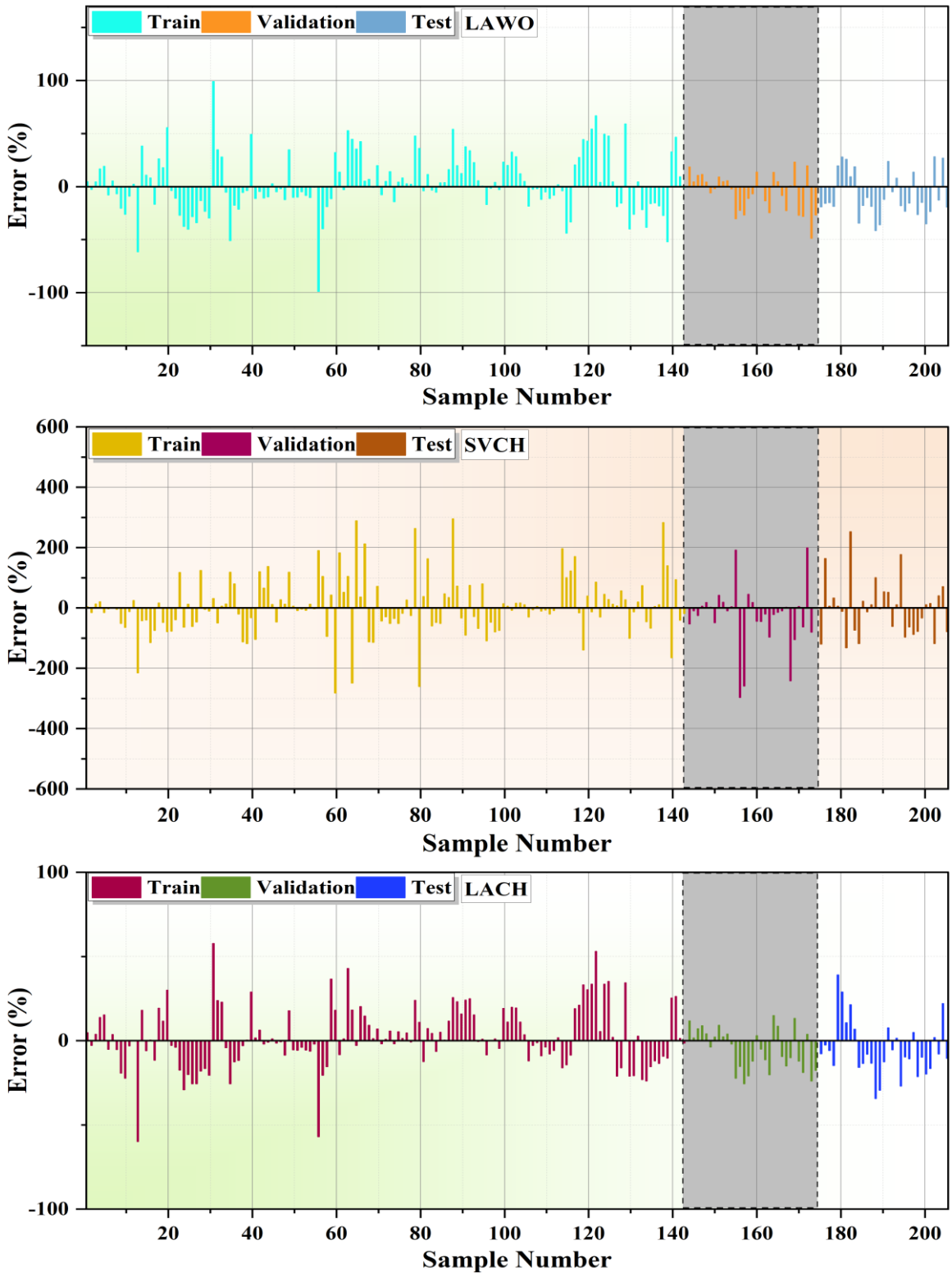


Figure 7: The schemes' error percentages for each of the sample values are displayed

The suggested schemes' half-box with symbol plot errors is displayed in Fig. 8. The closer the median line is to zero, the lower the error rate of the scheme. Analyzing the distribution of data within the interquartile range (IQR) might provide more details about the model's performance; a denser concentration of data within this range indicates more efficacy. When the LACH model is analyzed for training, the population is found to be strongly clustered around the confidence interval (IQR), with a few outliers. The model's concentration in the IQR range demonstrates its good performance. The model's lower error rate in the validation step is further corroborated by the median line's nearness to zero. On the other hand, the median line is situated apart from zero in the SVWO model, and there is a noticeable dispersion in the IQR range, especially during the test stage. Even in the

case when the population during the training stage is generally concentrated around the IQR range, a wider IQR implies potential unpredictability in the model's performance.

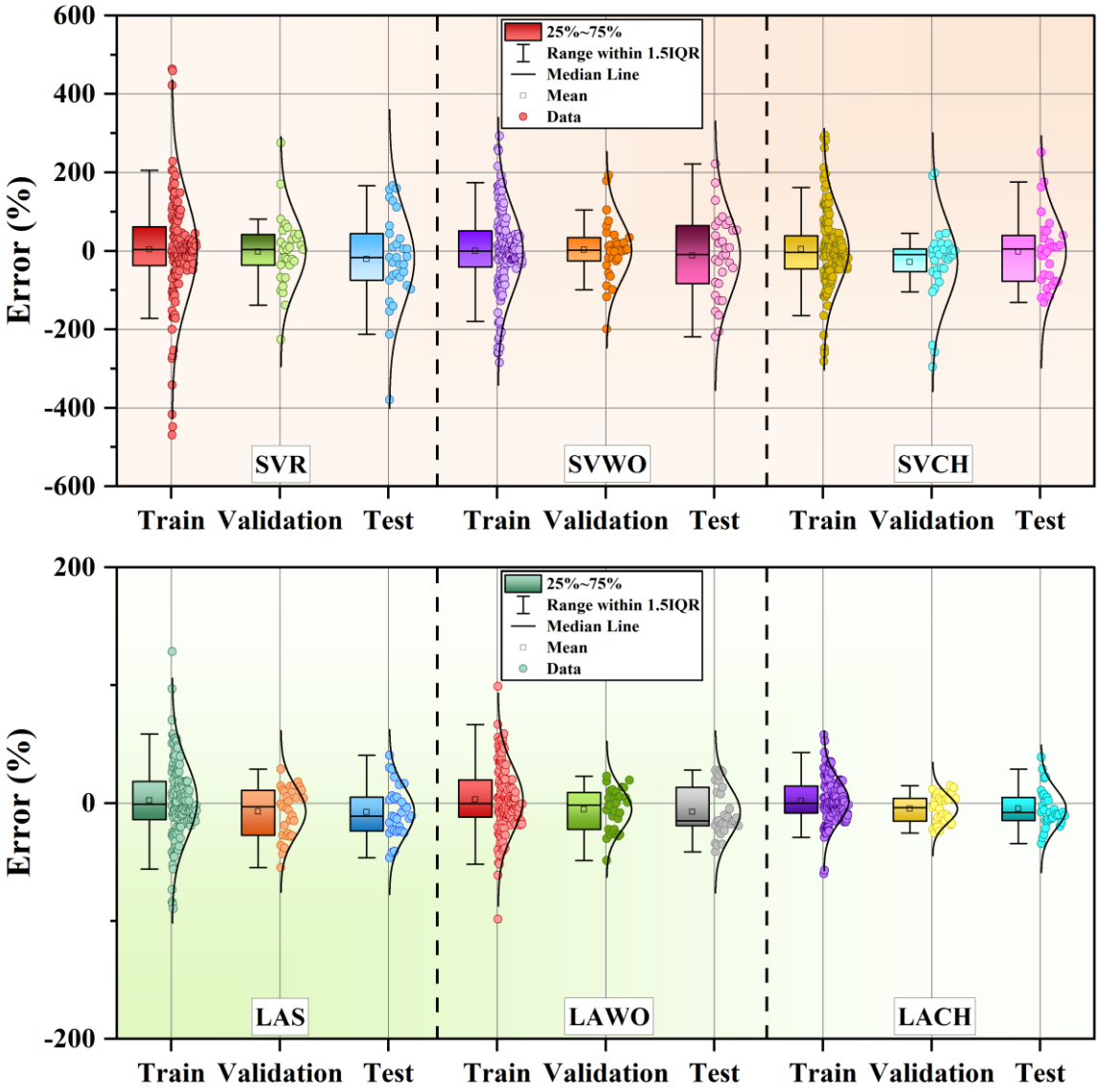


Figure 8: The half box with symbol plot errors of recommended schemes

To evaluate the statistical significance of the differences in performance between the developed models, the Wilcoxon signed-rank test was conducted, and the results are presented in Table 4. This non-parametric test is suitable for comparing paired data distributions and is particularly useful when the assumptions of normality are not satisfied, which is often the case in model performance metrics across repeated trials. The table lists the p-values and corresponding test statistics for each pairwise comparison among the baseline and hybrid models.

The results show that none of the comparisons between SVCH and the other models (SVWO, SVR, LACH, LAWO, and LAS) resulted in statistically significant differences, as all p-values are above the standard significance threshold of 0.05. Specifically, p-values for SVCH comparisons range from 0.204 to 0.269, suggesting that while SVCH may offer performance improvements numerically, these differences are not statistically significant at the 95% confidence level.

Similarly, comparisons between SVWO and other models, including SVR, LACH, LAWO, and LAS, yielded p-values between 0.425 and 0.705. These results again indicate no significant statistical difference in performance between SVWO and the compared models. Even though hybridization with the Walrus Optimization Algorithm appears to improve the prediction capability over the standalone SVR, these improvements are not strong enough to be considered statistically significant.

When comparing SVR with LAS-based models (LACH, LAWO, and LAS), the p-values are all above 0.67, further indicating no statistically significant difference despite the superior numerical performance observed in the LAS-based models. Notably, the LACH model demonstrated the highest accuracy across evaluation metrics such as RMSE, R^2, and MSE.

However, the comparison of LACH with its base model LAS resulted in a p-value of 0.267, again above the 0.05 threshold, implying that even the most improved hybrid model does not exhibit a statistically significant advantage over its non-hybrid baseline.

These results suggest that although the hybrid models, particularly LACH and LAWO, achieve numerically better performance in validation and testing stages, these improvements are not statistically significant under the Wilcoxon test. This may be attributed to several factors, including the relatively small dataset size (209 samples), low variance in performance across folds, and the inherently similar structure of the compared models. Therefore, while LACH may be favored in practical applications due to its strong numerical performance, the statistical findings call for cautious interpretation and indicate the need for further experimentation using larger or more diverse datasets to draw more definitive conclusions about the effectiveness of hybridization.

Table 4: Result of statistical analyses based on Wilcoxon

Models' difference	p-value	stats
SVCH with SVWO	0.241	9560
SVCH with SVR	0.265	9609
SVCH with LACH	0.204	9478
SVCH with LAWO	0.227	9530
SVCH with LAS	0.269	9618
SVWO with SVR	0.705	10235
SVWO with LACH	0.491	9972
SVWO with LAWO	0.446	9909
SVWO with LAS	0.425	9879
SVR with LACH	0.771	10310
SVR with LAWO	0.670	10195
SVR with LAS	0.724	10257
LACH with LAWO	0.677	10203
LACH with LAS	0.267	9613
LAWO with LAS	0.448	9912

5 Discussion

This section provides an in-depth comparative analysis of the proposed hybrid models against their base counterparts and discusses their practical implications, generalization abilities, and computational cost in the context of heterogeneous CPU scheduling.

5.1 Comparative analysis with baseline models

The results in Table 3 demonstrate that the hybrid LACH model (LAS + CHS) significantly outperforms both its non-hybrid counterpart, LAS, and the SVR-based models in all stages (train, validation, test). For instance, in the test phase, LACH achieves an RMSE of 9.068 and R^2 of 0.990, while the base LAS model scores an RMSE of 13.859 and R^2 of 0.976. The improvement in RMSE (a ~34.6% reduction) and R^2 reflects the hybrid model's superior ability to learn complex non-linear relationships within the dataset.

Compared to SVR-based models, LACH's superiority is even more pronounced. The base SVR model's test R^2 is only 0.660, with an RMSE of 31.431, highlighting the model's limited capacity to capture workload variability. Even with optimization (SVWO and SVCH), the SVR models remain less accurate than LAS-based counterparts, likely due to the regularization strengths and inherent sparsity handling of Lasso Regression, which contribute to better generalization.

5.2 Why LACH outperforms LAWO

Though both LACH and LAWO are built upon the LAS model, their performance differs due to the distinct nature of the optimization strategies employed. The CHS incorporates dynamic leadership-based exploration and intensified exploitation phases, which enhance convergence speed and model fine-tuning. In contrast, WOA's spiral search and bubble-net mechanisms, while effective, may lead to suboptimal local minima under certain configurations. This difference is reflected in the test RMSE values: 9.068 for LACH versus 12.716 for LAWO. Thus, CHS provides a more adaptive and efficient search mechanism for hyperparameter tuning in this context.

5.3 Lower performance of SVCH

SVCH, the hybrid of SVR with CHS, exhibits better performance than the baseline SVR but remains inferior to LAS-based hybrids. The underlying reason lies in SVR's sensitivity to kernel parameter tuning and its assumption of a fixed margin around the true value. Even with CHS optimization, the SVR model's rigidity in handling high-dimensional heterogeneity leads to suboptimal generalization. Furthermore, SVR's reliance on support vectors makes it less adaptable when dealing with noisy or diverse input features, reducing its overall robustness compared to Lasso-based models.

5.4 Real-world implications

From a practical perspective, accurate CPU workload prediction enables more intelligent scheduling in heterogeneous environments—where decisions must be made in real-time with limited information. LACH's high R^2 and low RMSE across all stages indicate reliable generalization to unseen workloads, which is critical for real-world deployment. Its ability to reduce prediction error allows schedulers to better match workloads with appropriate cores (e.g., high-power vs. low-power), ultimately improving throughput and energy efficiency.

5.5 Computational cost and generalization

While hybrid models like LACH show superior prediction performance, they do incur additional computational costs during the training phase due to the metaheuristic optimization process. However, this cost is one-time and can be amortized since predictions are fast once the model is trained. Moreover, the use of Lasso Regression—known for its ability to perform feature selection—further reduces overfitting and enhances generalization, as evidenced by the low test-stage RMSE and high R^2. In contrast, the SVR-based models, despite being computationally lighter during optimization, fail to generalize well, especially in test scenarios with unseen workloads. This trade-off highlights the importance of choosing not just an effective learning algorithm but also a compatible optimizer that enhances its inherent strengths.

5.6 Scalability across cores, architectures, and workloads

While the proposed hybrid models (SVR + CHS, SVR + WOA, Lasso + CHS, and Lasso + WOA) demonstrate improved prediction accuracy, their scalability across varying hardware configurations remains unexplored. Real-world heterogeneous systems often include multiple CPU cores, diverse architectures, and mixed workloads, which can significantly affect performance dynamics. The current models are trained on a fixed dataset and may not generalize well to systems with higher core counts or architectural differences. Factors such as thread-level parallelism, cache hierarchy, and interconnect latency can introduce non-linear behaviors that are not captured in the existing feature set. Additionally, workload diversity (compute-bound, memory-bound, I/O-bound) could lead to underperformance if not properly modeled. Training hybrid models is also more computationally expensive than single models. As systems scale, thias cost can increase. To enhance scalability, future work should consider integrating architecture-aware features, workload profiling, and possibly distributed or parallel implementations of CHS and WOA. Incorporating online or incremental learning techniques could further improve adaptability to unseen hardware configurations and dynamic workloads.

6 Conclusion

This work investigated the potential of ML schemes for enhancing CPU performance prediction in heterogeneous systems. SVR and LAS were explored as the foundation, with their capabilities further strengthened by incorporating CHS and WOA. The base model hybridized with the optimizers and resulted in these: LASSO + CHS (LACH), LASSO + WOA (LAWO), SVR + CHS (SVCH), and SVR + WOA (SVWO). Regarding RMSE, the LACH model performed better during the training stage, with a score of 13.228, closely followed by the LAWO model, which ranked second in performance with a performance measure of 18.780. As the model moved into the validation stage, the LACH model performed exceptionally well, achieving an R^2 value of 0.993 and taking the lead. On the other hand, with an R^2 value of 0.985, the LAS scheme turned out to have the third-best performance. The SVCH scheme performed the 4th best when it entered the testing stage, with an MSE score of 638.8. Then, not far behind, the SVWO model registered an MSE score of 809.3, placing it in fifth place overall. This approach addressed the challenge of accurately predicting CPU performance on diverse computing resources. By leveraging the strengths of both regression techniques and optimization algorithms, the proposed methodology aimed to pave the way for more informed scheduling decisions and, ultimately, improved system throughput in heterogeneous environments. Future research directions could involve exploring additional ML schemes or optimizers, as well as investigating the application of this approach to real-world heterogeneous computing workloads.

Acknowledgments

I would like to take this opportunity to acknowledge that there are no individuals or organizations that require acknowledgment for their contributions to this investigation.

Authors' contributions

All authors contributed to the study's conception and design. Data collection, simulation and analysis were performed by" Chunxia LIU and Yuquan ZHOU ". Also, the first draft of the manuscript was written by Chunxia LIU. Ling Ding commented on previous versions of the manuscript.

Availability of data and materials

Data can be shared upon request.

Conflicts of interest

The scholars claimed no conflicts of interest considering this investigation.

Author statement

The manuscript has been read and approved by all the authors, the requirements for authorship, as stated earlier in this document, have been met, and each author believes that the manuscript displays honest work.

Funding

This investigation received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Ethical approval

The paper has received ethical approval from the institutional review board, ensuring the protection of participants' rights and compliance with the relevant ethical guidelines.

References

[1] C.-K. Luk, S. Hong, and H. Kim, "Qilin: exploiting parallelism on heterogeneous multiprocessors with adaptive mapping," in *Proceedings of the 42nd Annual IEEE/ACM international symposium on microarchitecture*, 2009, pp. 45–55. https://doi.org/10.1145/1669112.1669121

[2] D. Grewe and M. F. P. O'Boyle, "A static task partitioning approach for heterogeneous systems using OpenCL," in *Compiler Construction: 20th International Conference, CC 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011, Saarbrücken, Germany, March 26–April 3, 2011. Proceedings 20*, Springer, 2011, pp. 286–305. https://doi.org/10.1007/978-3-642-19861-8_16

[3] B. N. Chandrashekhar and H. A. Sanjay, "Prediction model of an HPC application on CPU-GPU cluster using machine learning techniques," in *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, IEEE, 2020, pp. 92–97. https://doi.org/10.1109/ICIMIA48430.2020.9074866

[4] R. S. S. Dittakavi, "Deep Learning-Based Prediction of CPU and Memory Consumption for Cost-Efficient Cloud Resource Allocation," *Sage Science Review of Applied Machine Learning*, vol. 4, no. 1, pp. 45–58, 2021.

[5] S. J. Tarsa *et al.*, "Post-silicon cpu adaptation made practical using machine learning," in *Proceedings of the 46th International Symposium on Computer Architecture*, 2019, pp. 14–26. https://doi.org/10.1145/3307650.3322267

[6] E. Fink, "How to Solve It Automatically: Selection Among Problem Solving Methods.," in *AIPS*, 1998, pp. 128–136.

[7] Y. Wang, V. Lee, G.-Y. Wei, and D. Brooks, "Predicting new workload or CPU performance by analyzing public datasets," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 15, no. 4, pp. 1–21, 2019. https://doi.org/10.1145/3284127

[8] T. Wang, S. Ferlin, and M. Chiesa, "Predicting CPU usage for proactive autoscaling," in *Proceedings of the 1st Workshop on Machine Learning and Systems*, 2021, pp. 31–38. https://doi.org/10.1145/3437984.3458831

[9] J. L. Henning, "SPEC CPU2000: Measuring CPU performance in the new millennium," *Computer (Long Beach Calif)*, vol. 33, no. 7, pp. 28–35, 2000. https://doi.org/10.1109/2.869367

[10] M. Boyer, K. Skadron, S. Che, and N. Jayasena, "Load balancing in a changing world: dealing with heterogeneity and performance variability," in *Proceedings of the ACM International Conference on Computing Frontiers*, 2013, pp. 1–10. https://doi.org/10.1145/2482767.2482794

[11] C. Gregg and K. Hazelwood, "Where is the data? Why you cannot debate CPU vs. GPU performance without the answer," in *(IEEE ISPASS) IEEE International Symposium on Performance Analysis of Systems and Software*, IEEE, 2011, pp. 134–144. https://doi.org/10.1109/ISPASS.2011.5762730

[12] M. Walker, S. Bischoff, S. Diestelhorst, G. Merrett, and B. Al-Hashimi, "Hardware-validated CPU performance and energy modelling," in *2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, IEEE, 2018, pp. 44–53. https://doi.org/10.1109/ISPASS.2018.00013

[13] B. L. Peuto and L. J. Shustek, "An instruction timing model of CPU performance," *ACM SIGARCH Computer Architecture News*, vol. 5, no. 7, pp. 165–178, 1977. https://doi.org/10.1145/633615.810667

[14] L. Wang, Y. Huang, X. Chen, and C. Zhang, "Task scheduling of parallel processing in CPU-GPU collaborative environment," in *2008 international conference on computer science and information technology*, IEEE, 2008, pp. 228–232. https://doi.org/10.1109/ICCSIT.2008.27

[15] P. Bellens, J. M. Perez, R. M. Badia, and J. Labarta, "CellSs: a programming model for the Cell BE architecture," in *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, 2006, pp. 86-es. https://doi.org/10.1145/1188455.1188546

[16] E. Buber and D. Banu, "Performance analysis and CPU vs GPU comparison for deep learning," in *2018 6th International Conference on Control Engineering & Information Technology (CEIT)*, IEEE, 2018, pp. 1–6. https://doi.org/10.1109/CEIT.2018.8751930

[17] P. Greenhalgh, "Big. little processing with arm cortex-a15 & cortex-a7," *ARM White paper*, vol. 17, 2011.

[18] X. Zhang, Y. Qu, and L. Xiao, "Improving distributed workload performance by sharing both CPU and memory resources," in *Proceedings 20th*

IEEE International Conference on Distributed Computing Systems, IEEE, 2000, pp. 233–241. https://doi.org/10.1109/ICDCS.2000.840934

[19] X. Liu, "NCPP: Nova CPU Performance Predictor on a Novel Dataset," *arXiv e-prints*, p. arXiv-2407, 2024. https://ui.adsabs.harvard.edu/link_gateway/2024arXiv240703385L/doi:10.48550/arXiv.2407.03385

[20] M. Daraghmeh, A. Agarwal, and Y. Jararweh, "Anomaly Detection-Based Multilevel Ensemble Learning for CPU Prediction in Cloud Data Centers," in *2024 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, IEEE, 2024, pp. 559–564. https://doi.org/10.1109/CCECE59415.2024.10667074

[21] C. Wei-wei, H. Wei, Z. Hai-long, Z. Guo-hui, M. Quan-qi, and H. Peng, "A Processor Performance Prediction Method Based on Interpretable Hierarchical Belief Rule Base and Sensitivity Analysis.," *Computers, Materials & Continua*, vol. 74, no. 3, 2023. DOI: 10.32604/cmc.2023.035743

[22] C. Foots, "Cpu hardware classification and performance prediction using neural networks and statistical learning," *International Journal of Artificial Intelligence and Applications (IJAIA)*, vol. 11, no. 4, 2020. DOI: 10.5121/ijaia.2020.11401 1

[23] V. Vapnik, S. Golowich, and A. Smola, "Support vector method for function approximation, regression estimation and signal processing," *Adv Neural Inf Process Syst*, vol. 9, 1996.

[24] F. Zhang and L. J. O'Donnell, "Support vector regression," in *Machine learning*, Elsevier, 2020, pp. 123–140. https://doi.org/10.1016/B978-0-12-815739-8.00007-9

[25] S. R. Gunn, "Support vector machines for classification and regression," *ISIS technical report*, vol. 14, no. 1, pp. 5–16, 1998. https://cir.nii.ac.jp/crid/1573950399881570304

[26] L.-L. Li, Y.-B. Chang, M.-L. Tseng, J.-Q. Liu, and M. K. Lim, "Wind power prediction using a novel model on wavelet decomposition-support vector machines-improved atomic search algorithm," *J Clean Prod*, vol. 270, p. 121817, 2020. https://doi.org/10.1016/j.jclepro.2020.121817

[27] D. G. Luenberger and Y. Ye, *Linear and non-linear programming*, vol. 2. Springer, 1984. https://doi.org/10.1007/978-3-030-85450-8

[28] J. Ranstam and J. A. Cook, "LASSO regression," *Journal of British Surgery*, vol. 105, no. 10, p. 1348, 2018. https://doi.org/10.1002/bjs.10895

[29] C. Hans, "Bayesian lasso regression," *Biometrika*, vol. 96, no. 4, pp. 835–845, 2009. https://doi.org/10.1093/biomet/asp047

[30] R. Alhamzawi and H. T. M. Ali, "The Bayesian adaptive lasso regression," *Math Biosci*, vol. 303, pp. 75–82, 2018. https://doi.org/10.1016/j.mbs.2018.06.004

[31] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future generation computer systems*, vol. 97, pp. 849–872, 2019. https://doi.org/10.1016/j.future.2019.02.028

[32] V. Dinets, "Apparent coordination and collaboration in cooperatively hunting crocodilians," *Ethol Ecol Evol*, vol. 27, no. 2, pp. 244–250, 2015. https://doi.org/10.1080/03949370.2014.915432