Master's Thesis of Engineering

Lead Time Prediction of Shipbuilding Production Based on Meta-heuristic Optimization within Machine Learning

메타 휴리스틱 최적화를 이용한 기계학습 기반
선박 건조 공정 리드 타임 예측

August 2021

Graduate School of Engineering
Seoul National University
Naval Architecture and Ocean Engineering Major

Haoyu ZHU

Lead Time Prediction of Shipbuilding Production Based on Meta-heuristic Optimization within Machine Learning

Thesis supervisor: Prof. Woo Jong Hun

Submitting a master's thesis of Engineering

August 2021

Graduate School of Engineering
Seoul National University
Naval Architecture and Ocean Engineering Major

Haoyu ZHU

Confirming the master's thesis written by
Haoyu ZHU
August 2021

Chair	[illegible]
Vice Chair	[illegible]
Examiner	[illegible]

Abstract

Lead Time Prediction of Shipbuilding Production Based on Meta-heuristic Optimization within Machine Learning

Haoyu ZHU
Department of Naval Architecture and Ocean Engineering
The Graduate School
Seoul National University

In the shipbuilding industry, each production process has a respective lead time; that is, the duration between start and finish times. Lead time is basic data that is necessary for high-efficiency production planning and systematic production management. Therefore, lead time must be accurate. However, the traditional method of lead time management is not scientific because it mostly makes the plan by calculating the average lead times derived from historical data.

Therefore, to understand the complex relationship between lead time and other influencing factors, this study proposes to use machine learning (ML) algorithms, support vector machine (SVM) and artificial neural network (ANN), which are frequently applied in prediction fields.

Moreover, to improve prediction accuracy, this study proposes to apply meta-heuristic algorithms to optimize the parameters of the ML models. This thesis builds hybrid models, including meta-heuristic-ANN, meta-heuristic-SVM models. In addition, this study compares model's performance with each other.

In searching for the ML model's parameters, the results point out that the new self-organizing hierarchical particle swarm optimization (PSO) with jumping time-varying acceleration coefficients (NHPSO-JTVAC) algorithm is superior in terms of performance. More importantly, the test results demonstrate that the integrated models, based on NHPSO-JTVAC, have the smallest mean absolute percentage error (MAPE) test error in the three shipyard block process data sets, 11.79%, 16.03% and 16.45%, respectively. The results also demonstrate that the built models based on NHPSO-JTVAC can achieve further meaningful enhancements in terms of prediction accuracy.

Overall, the NHPSO–JTVAC-SVM, NHPSO–JTVAC-ANN models are feasible for predicting the lead time in shipbuilding.

Keywords: machine learning; artificial intelligence optimization techniques; support vector machine; artificial neural network; production lead time; hybrid model
Student Number: 2019-25206

Acknowledgements

To my parents and family for their unconditional love and support throughout my education and my future endeavours. To my friends for their continued help and encouragement. To my colleagues in the research laboratories (EnSITE & CASPER) for helping me with Korean and professional knowledge. Lastly, to Prof. Shin and Prof. Woo for their support, guidance, and extensive expertise in the field that allowed me to complete this work.

As the Chinese saying goes, "Receiving drips of water when in need, and I shall return the kindness with a spring" (滴水之恩当以涌泉相报), we must be grateful to those who have helped us. The wonderful and perfect life is a life with gratitude.

Published Content and Contribution

The following published article is included as a part of the thesis with permission.

[1] Zhu, H.; Woo, J.H. Hybrid NHPSO-JTVAC-SVM Model to Predict Production Lead Time. *Applied Sciences* **2021**, 11, 6369.

Contents

Chapter 1 Introduction 1

1.1 Background and Motivation 1

1.2 Related Works 3

1.2.1 Related Works for Lead Time Prediction 3

1.2.2 Related Works for Hybrid Predictive Model 4

1.3 Thesis Organization 6

Chapter 2 Machine Learning 7

2.1 Support Vector Machine 7

2.1.1 Support Vector Machine Algorithm 7

2.1.2 Hyperparameter Optimization for SVM 10

2.2 Artificial Neural Network 11

2.2.1 Artificial Neural Network Algorithm 11

2.2.2 Hyperparameter Optimization for ANN 15

Chapter 3 Meta-heuristic Optimization Algorithms....... 17

3.1 Particle Swarm Optimization 17

3.2 NHPSO-JTVAC: An Advanced Version of PSO 18

3.3 Bat Algorithm 19

3.4 Firefly Algorithm 21

3.5 Grasshopper Optimization Algorithm 22

3.6 Moth Search Algorithm 24

Chapter 4 Hybrid Artificial Intelligence Models 27

4.1 Hybrid Meta-heuristic-SVM Models 27

4.1.1 Hybrid PSO-SVM Model 29

4.1.2 Hybrid NHPSO-JTVAC-SVM Model 30

4.1.3 Hybrid BA-SVM Model 31

4.1.4 Hybrid FA-SVM Model 33
4.1.5 Hybrid GOA-SVM Model 34
4.1.6 Hybrid MSA-SVM Model 35
4.2 Hybrid Meta-heuristic-ANN Models 36
4.2.1 Hybrid PSO-ANN Model 38
4.2.2 Hybrid NHPSO-JTVAC-ANN Model 39
4.2.3 Hybrid BA-ANN Model 40
4.2.4 Hybrid FA-ANN Model 41
4.2.5 Hybrid GOA-ANN Model 42
4.2.6 Hybrid MSA-ANN Model 43

Chapter 5 Lead Time Prediction Based on Hybrid AI Models 44
5.1 Data and Preparation 44
5.1.1 Data Normalization 45
5.1.2 Feature Selection 45
5.2 Lead Time Prediction 46
5.3 Performance Metrics 47

Chapter 6 Experimental Results 49
6.1 Results Based on Hybrid SVM-based Models 49
6.2 Results Based on Hybrid ANN-based Models 55
6.3 Overall Results 60

Chapter 7 Conclusions and Future Works 62

Bibliography 63

Appendix A 68

Abstract in Korean 69

List of Tables

Table 1.1 The research literature on SVM optimization techniques. 5

Table 1.2 The research literature on ANN optimization techniques. 5

Table 2.1 The frequently used kernel functions of SVM. 9

Table 4.1 Parameter setting of PSO 29

Table 4.2 Pseudocode of the PSO-SVM 30

Table 4.3 Parameter setting of NHPSO-JTVAC 30

Table 4.4 Pseudocode of the NHPSO-JTVAC-SVM 31

Table 4.5 Parameter setting of BA 32

Table 4.6 Pseudocode of the BA-SVM 32

Table 4.7 Parameter setting of FA 33

Table 4.8 Pseudocode of the FA-SVM...33

Table 4.9 Parameter setting of GOA 34

Table 4.10 Pseudocode of the GOA-SVM 34

Table 4.11 Parameter setting of MSA 35

Table 4.12 Pseudocode of the MSA-SVM...35

Table 4.13 Pseudocode of the PSO-ANN 38

Table 4.14 Pseudocode of the NHPSO-JTVAC-ANN 39

Table 4.15 Pseudocode of the BA-ANN..40

Table 4.16 Pseudocode of the FA-ANN...41

Table 4.17 Pseudocode of the GOA-ANN...42

Table 4.18 Pseudocode of the MSA-ANN...43

Table 5.1 Data collection...44

Table 5.2 Regression prediction performance metrics 47

Table 5.3 Interpretation of MAPE values 48

Table 6.1 K-fold CV scores of each (meta-heuristic)-SVM model.............51

Table 6.2 Optimal values of the three SVM parameters ($\boldsymbol{C}$, $\boldsymbol{\varepsilon}$, and $\boldsymbol{\sigma}$)......54

Table 6.3 Test errors of each (meta-heuristic)-SVM model........................54

Table 6.4 Average MAPE of SVM, PSO-SVM, NHPSO-JTVAC-SVM, BA-SVM, FA-SVM, GOA-SVM, and MSA-SVM 55

Table 6.5 Test errors of each (meta-heuristic)-ANN model........................59
Table 6.6 Average MAPE of ANN, PSO-ANN, NHPSO-JTVAC-ANN, BA-ANN, FA-ANN, GOA-ANN, and MSA-ANN 60

List of Figures

Figure 1.1 (a) Planned lead time shorter than actual lead time; (b) Planned lead time longer than actual lead time..........2
Figure 1.2 Differences in planned and actual lead times..........2
Figure 2.1 Influence of hyperparameters on SVM regression..........11
Figure 2.2 Structure of ANN with one hidden layer..........12
Figure 2.3 Activation functions of neural network..........13
Figure 2.4 Conceptual graph of back-propagation process..........14
Figure 2.5 Example graph of gradient descent with different start position..16
Figure 4.1 Flow chart of mate-heuristic optimization within SVM..........28
Figure 4.2 Concept of k-fold cross-validation (k=5)..........29
Figure 4.3 Flow chart of mate-heuristic optimization within ANN..........37
Figure 5.1 Conceptual graph of proposed production lead time management method..........44
Figure 5.2 Flow chart of feature selection..........46
Figure 5.3 Flow chart of the prediction models..........46
Figure 6.1 Generation of the optimization SVM models in block assembly process performance dataset..........49
Figure 6.2 Generation of the optimization SVM models in block pre-outfitting process performance dataset..........50
Figure 6.3 Generation of the optimization SVM models in block pre-painting process performance dataset..........50
Figure 6.4 Test MAPE of each (meta-heuristic)-SVM model in block assembly process performance dataset..........52
Figure 6.5 Test MAPE of each (meta-heuristic)-SVM model in block pre-outfitting process performance dataset..........53
Figure 6.6 Test MAPE of each (meta-heuristic)-SVM model in block pre-painting process performance dataset..........53
Figure 6.7 Generation of the optimization ANN models in block assembly process performance dataset..........55
Figure 6.8 Generation of the optimization ANN models in block pre-outfitting

process performance dataset..56
Figure 6.9 Generation of the optimization ANN models in block pre-painting process performance dataset..56
Figure 6.10 Test MAPE of each (meta-heuristic)-ANN model in block assembly process performance dataset.......................................58
Figure 6.11 Test MAPE of each (meta-heuristic)-ANN model in block pre-outfitting process performance dataset.......................................58
Figure 6.12 Test MAPE of each (meta-heuristic)-ANN model in block pre-painting process performance dataset..59
Figure 6.13 Comparison of test MAPE between NHPSO-JTVAC-SVM and NHPSO-JTVAC-ANN..60
Figure A.1 Flow chart of NHPSO-JTVAC within SVM 68
Figure A.2 Sketch map of NHPSO-JTVAC within ANN..............................68

Abbreviations

PPS	*Production planning and scheduling*
AI	*Artificial intelligence*
ML	*Machine learning*
NLP	*Natural language processing*
SR	*Speech recognition*
SVM	*Support vector machine*
SRM	*Structural risk minimization*
QP	*Quadratic programming*
RBF	*Radial basis function*
ERM	*Empirical risk minimization*
MCM	*Model complexity minimization*
SVs	*Support vectors*
ANN	*Artificial neural network*
GD	*Gradient descent*
ReLU	*Rectified linear unit*
BPNN	*Back-propagation neural network*
RF	*Random forest*
PSO	*Particle swarm optimization*
LDIW	*Linearly-decreasing inertia weight*
HPSO–JTVAC	*Self-organizing hierarchical PSO with jumping time-varying acceleration coefficients*
NHPSO–JTVAC	*New self-organizing hierarchical PSO with jumping time-varying acceleration coefficients*
GS	*Grid search*
FA	*Firefly algorithm*
BA	*Bat algorithm*
GOA	*Grasshopper optimization algorithm*
MSA	*Moth search algorithm*
FS	*Feature selection*
CV	*Cross-validation*
RMSE	*Root mean square error*
MAE	*Mean absolute error*
MAPE	*Mean absolute percentage error*

Chapter 1. Introduction

1.1 Background and Motivation

In the shipbuilding industry, an essential part of scientific management is lead time, which is necessary for shipyards to arrange production plans, particularly in production organization and progress of production control [1,2]. Adequate lead-time management (for example, in the production planning stage, if the planned lead time can be as close as possible to the actual lead time) can shorten the construction period and create conditions for controlling the construction process. Furthermore, a scientific and reasonable lead time plan can effectively carry out construction man hour analysis, balance workload, and establish a planning control system to satisfy production demand and give full play to construction capacity [3]. But, if the evaluation of construction workload is insufficient (Fingure1.1a), and management did not prepare the necessary plan for construction peaks before, the construction cycle will be prolonged. Here, to not affect the construction cycle, the construction workers have to work overtime for a long time, resulting in the decline of construction quality. Conversely, if the construction workloads were overestimated (Fingure1.1b), in this case, it would lead to excess construction capacity problems and waste human resources [4]. Therefore, lead time management is not only the basis for balancing production lines and shortening the construction period but also a prerequisite for achieving efficient production planning and systematic production management.

However, due to the customers' unique demands, the rapid change of requirements, and the diversity of the basic unit blocks of the hull, the shipbuilding industry has a certain degree of uncertainty and variability [5]. It is exactly because of the high degree of uncertainty and variability in the process of shipbuilding. There is a significant difference between the planned lead time and the actual lead time (Figure 1.2), which leads to the decline of the efficiency of the lead time basic information management system. For a long time, shipyard managers often used the experience assessment strategy to specify lead time [6,7]. Be that as it may, this strategy is still time-consuming. At present, managers have used quarterly feedback or revision of annual business plans to overcome this problem, but these methods are inefficient, even wasting a lot of time. Many shipyards have also tried using other measures to predict and manage lead time. For example, D shipyard's pre-outfitting block process used the traditional regression method to predict the lead time, but there are still many defects, such as low accuracy. In this way, production planning and scheduling (PPS) can't be organized as expected, making the shipyard's management ineffective [7].

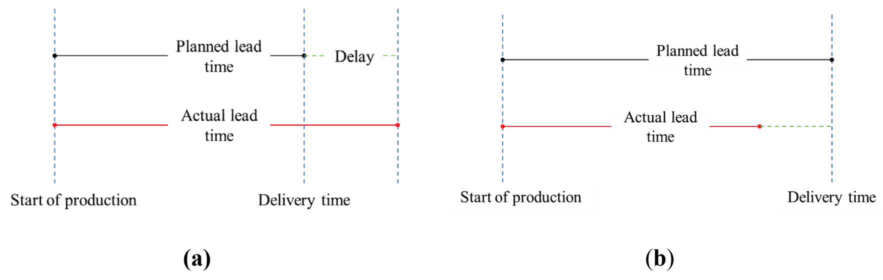


Figure 1.1 **(a)** Planned lead time shorter than actual lead time; **(b)** Planned lead time longer than actual lead time.

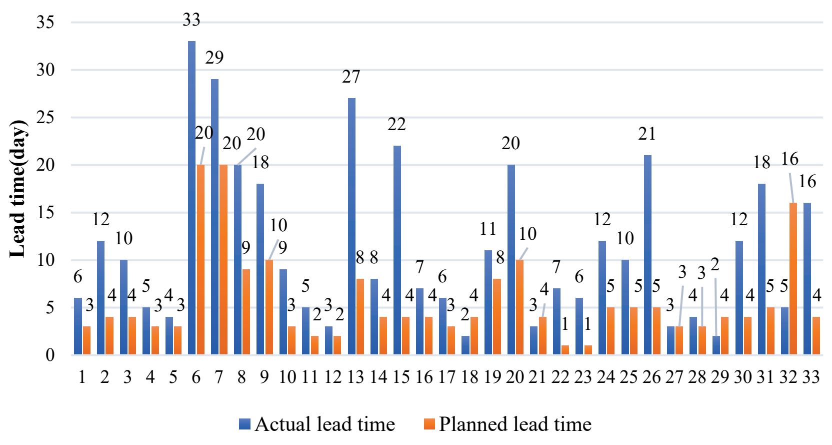


Figure 1.2 Differences in planned and actual lead times.

1.2 Related Works

1.2.1 Related Works for Lead Time Prediction

In the current research, machine learning (ML) algorithms have been widely used in various lead time management fields. Using ML can analyze the construction works' data, provide effective insights, and guide the project managers during decision-making phases [8].

In order to understand the complex relationship between lead time and its affecting factors, several researchers have studied the prediction of lead time by ML, and some results have been achieved [7,9,10]. Gyulai et al. [9] suggested applying ML models to predict the lead time of jobs in the manufacturing flow-shop environment; the results showed that ML algorithms could satisfactory understand the non-linear relationship between lead time and other factors which affect lead time, and they pointed out that ML regression models can provide good prediction accuracy. In the semiconductor manufacturing industry, Lingitz et al. [7] paid attention to do feature engineering in the lead time data preprocessing process. They calculated importance scores for each feature which is affecting lead time. Moreover, they selected the key features to develop the lead time prediction models. Specifically, to improve production planning efficiency in the shipbuilding field, Jeong et al. [10] attempted to predict the lead time of spool fabrication and painting processes. They applied ML models and compared the learning performance of each model.

1.2.2 Related Works for Hybrid Predictive Model

In ML, support vector machine (SVM) and artificial neural network (ANN) are the most popular in the prediction field [11]. However, their disadvantage is that they are too sensitive to parameters. Also, an efficient ML model can only be built after its parameters are carefully selected [12,13]. Therefore, many researchers have proposed optimizing parameters of SVM and ANN (Table 1.1& Table 1.2) [14-24].

For instance, in the SVM parameters optimization methods, Yu & Cai [14] suggested using the particle swarm optimization (PSO)-SVM hybrid model to predict man-hours in aircraft assembly. The prediction results showed that the PSO-SVM hybrid model was considerably better than the back-propagation neural network (BPNN) model. Wan & Fang [15] combined an SVM model with a PSO algorithm to predict the risk of the expressway project. The forecasting results indicated that the PSO-SVM model was more accurate and better than the traditional SVM model. Lv et al. [16] developed PSO-SVM and grid search (GS)-SVM models to predict steel corrosion. The prediction results demonstrated the PSO-SVM regression model was more accurate compared with the GS-SVM model.

Furthermore, in the compressive strength prediction field, the firefly algorithm (FA)-SVM model was proposed by Pham et al. [17]. The results indicated that FA-SVM has achieved the most desirable performance with low prediction errors and can be a valuable tool to predict compressive strength. Similarly, Sahoo et al. [18] proposed using a FA to optimize an SVM model in the river flood prediction field. The results showed that the FA is an excellent optimization algorithm to improve the prediction accuracy of an SVM. Moreover, other researchers combined other meta-heuristic algorithms such as the bat algorithm (BA) and the grasshopper optimization algorithm (GOA) with

SVMs and obtained good results [19,20].

	SVM Optimization Techniques	**Prediction Field**
Yu & Cai [14] Wan & Fang [15] Lv et al. [16]	PSO	1. Man-hours in aircraft assembly 2. Risk of the expressway project 3. Steel corrosion
Pham et al. [17] Sahoo et al. [18]	FA	1. Compressive strength of High-Performance Concrete 2. River flood prediction
Tavakkoli et al. [19]	BA	Printed circuit board sales
Barman & Choudhury [20]	GOA	Short-term load

Table 1.1 The research literature on SVM optimization techniques.

In the ANN parameters optimization techniques, likewise, Mohamad et al. [21] combined an ANN model with a PSO algorithm to predict the rock strength. The prediction results indicated that the PSO-ANN model can predict the unconfined compressive strength accurately. Mostafaeipour et al. [22] attempted to develop the hybrid FA-ANN model to predict air travel demand. The overall results showed that the ANN with a FA approach had presented the highest rate of adaptation between predicted and real data.

In addition, Hussin et al. [23] used a BA-ANN model to predict electricity production and water consumption. Moayedi et al. [24] developed a GOA-ANN model to predict soil compression coefficient. The results all show the excellent predictive performance of the proposed integrated models.

	ANN Optimization Techniques	**Prediction Field**
Mohamad et al. [21]	PSO	Rock strength
Mostafaeipour et al. [22]	FA	Air travel demand
Hussin et al. [23]	BA	Electricity production and water consumption
Moayedi et al. [24]	GOA	Soil compression coefficient

Table 1.2 The research literature on ANN optimization techniques.

This thesis adopts the meta-heuristic optimization approaches (such as PSO, FA, BA, GOA) proposed by previous research to optimize the parameters of SVM and ANN. And on top of that, two new algorithms have been added. One is the moth search algorithm (MSA), and the other one is an enhanced version of PSO: new self-organizing hierarchical PSO with jumping time-varying acceleration coefficients (NHPSO-JTVAC).

Overall, firstly, this study uses six meta-heuristic algorithms to optimize the parameters of SVM and ANN. Next, the hybrid SVM-based, ANN-based model is built to predict the lead time in the shipyard's block assembly, pre-painting and pre-outfitting processes. Finally, the performance of the hybrid models is compared with each other to select the suitable forecasting model.

1.3 Thesis Organization

The remainder of this thesis is organized as follows: Chapter 2 introduces the algorithm principle of the ML (SVM & ANN). Chapter 3 presents the optimization principle of the meta-heuristic algorithm. Chapter 4 introduces the algorithm principle of the hybrid predictive model. Chapter 5 describes the construction of the model. Chapter 6 discusses the experimental results, and Chapter 7 summarizes and concludes the thesis.

Chapter 2. Machine Learning

2.1 Support Vector Machine

2.1.1 Support Vector Machine Algorithm

As the traditional ML algorithm, SVM was developed by Cortes & Vapnik [25] in 1995. It was used for classification problems such as pattern recognition for the first time. After that, it was extended to non-linear regression, which has shown a good effect. SVM has a solid theoretical foundation and good generalization ability as one of the most popular algorithms in ML algorithms. Also, it has unique advantages for small sample data and non-linear prediction. Moreover, SVM has made a series of successful attempts in the time forecast, and it has become a popular research topic in the field of industrial forecasting. Thissen et al. [26] applied the SVM model to predict the time series. They demonstrated that the SVM model has a good performance in time series forecasting. Zhang [27] proposed using the SVM model to forecast the short-term load of the electric power system, and they showed that the forecast performance of the SVM model was better than BPNN. Astudillo et al. [28] used an SVM model to predict copper prices. The results indicated that the SVM model can predict copper-price volatilities near reality.

SVM algorithm is based on the principle of structural risk minimization (SRM). To let the SVM algorithm converge to the global optimal solution, it converted the situation into a quadratic programming (QP) problem and inlet the kernel function method, which maps the original training data from low-dimensional space to a high-dimensional space. So, an SVM can theoretically converge to the global optimal solution of a problem [29].

For non-linear regression problems, suppose the training set of data is $\{(x_1, y_1), (x_2, y_2), \cdots (x_l, y_l)\}, i = 1,2,3, \ldots, l, x_i \subset R^n, y_i \subset R$ where x_i is an input value, y_i is a target value of output and l is the total number of training set. The non-linear regression idea of SVM is to find a non-linear mapping by kernel function from input to output and map the training set to a high-dimensional feature space, where the training set can be regressed through the regression equation $f(x)$ in this feature space. $f(x)$ is defined as Equation 2.1 [30]:

$$f(x) = w \cdot \varphi(x) + b \tag{2.1}$$

where $w \subset R^n$ and $b \subset R$ represent the weight vector, threshold vector. φ denotes a non-linear mapping to high-dimensional space.

To find the optimum values of w and b, by SRM principle, the problem can be transformed to minimize the regularized risk function (2.2) [31]:

$$R_{reg}(f) = \frac{1}{2}\|w\|^2 + C \ \cdot R_{emp}(f) = \frac{1}{2}\|w\|^2 + C\frac{1}{l}\sum_{i=1}^{l} L_\varepsilon(y_i, f(x_i)) \tag{2.2}$$

$$L_\varepsilon(y_i, f(x_i)) = \begin{matrix} |y_i - f(x_i)| - \varepsilon\,, for\ |y_i - f(x_i)| \geq \varepsilon & (2.3) \\ 0, otherwise & (2.4) \end{matrix}$$

where the first term $\frac{1}{2}\|w\|^2$ of the objective function is regularized term that represents the generalization ability of the regression function, the second term $C\frac{1}{l}\sum_{i=1}^{l} L_\varepsilon(y_i, f(x_i))$ is referred to as empirical error, L_ε denotes an ε-insensitive loss function defined by Equations (2.3 & 2.4) [32], ε is called as a tube width. Moreover, C is a constant called the regularization parameter.

Next, SVM introduced "slack variables" (ξ_i & ξ_i^*) to avoid the model being too complicated and reduce the model's sensitivity of noise points [33,34].

minimize:
$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \tag{2.5}$$

$$\textbf{subject to:} \quad \begin{cases} y_i - (w \cdot \varphi(x_i)) - b \leq \varepsilon + \xi_i \\ (w \cdot \varphi(x_i)) + b - y_i \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0, \qquad i = 1,2,3 \ldots l. \end{cases} \tag{2.6}$$

Finally, to facilitate the introduction of the kernel function and simplify the process of the problem, the SVM problem can be transformed into a dual optimization problem by using Lagrange multipliers (α_i, α_i^*) [35]:

$$\textbf{maximize:} \quad \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)K(x_i, x_j) \\ -\varepsilon\sum_{i=1}^{l}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{l} y_i(\alpha_i - \alpha_i^*) \end{cases} \tag{2.7}$$

$$\textbf{subject to:} \quad \sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0, \qquad 0 \leq \alpha_i^* \leq C, i = 1,2,3 \ldots l. \tag{2.8}$$

After this series of derivations, the final regression function of SVM can be obtained as [36]:

$$f(x) = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)\, K(x_i, x) + b \tag{2.9}$$

In (2.9), $K(x_i, x)$ is the kernel function of the SVM model. Frequently used kernel functions include linear kernel function, radial basis function (RBF) kernel function, polynomial kernel function, shown in Table 2.1.

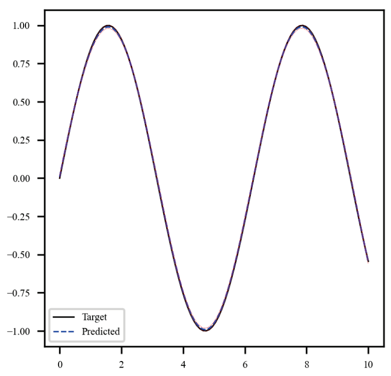
Kernel	**Equation**
Linear	$K(x, y) = (x \cdot y)$
RBF	$K(x, y) = exp\{-\lVert x - y \rVert^2 / 2\sigma^2\}$
Polynomial	$K(x, y) = (x \cdot y + 1)^d$

Table 2.1 The frequently used kernel functions of SVM.

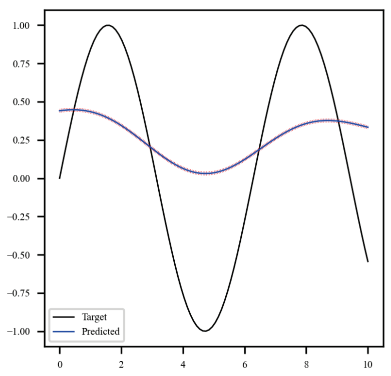
Among them, the performance of the RBF kernel function to handle complex samples on high-dimensional is better than the linear kernel function [37,38]. Moreover, in some cases, compared with the polynomial kernel function, the prediction accuracy of the model which used an RBF kernel is significantly high [39]. So far RBF kernel is the best choice for practical applications [40]. Therefore, this study chose the RBF kernel function to solve the problems.

2.1.2 Hyperparameter Optimization for SVM

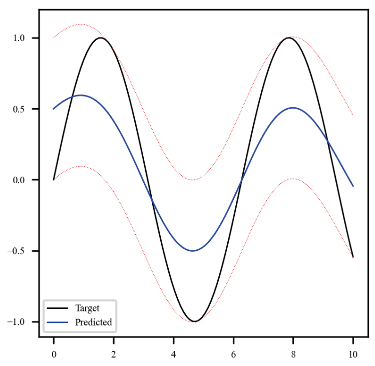
As mentioned above, the prediction model of SVM is susceptible to hyperparameters' selection process. Also, the prediction performance of SVM strongly depends on whether the hyperparameters were selected reasonably [41]. The hyperparameters affect the prediction performance of SVM in different aspects. It mainly includes the regularization parameter C, the coefficient of the kernel function σ (once an RBF kernel has been chosen), and the tube width ε. The regularization parameter C represents a balance between model accuracy and complexity. If the value of C is too large or too small, it will lead to an imbalance between empirical risk minimization (ERM) and model complexity minimization (MCM); σ controls the size of the RBF kernel. The larger the σ value is, the more closely the SVM will fit the data. ε determines the width of the ε-insensitive zone and its value affects number of support vectors (SVs) employed to the regression. Likewise, it is crucial to choose an optimal value of ε. Figure 2.1 demonstrates the influence of each parameter on the SVM regression. In summary, selecting the three most appropriate hyperparameters can significantly improve the performance of the model. Therefore, this thesis introduced the meta-heuristic algorithms (Chapter 3) to optimize the parameters (C, σ, ε) of the SVM model.



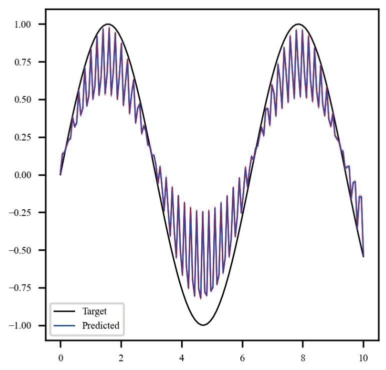
(**a**) SVM with properly selection parameters



(**b**) SVM with a poor selection of C



(c) SVM with a poor selection of ε



(d) SVM with a poor selection of σ

Figure 2.1 Influence of hyperparameters on SVM regression.

2.2 Artificial Neural Network

2.2.1 Artificial Neural Network Algorithm

ANN is a computational model based on the inspiration of the human brain to transmit information through neurons [42]. Its structure comprises the input layer, hidden layer(s), and output layer [43]; Figure 2.2 illustrates the most basic and frequently used three-layer neural network structure. In addition, ANN has had many breakthrough successes in computer vision, natural language processing (NLP), speech recognition (SR), etc. It is hotspot research in the field of artificial intelligence (AI).

In the ANN's training process, the number of nodes in the input layer and output layer can be determined by input data, output data (target value); Moreover, the number of hidden layers and the number of nodes in the hidden layer can be established through experiments. After ANN's structure was fixed, the neural network's training will be performed on the input and output training sets. This series of processes is the learning of the neural network.

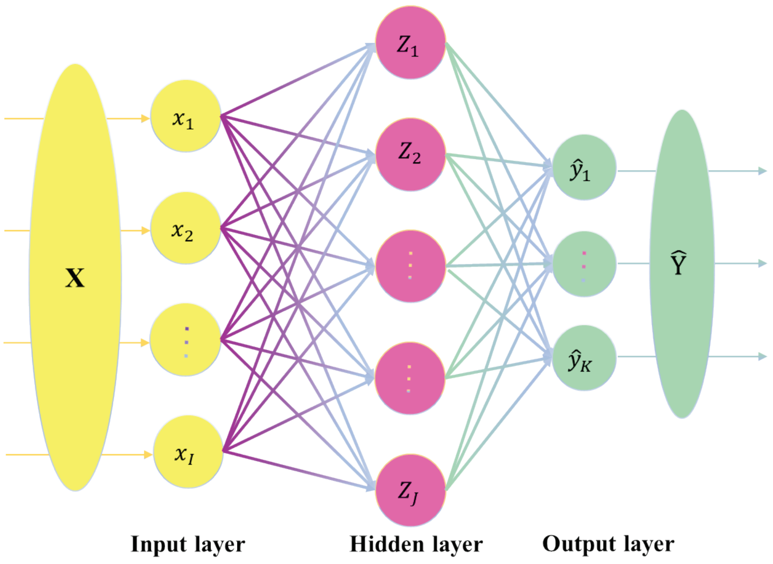


Figure 2.2 Structure of ANN with one hidden layer.

The learning process of ANN's training consists of two processes: forward and backward propagation.

(I) Forward propagation can be defined as the following formulas [44]:

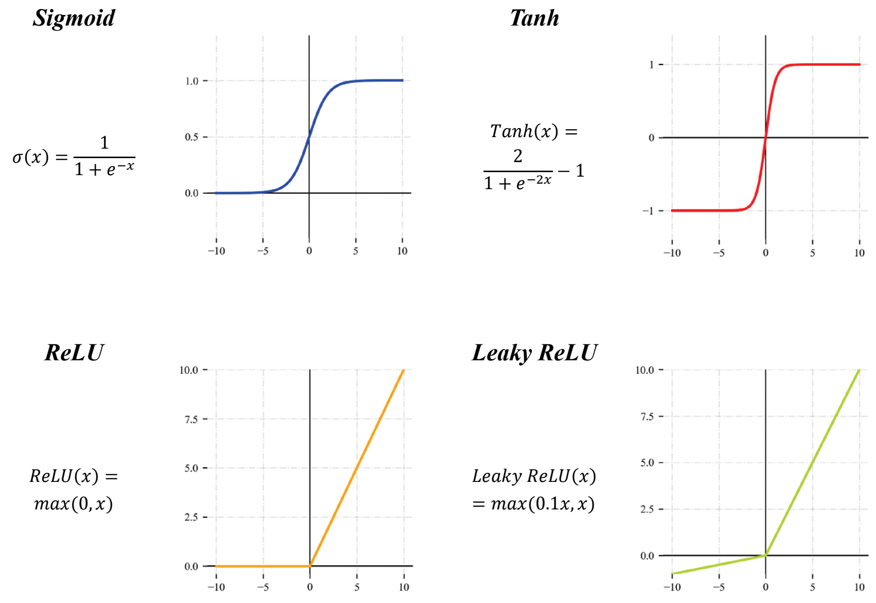
$$Z_j = f_H\left(\sum_{i=1}^{I} w_{ij}x_i + b_j\right), j = 1,2,\ldots,J \tag{2.10}$$

$$y_k = f_O\left(\sum_{j=1}^{J} w_{jk}Z_j + b_k\right), k = 1,2,\ldots,K \tag{2.11}$$

where Z_j is the output of the hidden layer, y_k is the output of the output layer. w_{ij} is the connection weight between input layer's node i and hidden layer's node j, w_{jk} is the connection weight between the hidden layer's node j and the output layer's node k. Furthermore, b_j is a threshold of hidden layer's node j, b_k is a threshold of the output layer's node k. Besides, f_H represents an activation function of the hidden layer and f_O denotes an activation function of the output layer. For regression problems, the activation function is not used in the output layer, and the formula (2.11) can be transformed into:

$$y_k = \sum_{j=1}^{J} w_{jk} Z_j + b_k, k = 1,2,\ldots,K. \tag{2.12}$$

Furthermore, a part of the neural network's activation functions is shown in Figure 2.3. A rectified linear unit (ReLU) activation function is one of the most important activation functions frequently used for hidden layer(s) in training the neural network [45]. It is simple to implement and converges faster. Therefore, this thesis selected the ReLU activation function to training the neural network.



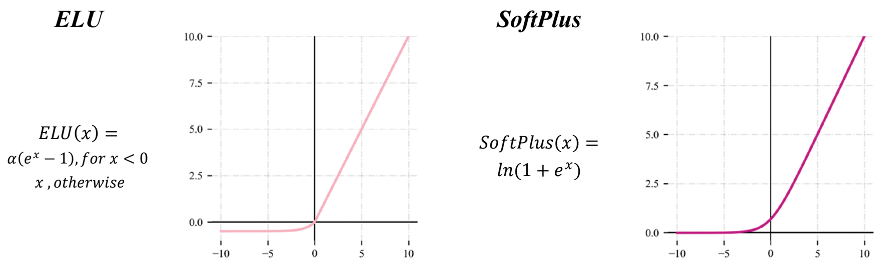


Figure 2.3 Activation functions of neural network.

(II) Backward propagation:

As the widely used algorithm in deep learning, the backwards propagation algorithm is a general method for ANN's training process [46]. It works by using gradient descent (GD) to minimize errors in target values(labels) and the model's predicted value. Figure 2.4 demonstrates the backward propagation process in the three layers neural network.

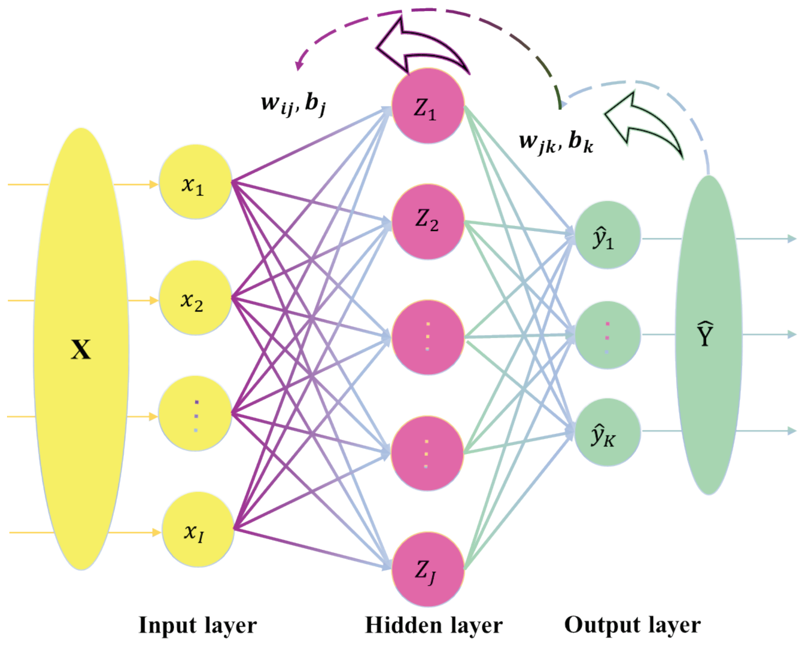


Figure 2.4 Conceptual graph of back-propagation process.

When the predicted values of the neural network's output layer did not match the target values (labels), the error computed by the loss function would be propagated back from the output layer to the hidden layer(s) and the input layer. It means each layer’s connection weights and thresholds would be updated(adjusted) according to the GD method. For example, the neural network’s loss function can be defined in Equation 2.13 [47]:

$$J(w,b) = \frac{1}{2}\sum_{k=1}^{K}(y_k - \hat{y}_k)^2, \quad k = 1,2,\dots,K \tag{2.13}$$

Among them: w, b represent the neural network’s weight, threshold. $\hat{y}_k$ is the target value(label) of the output layer’s node k. w and b can be adjusted by GD in the following formulas (2.14& 2.15) [48]:

$$w^{new} = w^{old} - \eta\frac{\partial\big(J(w,b)\big)}{\partial(w^{old})} \tag{2.14}$$

$$b^{new} = b^{old} - \eta\frac{\partial\big(J(w,b)\big)}{\partial(b^{old})} \tag{2.15}$$

where η denotes the learning rate of the neural network, which has a small positive value between 0 and 1.

2.2.2 Hyperparameter Optimization for ANN

As a very general optimization algorithm, GD is an important part of the neural network. The central idea of GD is to minimize the loss function by updating the neural network’s connection weights and thresholds with iteration.

However, the initial weight and thresholds of the model are significant for the training of the network [49]. Poor initialization parameters (start position) selection will cause problems with GD, for example, reduce the training speed and stuck in a saddle point or local solution easily (Figure 2.5); suitable initialization parameters can accelerate the training convergence speed, and it is more likely to find an optimal solution.

In this thesis, likewise, the meta-heuristic algorithms (Chapter 3) are introduced and proposed to optimize the initial weights and thresholds of the ANN model.

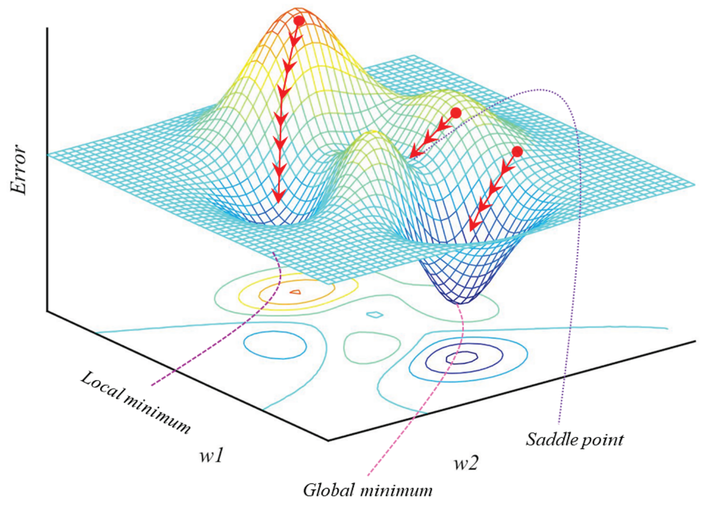


Figure 2.5 Example graph of gradient descent with different start positions.

Chapter 3. Meta-heuristic Optimization Algorithms

3.1 Particle Swarm Optimization

In 1995, inspired by the flocking behaviour of birds, a PSO algorithm was introduced and developed by Kennedy & Eberhart [50]. The basic idea of PSO is that birds' flocking behaviour can find the optimal destination through collective information sharing. Certainly, PSO has successfully solved many complex optimization problems. In PSO, the potential solving scheme of an optimization problem is each bird called particle in the exploring space. The particle has a fitness value decided by the objective function and a speed that determines the direction and range of its movement. Then, the particles will do a search following the current optimal particle in the solution space. In each iteration, the particle updates the velocity and positions by two factors. The first factor is the best-known position called the personal best (*p-best*) found by each particle. The other factor is the best-known position called global best (*g-best*) found by the entire particle swarm.

Suppose that swarm consists of m particles in a D-dimensional search space, the position of the $i\text{-}th$ particle can be represented as a vector $x_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,d})$, where $i = 1,2,3,\cdots,m, d = 1,2,3,\cdots,D$. The fitness value can be calculated by bringing x_i into the objective function, and the particle's performance can be evaluated according to the fitness value. Likewise, the *i-th* particle's velocity vector can be expressed as $v_i = (v_{i,1}, v_{i,2}, \cdots, v_{i,d})$, the best position for its search can be described as $p_i = (p_{i,1}, p_{i,2}, \cdots, p_{i,d})$, and $p_g = (p_{g,1}, p_{g,2}, \cdots, p_{g,d})$ indicates the best solution obtained by the whole particle swarm. According to the following formulas (3.1&3.2), each particle updates its velocity and position to search for the optimal position.

$$v_{i,d}^{new} = \omega v_{i,d}^{old} + c_1 r_1 (p_{i,d} - x_{i,d}) + c_2 r_2 (p_{g,d} - x_{i,d}) \tag{3.1}$$

$$x_{i,d}^{new} = x_{i,d}^{old} + v_{i,d}^{new} \tag{3.2}$$

where $i = 1,2,3,\cdots,m, d = 1,2,3,\cdots,D$. ω is defined as the inertia weight used to control the influence of the previous velocity on the current velocity. c_1 and c_2 are learning factors, also known as acceleration constants. r_1 and r_2 are uniform random numbers in the range of [0,1]. The inertia weight ω (non-negative) controls the current velocity of the particle. The particle's velocity will be greater with the increasing value of ω, and the particle will perform a global search with a more significant step size. Conversely, the particle tends to perform a more finely local search with smaller values of ω. In addition, ω is frequently assumed as a dynamic value to make the global and local search capabilities balance. The typical way to determine the value of ω is the linearly-decreasing inertia weight (LDIW) strategy [51].

$$\omega = \omega_{max} - (\omega_{max} - \omega_{min}) \times \frac{Iter}{Iter_{max}} \tag{3.3}$$

where $Iter$ represents the current number of iterations, and $Iter_{max}$ means the maximum number of iterations, and ω_{max} and ω_{min} are maximal/minimal inertia weights, frequently set to 0.9 and 0.4, respectively [52].

3.2 NHPSO-JTVAC: An Advanced Version of PSO

Although the convergence speed of the PSO algorithm is fast, it still falls into a local optimum easily in most complex optimization problems. So, it is uncertain whether the PSO algorithm can find an optimal solution [52].

To solve the above problems, the self-organizing hierarchical PSO with time-varying acceleration coefficients (HPSO-TVAC) was proposed by Ratnaweera et al. [53]. HPSO-TVAC is an efficient improved algorithm of the classic PSO algorithm. After that, Ghasemi et al. [54] proposed an enhanced version of HPSO-TVAC called NHPSO-JTVAC, which has better performance than the original HPSO-TVAC algorithm. In order to avoid particles getting stuck with local optimal, they are afforded the ability to suddenly jump out during the algorithm iteration according to Equations (3.4) – (3.6):

$$c^{Iter} = \left(c_f - c_i\right) \times \frac{Iter}{Iter_{max}} + c_i \tag{3.4}$$

$$c_1^{Iter} = |w|^{(c^{Iter} \times w)} \tag{3.5}$$

$$c_2^{Iter} = |1 - w|^{(c^{Iter}/(1-w))} \tag{3.6}$$

where c^{Iter} changes from $c^1 = c_i = 0.5$ to $c^{Iter_{max}} = c_f = 0$ and w is defined as a standard normal random value. Unlike Equation (3.1), the new search equation is given as:

$$\begin{aligned} v_{i,d}^{Iter+1} = c_1^{Iter} \times r_{1,d} \times \left(p_{i,d}^{Iter} - x_{i,d}^{Iter}\right) + c_2^{Iter} \times r_{2,d} \\ \times \left(\left(p_{g,d}^{Iter} + p_{r,d}^{Iter}\right) - 2 \times x_{i,d}^{Iter}\right) \end{aligned} \tag{3.7}$$

where $p_{r,d}^{Iter}$ represents the best personal solution of a randomly selected particle (such as the *r-th* particle).

3.3 Bat Algorithm

BA is an optimization algorithm based on swarm intelligence, inspired by the echolocation behaviour of microbats. It was proposed by Yang [55] in 2010.

In the BA, the solution of each optimization problem is in the search space. The search method of the BA is similar to the PSO. Each bat has a fitness value determined by an optimization problem. Moreover, each bat updates the velocity and position following the current optimal bat position by adjusting the pulse frequency, loudness, and pulse rate.

The algorithm is based on the following basic assumptions:

(1) Each bat can sense distance by using echolocation, and they can perceive the difference between food and background barriers.

(2) To search for the target, bats fly randomly at position x_i with speed v_i with a fixed frequency f_{min}, varying wavelength λ and loudness

A_0. They adjust the wavelength (or frequency) of their emitted pulses according to the distance between the target and themselves. Also, they change the rate of pulse emission $r \in [0,1]$ determined by the proximity of their prey.

(3) Assume that the loudness changes from the maximum (positive) value A_0 to the minimum constant value A_{min}.

For BA, firstly, each bat calculates the frequency f before updating its velocity and position:

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{3.8}$$

where f_{min} and f_{max} indicate the maximum and minimum frequency separately. $\beta \in[0,1]$ is a random vector that obeys the uniform distribution.

Next, each bat updates the velocity and position to reach the optimal position according to the following formulas:

$$v_i^t = v_i^{t-1} + \left(x_i^t - x_*\right)f_i \tag{3.9}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{3.10}$$

Among them, x_* is the current global best position, which is confirmed after comparing all the solutions of all bats. Additionally, each bat's initial frequency is randomly determined by uniform values in the range of $[f_{min}, f_{max}]$.

In local search, the position update formula for each bat is as follows:

$$x_{new} = x_{old} + \varepsilon A^t \tag{3.11}$$

where ε is a uniform random number within [-1,1], A^t is the average of the loudness of all bats in the current generation.

Finally, the loudness and pulse emission rate are also updated along with the generation process. The formula for updating the loudness and pulse emission rate is as follows:

$$A_i^{t+1} = \alpha A_i^t \tag{3.12}$$

$$r_i^t = r_i^0[1 - \exp(-\gamma t)] \tag{3.13}$$

where α and γ are constants, each bat's loudness and pulse emission rate are randomly initialized individually. In general, we define that the initial loudness is between [1, 2], and the initial pulse emission rate is usually around zero [56]. Thus, the loudness and emissivity of the bat will be continuously updated along with the search process to fly to the optimal solution gradually.

3.4 Firefly Algorithm

As an intelligent stochastic algorithm for solving global optimization problems, FA is on the basis of the flash behaviour of fireflies. [57].

In FA, all fireflies are thought to be hermaphroditic, attracting each other regardless of gender. The algorithm is based on two key concepts: the intensity of the light emitted and the attraction between two fireflies.

To imitate the scintillation display of fireflies, the three rules were proposed as follows:

(1) Assuming that all fireflies are hermaphroditic, one firefly may be attracted to any other fireflies.

(2) The luminosity of fireflies determines their attraction. The brighter ones will attract the darker ones. If there is no firefly brighter than the one under consideration, it will move randomly.

(3) The optimal value of the function is proportional to the brightness of the firefly.

Firstly, the relative light intensity of fireflies can be defined as follows:

$$I(r) = I_0 e^{-\gamma {r_{ij}}^2} \tag{3.14}$$

where I_0 is the original light intensity at a distance $r = 0$. γ is the absorption coefficient that can be set as a constant because the fluorescence will be weakened by increasing distance and media absorption. r is the distance between two fireflies i and j.

Next, the following formula can be expressed as the attraction between fireflies：

$$\beta = \beta_0 e^{(-\gamma \times r_{ij}{}^2)} \tag{3.15}$$

where β_0 is the firefly attractiveness value at $r = 0$.

In addition, the distance between two fireflies can be defined as Cartesian Distance (3.16):

$$r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2} \tag{3.16}$$

where d is the dimension of the space, x_i, x_j are the spatial positions of firefly i and j.

Lastly, the firefly will move towards the brighter firefly:

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2}(x_j^t - x_i^t) + \alpha \varepsilon_i^t \tag{3.17}$$

where t is the current iteration number, x_t is the current position of the firefly i, the second term is due to the attraction, and the third term $\alpha\varepsilon$ is defined as the randomization term.

3.5 Grasshopper Optimization Algorithm

As an emerging intelligent algorithm, a GOA [58] is inspired by the life cycle of the grasshoppers. Grasshoppers exist alone in nature, but they are considered as one of the largest groups. The uniqueness of this creature is that the grasshopper swarm's change both in the larval and adult phases. The main characteristic of the grasshopper swarm in the larval stage is their slow movement and small steps. Conversely, their main characteristics in the adult stage are fast movement and large steps. Moreover, food source seeking is another critical characteristic. To find the food source, the search behaviour of the algorithm is divided into two parts: global exploration and local exploitation. In global exploration, the grasshopper swarm is encouraged to jump outwards and moves quickly. However, in local exploitation, the grasshopper swarm tends to move slowly in a small area conducive to local search. The mathematical formula of the GOA is as follows:

$$X_i = S_i + G_i + A_i \tag{3.18}$$

where X_i shows the position of the *i-th* grasshopper in the grasshopper swarm, S_i is the mutual influence between the *i-th* grasshopper and the other grasshoppers, G_i is the gravity force on the *i-th* grasshopper, A_i defines the wind advection affected the *i-th* grasshopper. Then, considering the influence of random factors, formula (3.18) can be rewritten as:

$$X_i = r_1 S_i + r_2 G_i + r_3 A_i \tag{3.19}$$

where r_1, r_2, and r_3 are random numbers in [0,1]. The formula S_i is as follows:

$$S_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} s(d_{ij}) \hat{d}_{ij} \tag{3.20}$$

where d_{ij} is the distance between the *i-th* and the *j-th* grasshopper, $\hat{d}_{ij}$ is the unit vector from the *i-th* grasshopper to the *j-th* grasshopper, calculated as $\hat{d}_{ij} = \frac{x_j - x_i}{d_{ij}}$, N is the number of grasshoppers, and s is defined as the function of social forces' strength between grasshoppers. s function can be calculated as the following equation:

$$s(r) = f e^{\frac{-r}{l}} - e^{-r} \tag{3.21}$$

where f represents the intensity of attraction, l is the attraction's length scale parameter. Through the s function, space, where the grasshoppers are located, can be divided into the attraction zone, the repulsion zone and the comfort zone. Because grasshoppers reach the comfort zone quickly and the grasshopper swarm can not converge to the global best solution, a modified version was introduced to solve optimization problems. The formula (3.19) can be replaced as follows:

$$X_i^d = c\left(\sum_{\substack{j=1\\ j\neq i}}^{N} c\frac{ub_d - lb_d}{2}s(| x_j^d - x_i^d|)\frac{x_j - x_i}{d_{ij}}\right) + \hat{T}_d \tag{3.22}$$

where ub_d and lb_d are the upper and lower bounds of the D-dimensional search space, $\hat{T}_d$ is the optimal solution in the current grasshopper swarm, c is the decreasing coefficient to shrink the comfort zone, repulsion zone, and attraction zone, and it can be expressed as follows:

$$c = c_{max} - l \times \frac{c_{max} - c_{min}}{L} \tag{3.23}$$

where c_{max} and c_{min} are the maximum, minimum value of c. l represents the current number of iterations, and L is the maximum number of iterations.

3.6 Moth Search Algorithm

It is known that the phototaxis and Levy flights are two representative features of moths. The phototaxis is the movement in response to light of the organism, and following with Lévy flights is one of the characteristics of moths.

These two characteristics correspond to the development and exploration process of any meta-heuristic optimization method. By using these two features, Wang [59] constructed a new metaheuristic optimization method called MSA.

To prove the superiority of its performance, they compared the MSA with other advanced meta-heuristic optimization algorithms. The results showed that MSA in most cases, such as test functions and engineering cases, is superior to other methods.

There are three concepts of MSA:

(1) The fittest moth is viewed as the light source.

(2) The moths near the best one displayed on the form of Lévy flights.

(3) The moths that far from the best one will fly directly to the best one.

Moths' positions are updated by performing Lévy flights. Lévy flight is defined as a random walk, and its step size comes from the Lévy distribution. The distribution of Lévy flights can be expressed mathematically in a power-law formula:

$$L(s) \sim |s|^{-\beta} \tag{3.24}$$

Where β is an index and $1 < \beta \leq 3$.

Firstly, the position of *i-th* moth can be updated by the following equation:

$$x_i^{t+1} = x_i^t + \alpha L(s) \tag{3.25}$$

where t represents the current number of generations, x_i^{t+1} represents the present position, x_i^t represents the previous position. $L(s)$ is the step extracted from Lévy flights. The parameter α related to the problem of interest can be expressed as:

$$\alpha = S_{max}/t^2 \tag{3.26}$$

where S_{max} is the maximum walking step length, and its value is based on the given question.

The Lévy distribution $L(s)$ in Equation (3.24) can be expressed as follows:

$$L(s) = \frac{(\beta - 1)\Gamma(\beta - 1)\sin\left(\frac{\pi(\beta - 1)}{2}\right)}{\pi s^{\beta}} \tag{3.27}$$

where s is a number bigger than 0 and $\Gamma(x)$ is the gamma function. The moths that far from the best one will fly straightly, and its flight can be described as below:

$$x_i^{t+1} = \lambda \times \left(x_i^t + \phi \times \left(x_{best}^t - x_i^t\right)\right) \tag{3.28}$$

where x_{best}^t means the best moth at generation t, λ is a scale factor and φ is the acceleration factor set to the golden ratio. To mention the situation that the moths might fly beyond the position of the light source, we can express the final position of moth i as below:

$$x_i^{t+1} = \lambda \times \left(x_i^t + \frac{1}{\varphi} \times \left(x_{best}^t - x_i^t \right) \right) \tag{3.29}$$

Moth's position can be updated by these two equations (Eq. (3.28) or Eq. (3.29)) above with 50% probability.

In MSA, for simplicity, the population is divided into two Subpopulations (Subpopulation1 & Subpopulation 2) according to the fitness estimate of the entire moth population. In summary, for all moths in Subpopulation 1, their positions are updated by performing Lévy flights (Eq. (3.25)); On the other hand, they are moved by performing Fly straightly (Eq. (3.28) or Eq. (3.29)) for all moths in Subpopulation 2.

Chapter 4. Hybrid Artificial Intelligence Models

4.1 Hybrid Meta-heuristic-SVM Models

To select the suitable parameters for SVM, the meta-heuristic algorithms, PSO, NHPSO-JTVAC, BA, FA, GOA, and MSA, which were mentioned in Chapter 3, were adopted to optimize the SVM parameters.

The fitness function (Equation 4.2) selected in the hybrid SVM-based model was the mean absolute percentage error (MAPE) based on cross-validation (CV). Figure 4.2 shows the concept of k-fold CV. To prevent the model from overfitting, a 5-fold CV method was adopted in this study [60].

$$MAPE = \frac{100\%}{m}\sum_{i=1}^{m}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{4.1}$$

$$fitness(MAPE_{CV_{(k)}}) = \frac{1}{k}\sum_{n=1}^{k} MAPE_n \tag{4.2}$$

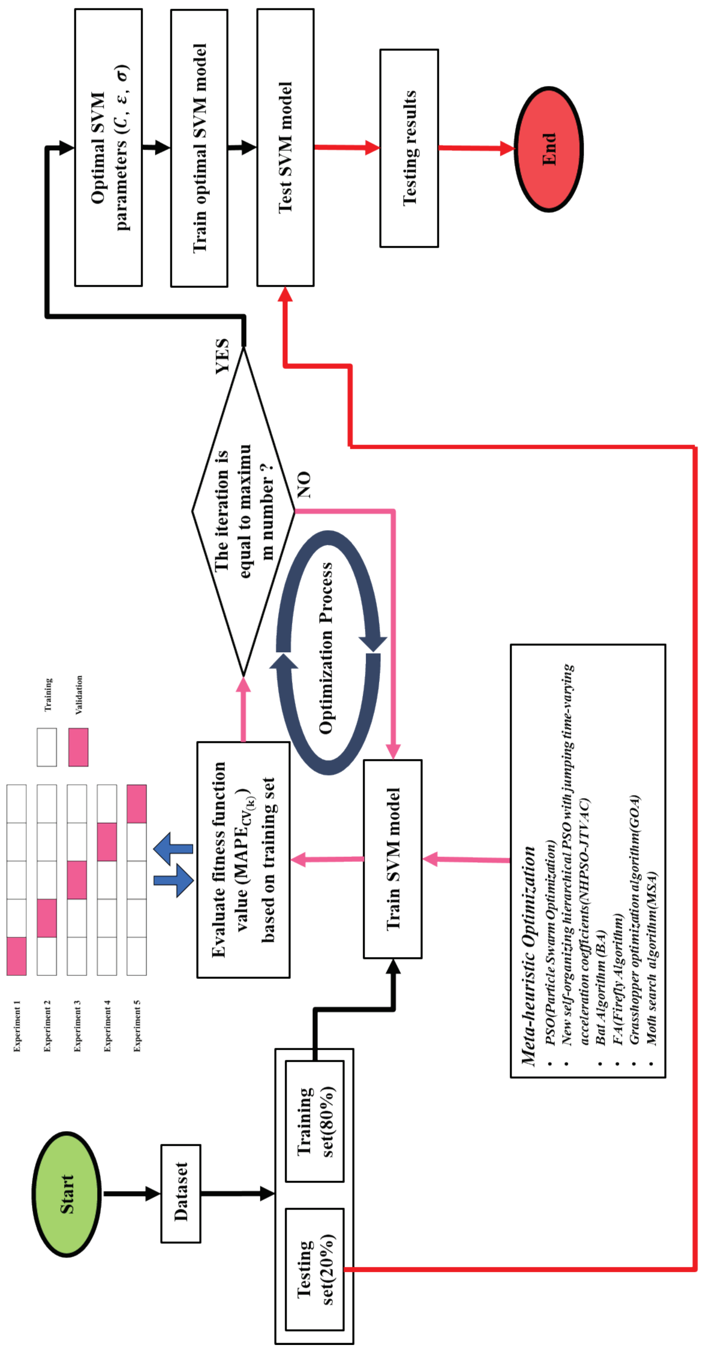
where m represents the training samples' number of each fold.

In all the meta-heuristic algorithms, the population size was unified and set to 20, and the number of iterations was set to 500. The search space dimension of each algorithm was set to 3, which represented the three parameters (C, ε, σ) of the SVM. The search range of C was set to [10^0, 10^3], ε was set to [10^{-3}, 10^0], and σ was set to [10^{-2}, 10^1].

Figure 4.1 illustrates the flow chart of SVM based on meta-heuristic optimization. The optimization procedure of the SVM parameters is as follows:

- Preprocess the data, and then split the dataset randomly into a training set and test set at the ratio of 8:2.
- The meta-heuristic optimization process is started when the SVM training begins.

- If the termination condition is not satisfied, the meta-heuristic optimization process will be continued; otherwise, the optimal solution is obtained, and the algorithm will be finished.



Start
Dataset
Testing set(20%)
Training set(80%)
Experiment 1
Experiment 2
Experiment 3
Experiment 4
Experiment 5
Training
Validation
Evaluate fitness function value (MAPE$_{CV(k)}$) based on training set
Train SVM model
Optimization Process
The iteration is equal to maximum number ?
YES
NO
Meta-heuristic Optimization
PSO(Particle Swarm Optimization)
New self-organizing hierarchical PSO with jumping time-varying acceleration coefficients(NHPSO-JTVAC)
Bat Algorithm (BA)
FA(Firefly Algorithm)
Grasshopper optimization algorithm(GOA)
Moth search algorithm(MSA)
Optimal SVM parameters (C, ε, σ)
Train optimal SVM model
Test SVM model
Testing results
End

Figure 4.1 Flow chart of mate-heuristic optimization within SVM.

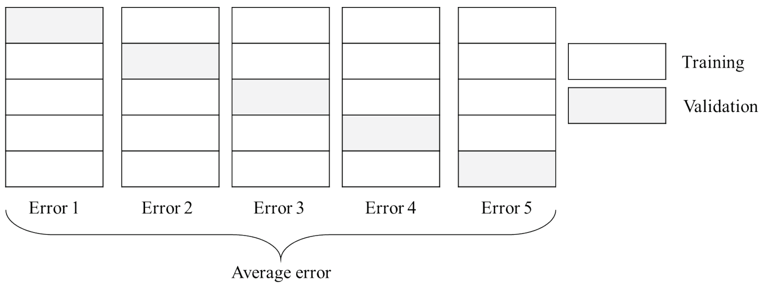


Figure 4.2 Concept of k-fold cross-validation (k=5).

4.1.1 Hybrid PSO-SVM Model

Table 4.1 shows the parameter setting of the PSO. The learning factors c_1, c_2 were set to 2, and the maximal/minimal inertia weights were set to 0.4, 0.9. Table 4.2 illustrates the pseudocode of the PSO-SVM.

Algorithm 1	Parameter	Value
PSO	c_1	2
	c_2	2
	ω_{min}	0.4
	ω_{max}	0.9
	Number of particles	20

Table 4.1 Parameter setting of PSO.

```
Hybrid-model 1 PSO-SVM
Begin
  Initialize parameters of PSO by using Table 4.1
  Initialize the population
  Set the Eq. (4.2) as the fitness function
  while Generation < 500 do
    Update the inertia weight by Eq. (3.3)
    for $i = 1$ to number of particles do
      Train the SVM and evaluate the $fitness(MAPE_{CV})$
      Update the p-best and g-best
      for $d = 1$ to number of dimensions do
        Update the velocity of particles by Eq. (3.1)
        Update the position of particles by Eq. (3.2)
      end for $d$
    end for $i$
  end while
  Output the optimum parameters ($C$, $\varepsilon$, $\sigma$) of the SVM
End
```

Table 4.2 Pseudocode of the PSO-SVM.

4.1.2 Hybrid NHPSO-JTVAC-SVM Model

Table 4.3 shows the parameter setting of the NHPSO-JTVAC. c_i was set to 0.5, c_f was set to 0. Table 4.4 illustrates the pseudocode of the NHPSO-JTVAC-SVM.

Algorithm 2	Parameter	Value
NHPSO-JTVAC	c_i	$c^1 = c_i = 0.5$
	c_f	$c^{Iter_{max}} = c_f = 0$
	Number of particles	20

Table 4.3 Parameter setting of NHPSO-JTVAC.

Hybrid-model 2 NHPSO-JTVAC-SVM

```
Begin
  Initialize parameters of NHPSO-JTVAC by using Table 4.3
  Initialize the population
  Set the Eq. (4.2) as the fitness function
  while Generation < 500 do
    Update the $c^{Iter}$ by Eq. (3.4)
    for $i = 1$ to number of particles do
      Train the SVM model and evaluate the $fitness(MAPE_{CV})$
      Update the p-best and g-best
      for $d = 1$ to number of dimensions do
        Calculate the new velocity of particles by search Eq. (3.7)
        if $v_{i,d}^{Iter+1} = 0$
          if $rand(\cdot) < 0.5$
            $v_{i,d}^{Iter+1} = rand(\cdot) * v_{max}(d)$
          else
            $v_{i,d}^{Iter+1} = -rand(\cdot) * v_{max}(d)$
          end if
        end if
        $v_{i,d}^{Iter+1} = sign(v_{i,d}^{Iter+1}) * min(abs(v_{i,d}^{Iter+1}, v_{max}))$
        Update the particle's position by Eq. (3.2)
      end for $d$
    end for $i$
  end while
  Output the optimum parameters $(C, \varepsilon, \sigma)$ of the SVM
End
```

Table 4.4 Pseudocode of the NHPSO-JTVAC-SVM.

4.1.3 Hybrid BA-SVM Model

Table 4.5 shows the parameter setting of the BA. Loudness A was set to 0.8, and pulse rate r was set to 0.95, pulse frequency minimum f_{min} was set to 0, and pulse frequency maximum f_{max} was set to 10. Table 4.6 illustrates the pseudocode of the BA-SVM.

Algorithm 3	Parameter	Value
BA	A	0.8
	r	0.95
	f_{min}	0
	f_{max}	10
	Population size	20

Table 4.5 Parameter setting of BA.

```
Hybrid-model 3 BA-SVM
Begin
  Initialize parameters of BA by using Table 4.5
  Initialize the population
  Set the Eq. (4.2) as the fitness function
  while Generation < 500 do
    for $i = 1$ to number of bats do
      Generate new solutions by using Eqs. (3.8)-(3.10)
      if $rand > r_i$
        Choose a solution among the best solutions and then generate a local
        solution around it
      end if
      Generate a new solution through flying randomly
      if $rand < A_i$ & $f_{current}(MAPE_{CV}) > f_{new}(MAPE_{CV})$
        Accept the new solution
        Increase $r_i$ and reduce $A_i$ by using Eqs. (3.12) & (3.13)
      end if
    end for
    Find the current best bat
  end while
  Output the optimum parameters ($C$, $\varepsilon$, $\sigma$) of the SVM
End
```

Table 4.6 Pseudocode of the BA-SVM.

4.1.4 Hybrid FA-SVM Model

Table 4.7 shows the parameter setting of the FA. Firefly attractiveness value β_0 was set to 1, randomization parameter α was set to 0.2, and absorption coefficient γ was set to 1. Table 4.8 illustrates the pseudocode of the FA-SVM.

Algorithm 4	**Parameter**	**Value**
FA	β_0	1
	α	0.2
	γ	1
	Population size	20

Table 4.7 Parameter setting of FA.

Hybrid-model 4 FA-SVM

```
Begin
   Initialize parameters of FA by using Table 4.7
   Initialize the population
   Set the Eq. (4.2) as the fitness function
   while Generation < 500 do
      for $i = 1$ to number of fireflies do
         for $j = 1$ to number of fireflies do
            if ($I_j > I_i$)
               Move firefly $i$ towards $j$ ($d - dimension$) by Eq. (3.17)
            end if
            Chance attractiveness with distance $r$
            Evaluate new solutions and update light intensity
         end for $j$
      end for $i$
   Find the current best firefly
   end while
   Output the optimum parameters ($C$, $\varepsilon$, $\sigma$) of the SVM
End
```

Table 4.8 Pseudocode of the FA-SVM.

4.1.5 Hybrid GOA-SVM Model

Table 4.9 shows the parameter setting of the GOA. Firefly attractiveness value f represents the intensity of attraction, which was set to 0.5, attractive length scale l was set to 1.5, and the maximum, minimum value of c was set to 1, 0.00004. Table 4.10 illustrates the pseudocode of the GOA-SVM.

Algorithm 5	**Parameter**	**Value**
GOA	f	0.5
	l	1.5
	c_{min}	0.00004
	c_{max}	1
	Population size	20

Table 4.9 Parameter setting of GOA.

Hybrid-model 5 GOA-SVM

```
Begin
  Initialize parameters of GOA by using Table 4.9
  Initialize the population
  Set the Eq. (4.2) as the fitness function
  while Generation < 500 do
    Update $c$ by using Eq. (3.23)
    for $i = 1$ to number of grasshoppers do
      Normalize the distances between grasshoppers in [1,4]
      Update the position of current grasshopper by using Eq. (3.22)
      Bring the current grasshopper back if it goes outside the boundaries
    end for $i$
    Find the current best grasshopper
  end while
  Output the optimum parameters ($C$, $\varepsilon$, $\sigma$) of the SVM
End
```

Table 4.10. Pseudocode of the GOA-SVM

4.1.6 Hybrid MSA-SVM Model

Table 4.11 illustrates the parameter setting of the MSA. Table 4.12 illustrates the pseudocode of the MSA-SVM. This study has chosen β = 0.5, S_{max} = 1, and φ = ($5^{1/2}$− 1) ≈ 0.618. Table 4.12 illustrates the pseudocode of the MSA-SVM.

Algorithm 6	Parameter	Value
MSA	β	1.5
	S_{max}	1
	φ	($5^{1/2}$− 1)/2
	Population size	20

Table 4.11 Parameter setting of MSA.

```
Hybrid-model 6 MSA-SVM
Begin
    Initialize parameters of MSA by using Table 4.11
    Initialize the population
    Set the Eq. (4.2) as the fitness function
    while Generation < 500 do
        Sort all moths according to evaluate their fitness
        for $i = 1$ to $NP/2$ (moth individuals in Subpopulation 1) do
            Update the position of current moth by using Eq. (3.25)
        end for $i$
        for $i = NP/2 + 1$ to $NP$ (moth individuals in Subpopulation 2) do
            if ($rand > 0.5$)
                Update the position of current moth by using Eq. (3.28)
            else
                Update the position of current moth by using Eq. (3.29)
            end if
        end for $i$
        Evaluate the population based on newly updated positions
    end while
    Output the optimum parameters ($C$, $\varepsilon$, $\sigma$) of the SVM
End
```

Table 4.12 Pseudocode of the MSA-SVM

4.2 Hybrid Meta-heuristic-ANN Models

Likewise, in order to select the appropriate parameters for an ANN, the meta-heuristic algorithms were used to optimize the ANN's initial weights and bias. The fitness function chosen in the hybrid ANN-based model was the MAPE function based on the training set.

$$fitness(MAPE_{training}) = \frac{100\%}{m}\sum_{i=1}^{m}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{4.3}$$

where m represents the number of total training samples.

In all the meta-heuristic algorithms, the population size was unified and set to 20, and the number of iterations set to 200. The search range of ANN's initial weights and bias was set to [-5, 5] [61].

Figure 4.3 illustrates the flow chart of the meta-heuristic optimization within ANN. The optimization procedure of the ANN parameters is as follows:

- Preprocess the data, and then split the dataset randomly into a training and test set (8:2).
- Determine the structure of the ANN.
- The meta-heuristic optimization process is started when the training of the ANN begins.
- If the termination condition is not satisfied, the meta-heuristic optimization process will be continued; otherwise, the optimal solution is obtained, and the algorithm will be finished.

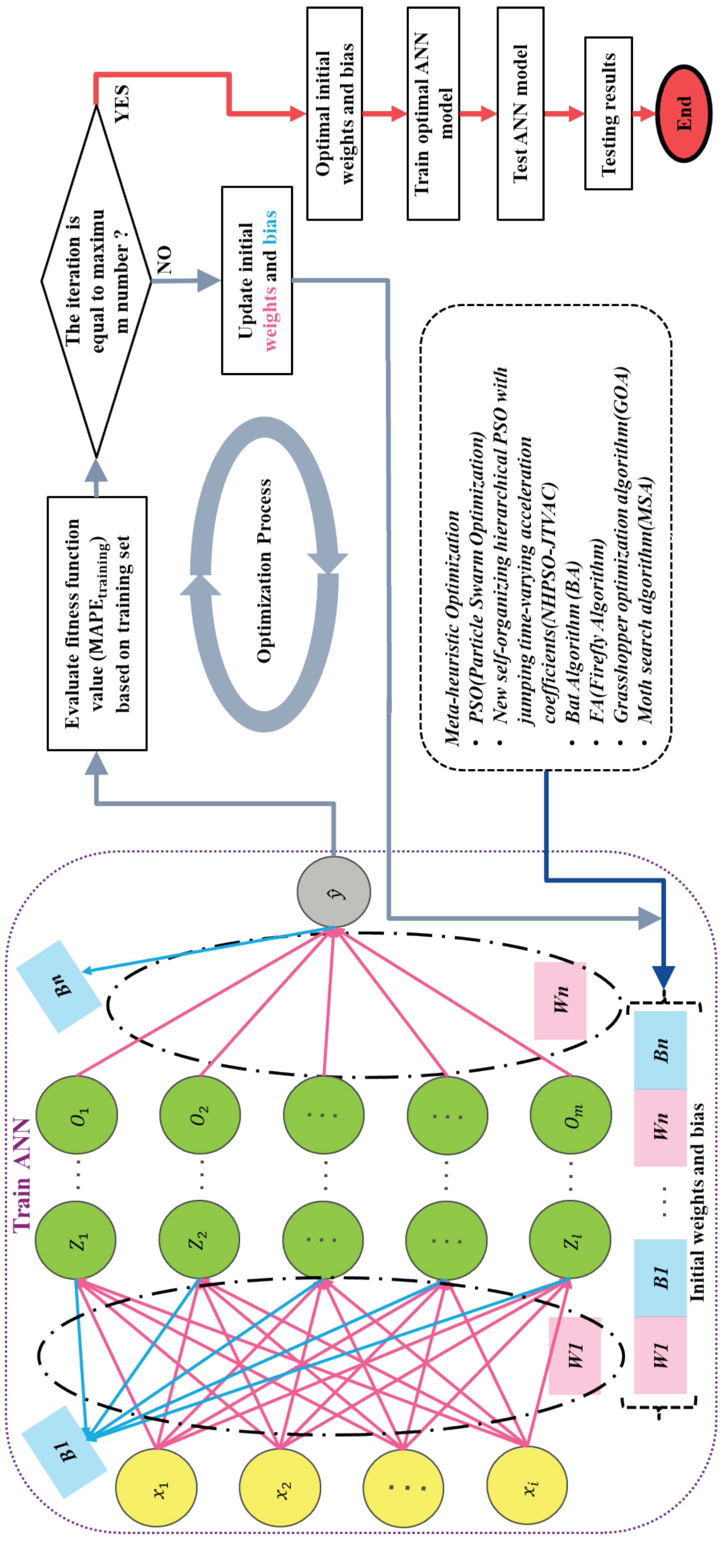

Train ANN
x_1
x_2
. . .
x_i
B1
W1
Z_1
Z_2
Z_l
O_1
O_2
O_m
B''
Wn
$\hat{y}$
Initial weights and bias
W1
B1
Wn
Bn
Evaluate fitness function value (MAPE$_{training}$) based on training set
The iteration is equal to maximum number ?
YES
NO
Update initial weights and bias
Optimization Process
Meta-heuristic Optimization
PSO(Particle Swarm Optimization)
New self-organizing hierarchical PSO with jumping time-varying acceleration coefficients(NHPSO-JTVAC)
Bat Algorithm (BA)
FA(Firefly Algorithm)
Grasshopper optimization algorithm(GOA)
Moth search algorithm(MSA)
Optimal initial weights and bias
Train optimal ANN model
Test ANN model
Testing results
End

Figure 4.3 Flow chart of mate-heuristic optimization within ANN.

4.2.1 Hybrid PSO-ANN Model

The parameters setting of PSO is consistent with Table 4.1. Table 4.13 illustrates the pseudocode of the PSO-ANN.

Hybrid-model 7 PSO-ANN

```
Begin
  Initialize parameters of PSO by using Table 4.1
  Initialize the population
  Set the Eq. (4.3) as the fitness function
  while Generation < 200 do
    Update the inertia weight by Eq. (3.3)
    for $i = 1$ to number of particles do
      Train the ANN and evaluate the $fitness(MAPE_{training})$
      Update the p-best and g-best
      for $d = 1$ to number of dimensions do
        Update the velocity of particles by Eq. (3.1)
        Update the position of particles by Eq. (3.2)
      end for $d$
    end for $i$
  end while
  Output the optimum initial weights and biases of the ANN
End
```

Table 4.13 Pseudocode of the PSO-ANN.

4.2.2 Hybrid NHPSO-JTVAC-ANN Model

The parameters setting of NHPSO-JTVAC is consistent with Table 4.3. Table 4.14 illustrates the pseudocode of the NHPSO-JTVAC-ANN.

Hybrid-model 8 NHPSO-JTVAC-ANN

```
Begin
  Initialize parameters of NHPSO-JTVAC by using Table 4.3
  Initialize the population
  Set the Eq. (4.3) as the fitness function
  while Generation < 200 do
    Update the $c^{Iter}$ by Eq. (3.4)
    for $i = 1$ to number of particles do
      Train the SVM model and evaluate the $fitness(MAPE_{training})$
      Update the p-best and g-best
      for $d = 1$ to number of dimensions do
        Calculate the new velocity of particles by search Eq. (3.7)
        if $v_{i,d}^{Iter+1} = 0$
          if $rand(\cdot) < 0.5$
            $v_{i,d}^{Iter+1} = rand(\cdot) * v_{max}(d)$
          else
            $v_{i,d}^{Iter+1} = -rand(\cdot) * v_{max}(d)$
          end if
        end if
        $v_{i,d}^{Iter+1} = sign(v_{i,d}^{Iter+1}) * min(abs(v_{i,d}^{Iter+1}, v_{max}))$
        Update the particle's position by Eq. (3.2)
      end for $d$
    end for $i$
  end while
  Output the optimum initial weights and biases of the ANN
End
```

Table 4.14 Pseudocode of the NHPSO-JTVAC-ANN.

4.2.3 Hybrid BA-ANN Model

The parameters setting of BA is consistent with Table 4.5. Table 4.15 illustrates the pseudocode of the BA-ANN.

Hybrid-model 9 BA-ANN

```
Begin
    Initialize parameters of BA by using Table 4.5
    Initialize the population
    Set the Eq. (4.3) as the fitness function
    while Generation < 200 do
        for $i = 1$ to number of bats do
            Generate new solutions by using Eqs. (3.8)-(3.10)
            if $rand > r_i$
                Choose a solution among the best solutions and then generate a local
                solution around it
            end if
            Generate a new solution through flying randomly
            If $rand < A_i$ & $f_{current}(MAPE_{training}) > f_{new}(MAPE_{training})$
                Accept the new solution
                Increase $r_i$ and reduce $A_i$ by using Eqs. (3.12) & (3.13)
            end if
        end for
        Find the current best bat
    end while
    Output the optimum initial weights and biases of the ANN
End
```

Table 4.15 Pseudocode of the BA-ANN.

4.2.4 Hybrid FA-ANN Model

The parameters setting of FA is consistent with Table 4.7. Table 4.16 illustrates the pseudocode of the FA-ANN.

Hybrid-model 10 FA-ANN

```
Begin
  Initialize parameters of FA by using Table 4.7
  Initialize the population
  Set the Eq. (4.3) as the fitness function
  while $Generation$ < 200 do
    for $i = 1$ to number of fireflies do
      for $j = 1$ to number of fireflies do
        if ($I_j > I_i$)
          Move firefly $i$ towards $j$ ($d - dimension$) by Eq. (3.17)
        end if
        Chance attractiveness varies with distance $r$
        Evaluate new solutions and update light intensity
      end for $j$
    end for $i$
  Find the current best firefly
  end while
  Output the optimum initial weights and biases of the ANN
End
```

Table 4.16 Pseudocode of the FA-ANN.

4.2.5 Hybrid GOA-ANN Model

The parameters setting of GOA is consistent with Table 4.9. Table 4.17 illustrates the pseudocode of the GOA-ANN.

Hybrid-model 11 GOA-ANN

```
Begin
  Initialize parameters of GOA by using Table 4.9
  Initialize the population
  Set the Eq. (4.3) as the fitness function
  while Generation < 200 do
    Update $c$ by using Eq. (3.23)
    for $i = 1$ to number of grasshoppers do
      Normalize the distances between grasshoppers in [1,4]
      Update the position of current grasshopper by using Eq. (3.22)
      Bring the current grasshopper back if it goes outside the boundaries
    end for $i$
    Find the current best grasshopper
  end while
  Output the optimum initial weights and biases of the ANN
End
```

Table 4.17 Pseudocode of the GOA-ANN.

4.2.6 Hybrid MSA-ANN Model

The parameters setting of MSA is consistent with Table 4.11. Table 4.18 illustrates the pseudocode of the MSA-ANN.

Hybrid-model 12 MSA-ANN

```
Begin
  Initialize parameters of MSA by using Table 4.11
  Initialize the population
  Set the Eq. (4.3) as the fitness function
  while Generation < 200 do
    Sort all moths according to evaluate their fitness
    for $i = 1$ to $NP/2$ (moth individuals in Subpopulation 1) do
      Update the position of current moth by using Eq. (3.25)
    end for $i$
    for $i = NP/2 + 1$ to $NP$ (moth individuals in Subpopulation 2) do
      if ($rand > 0.5$)
        Update the position of current moth by using Eq. (3.28)
      else
        Update the position of current moth by using Eq. (3.29)
      end if
    end for $i$
    Evaluate the population based on newly updated positions
  end while
  Output the optimum initial weights and biases of the ANN
End
```

Table 4.18 Pseudocode of the MSA-ANN.

Chapter 5. Lead Time Prediction Based on Hybrid Artificial Intelligence Models

5.1 Data and Preparation

This study predicts the lead time in the shipyard block processes by presenting the hybrid AI models. As shown in Table 5.1 and Figure 5.1, this thesis applied it to three datasets collected from a shipyard's block assembly, a pre-outfitting and a pre-painting process to evaluate the proposed model. The assembly, pre-outfitting and pre-painting processes consisted of information from 4779, 4198 and 6634 blocks. Each dataset was split into training and test data. 80% of each dataset, 3823, 3358 and 5307 data points were used to train data individually, and 20% of each dataset, 956, 840 and 1327 data points, were used to test data separately.

No	Dataset	Prediction target
1	Assembly process performance data	Lead time
2	Pre-outfitting process performance data	Lead time
3	Pre-painting process performance data	Lead time

Table 5.1 Data collection.

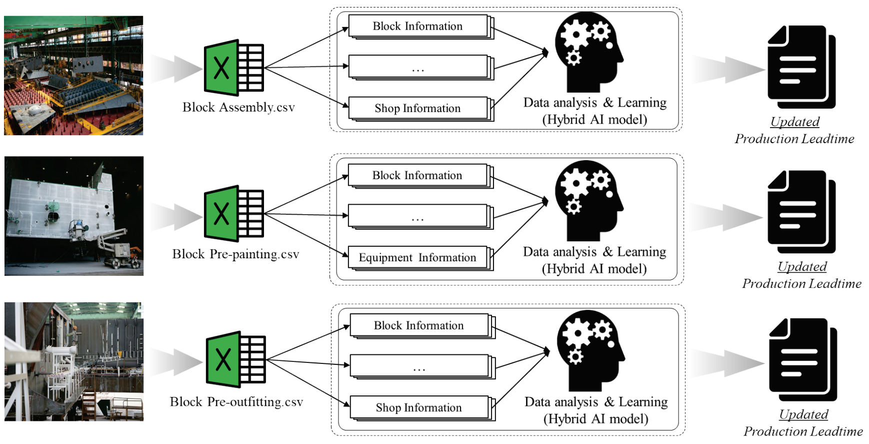

Block Assembly.csv
Block Information
…
Shop Information
Data analysis & Learning (Hybrid AI model)
Updated Production Leadtime
Block Pre-painting.csv
Block Information
…
Equipment Information
Data analysis & Learning (Hybrid AI model)
Updated Production Leadtime
Block Pre-outfitting.csv
Block Information
…
Shop Information
Data analysis & Learning (Hybrid AI model)
Updated Production Leadtime

Figure 5.1 Conceptual graph of proposed production lead time management method.

5.1.1 Data Normalization

In ML, to make the ML model more suitable for the actual situation, it is recommended that the data set be normalized [62-65]. This study performed normalization preprocessing on the training and test samples, which helps to eliminate significant differences between different scales on the learning speed, prediction accuracy, and generality of SVM and ANN. The data were normalized to [0,1] by using the normalization formula (5.1).

$$x_{scaled} = \frac{x - x_{m_{in}}}{x_{m_{ax}} - x_{m_{in}}} \tag{5.1}$$

5.1.2 Feature Selection

Feature selection (FS) can reduce computation time, improve learning accuracy, and better understand the learning model or data [66]. So, FS is essential when building a ML model. This study performed feature engineering and removed irrelevant features.

Step 1. The data are split into the training and test sets (8:2).

Step 2. The random forest (RF) model is trained using a training set.

Step 3. The importance score for each feature in the training set is calculated. Features are ranked by feature importance scores.

Step 4. The feature, which had a minimum importance score, will be deleted when the model's accuracy and execution time are not satisfied. In this situation, Steps 2 and 3 will be repeated until the desired number of features is obtained. Otherwise, the feature subset is obtained directly.

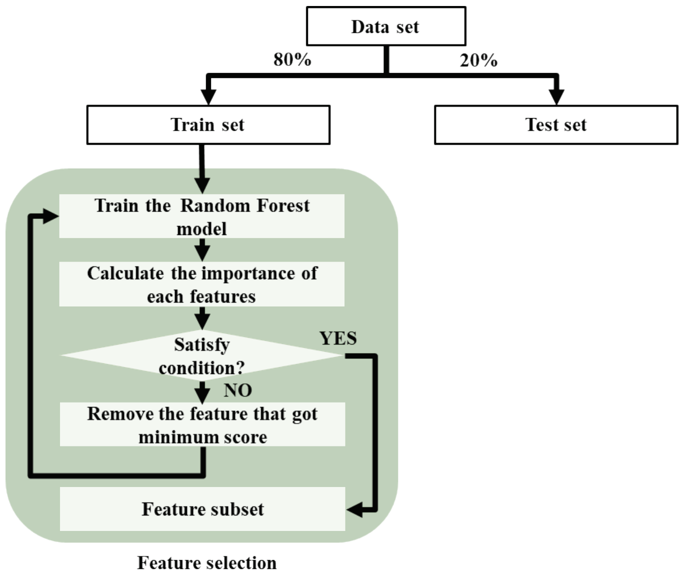
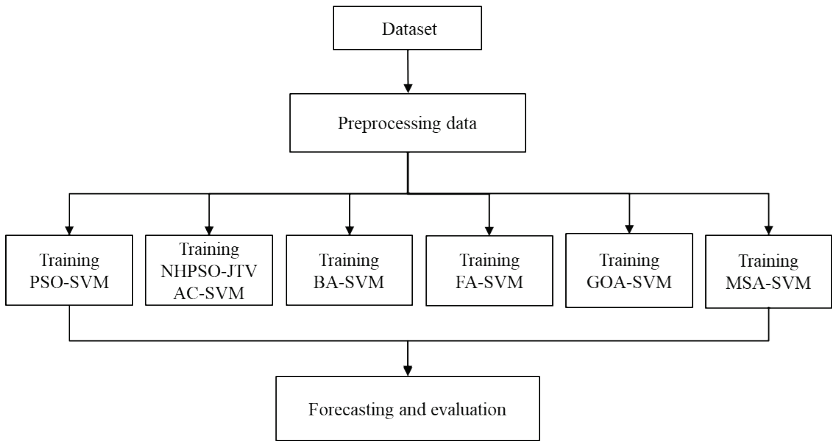

Data set
80%
20%
Train set
Test set
Train the Random Forest model
Calculate the importance of each features
Satisfy condition?
YES
NO
Remove the feature that got minimum score
Feature subset
Feature selection

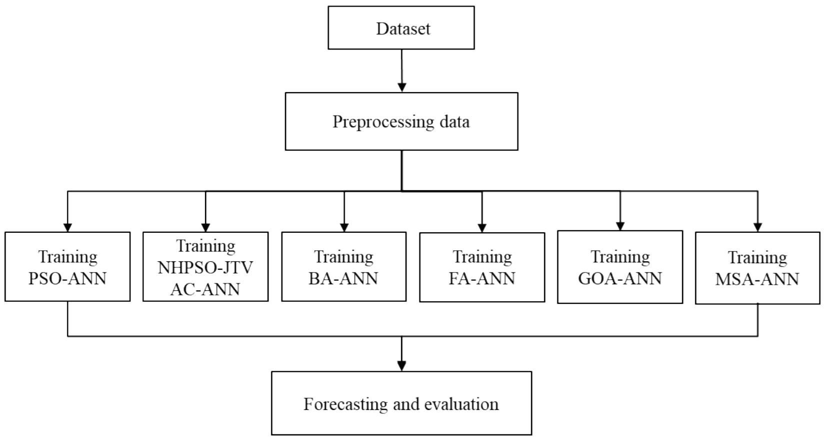
Figure 5.2 Flow chart of feature selection.

5.2 Lead Time Prediction

As shown in Figure 5.3a,b, the hybrid SVM-based, ANN-based model is built to predict the lead time in the shipyard's block processes. Moreover, it also compared the performance of each algorithm and model with the others. The parameter setting's detail and prediction process can be seen in Chapter 4.



(**a**)Hybrid SVM-based models



(**b**) Hybrid ANN-based models

Figure 5.3 Flow chart of the prediction models.

5.3 Performance Metrics

To estimate the forecasting accuracy of each model, this study adopted certain widely used regression prediction performance metrics: root mean square error (RMSE), mean absolute error (MAE), and MAPE, as shown in Table 5.2. Here, N is the sample size, y_i is the actual value, and $\hat{y}_i$ is the predicted value.

Metrics	Calculation
RMSE $(y, \hat{y})$	$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$
MAE $(y, \hat{y})$	$\frac{1}{N}\sum_{i=1}^{N}\lvert y_i - \hat{y}_i\rvert$
MAPE $(y, \hat{y})$	$\frac{100\%}{N}\sum_{i=1}^{N}\left\lvert\frac{y_i - \hat{y}_i}{y_i}\right\rvert$

Table 5.2 Regression prediction performance metrics.

Lower values of MAPE, MAE, and RMSE indicate higher accuracy of the model, meaning that the prediction results are more convincing. In particular, the MAPE metric, listed in Table 5.3, is commonly used to evaluate industrial and business data [67].

MAPE	**Interpretation**
<10%	Highly accurate forecasting
10–20%	Good forecasting
20–50%	Reasonable forecasting
>50%	Inaccurate forecasting

Table 5.3 Interpretation of MAPE values.

Chapter 6. Experimental Results

6.1 Results Based on Hybrid SVM-based models

This thesis conducted the prediction experiments using the test data to verify the proposed hybrid SVM-based models. Also, this study had compared SVM, PSO-SVM, NHPSO-JTVAC-SVM, BA-SVM, FA-SVM, GOA-SVM, and MSA-SVM with each other. The 5-fold CV scores in the iterative process of the integrated models are shown graphically in Figures 6.1, 6.2, and 6.3. The optimal solutions searched by the models are presented in Table 6.1.

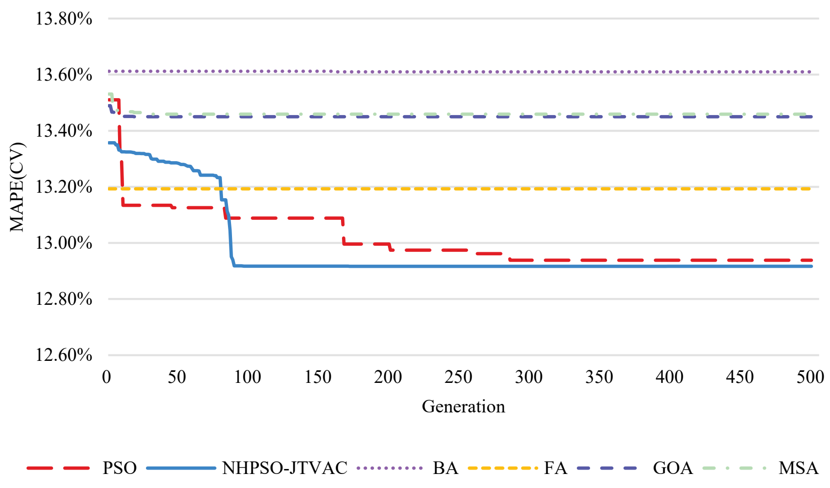


Figure 6.1 Generation of the optimization SVM models in block assembly process performance dataset.

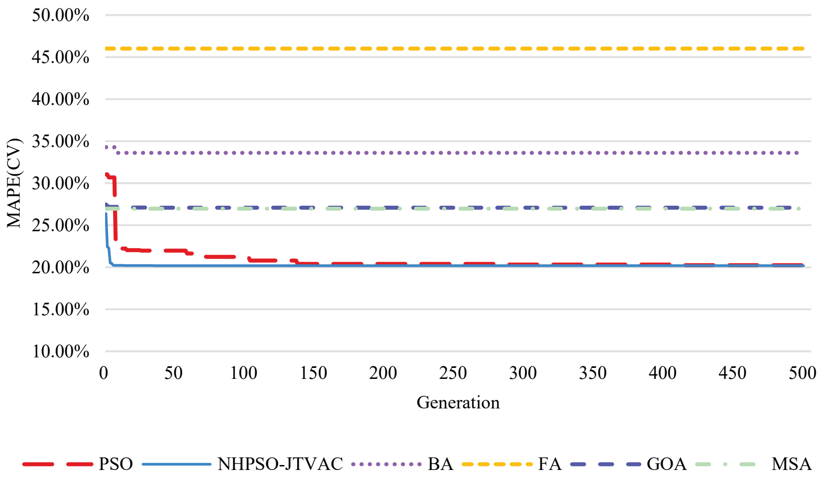


Figure 6.2 Generation of the optimization SVM models in block pre-outfitting process performance dataset.

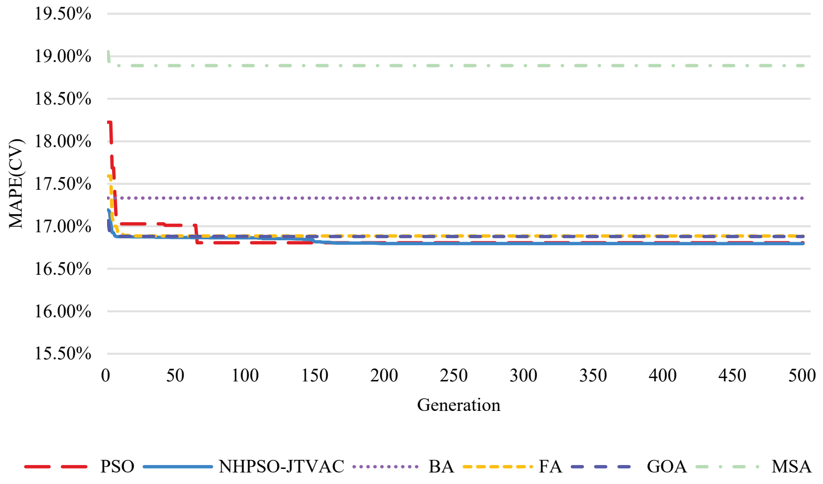


Figure 6.3 Generation of the optimization SVM models in block pre-painting process performance dataset.

Dataset		SVM	PSO-SVM	NHPSO-JTVAC-SVM	BA-SVM	FA-SVM	GOA-SVM	MSA-SVM
Assembly process performance data	Kfold_1	13.71%	13.26%	13.24%	14.01%	13.65%	13.68%	13.21%
	Kfold_2	13.96%	12.46%	12.44%	13.25%	12.93%	13.10%	13.46%
	Kfold_3	14.07%	13.49%	13.45%	14.46%	13.58%	14.10%	13.86%
	Kfold_4	13.27%	12.31%	12.31%	12.53%	12.49%	12.93%	12.97%
	Kfold_5	14.08%	13.17%	13.14%	13.80%	13.32%	13.45%	13.80%
	$\mathrm{MAPE_{CV}}$(Mean)	13.82%	12.94%	**12.92%**	13.61%	13.19%	13.45%	13.46%
Pre-outfitting process performance data	Kfold_1	24.29%	20.60%	20.49%	34.42%	45.41%	28.15%	26.77%
	Kfold_2	24.22%	20.98%	20.95%	32.55%	45.12%	27.64%	27.53%
	Kfold_3	23.55%	20.28%	20.11%	35.04%	45.71%	26.21%	26.52%
	Kfold_4	22.79%	19.71%	19.63%	34.93%	45.28%	26.66%	25.92%
	Kfold_5	22.78%	19.65%	19.77%	31.10%	48.54%	26.77%	28.11%
	$\mathrm{MAPE_{CV}}$(Mean)	23.53%	20.25%	**20.19%**	33.61%	46.01%	27.09%	26.97%
Pre-painting process performance data	Kfold_1	16.88%	16.76%	16.73%	17.28%	17.89%	16.81%	19.20%
	Kfold_2	16.97%	16.85%	16.89%	17.49%	18.10%	16.93%	19.22%
	Kfold_3	17.14%	16.96%	16.97%	17.19%	17.29%	17.09%	18.81%
	Kfold_4	16.56%	16.42%	16.40%	17.13%	17.55%	16.49%	18.44%
	Kfold_5	17.11%	17.04%	17.00%	17.57%	18.08%	17.10%	18.78%
	$\mathrm{MAPE_{CV}}$(Mean)	16.93%	16.81%	**16.80%**	17.33%	17.78%	16.88%	18.89%

Table 6.1 K-fold CV scores of each (meta-heuristic)-SVM model.

The results indicated that the NHPSO-JTVAC algorithm had the best search performance with the optimal fitness values of 12.92%, 20.19% and 16.80% in the block assembly process performance dataset, pre-outfitting process performance dataset and pre-painting process performance dataset, respectively.

At the same time, what can be observed is that the searching ability of the PSO algorithm was the second best. Optimal fitness values which PSO searched in the block assembly process performance data set, pre-outfitting process performance data set, and pre-painting process performance data set are 12.94%, 20.25% and 16.81%.

It is noteworthy that in the block assembly process performance dataset, the 5-fold CV scores of the proposed algorithms are all better than standard SVM. However, in the second, third dataset, algorithms such as BA, FA etc., seem to fall into a local optimum solution too early.

Table 6.2 illustrates the optimal values of the SVM parameters (C, ε, and σ) for each hybrid SVM-based model. In addition, Table 6.3 presents the test accuracy of these SVM models based on the MAE, RMSE, and MAPE. The test error of MAPE is graphically demonstrated in Figures 6.4, 6.5, and 6.6. It can be observed that the NHPSO-JTVAC-SVM model had the smallest MAPE in the training set (5-fold CV). In the block assembly process performance dataset, the MAPE of the NHPSO-JTVAC-SVM algorithm was 11.79%, and the MAE was 0.89. In the pre-outfitting process performance dataset, the MAPE and MAE were 17.86% and 0.96, respectively. Moreover, in the pre-painting process performance dataset, the MAPE and MAE were 16.57% and 1.85, separately. In addition, the NHPSO-JTVAC-SVM model was significantly better than the SVM model.

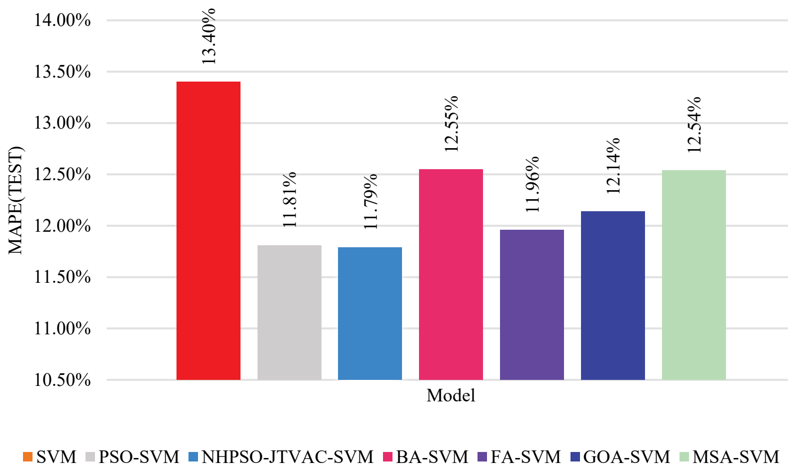


Figure 6.4 Test MAPE of each (meta-heuristic)-SVM model in block assembly process performance dataset.

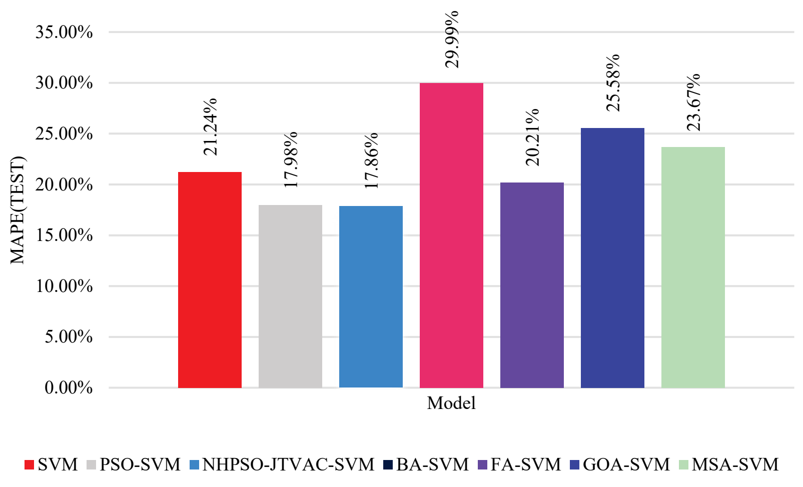


Figure 6.5 Test MAPE of each (meta-heuristic)-SVM model in block pre-outfitting process performance dataset.

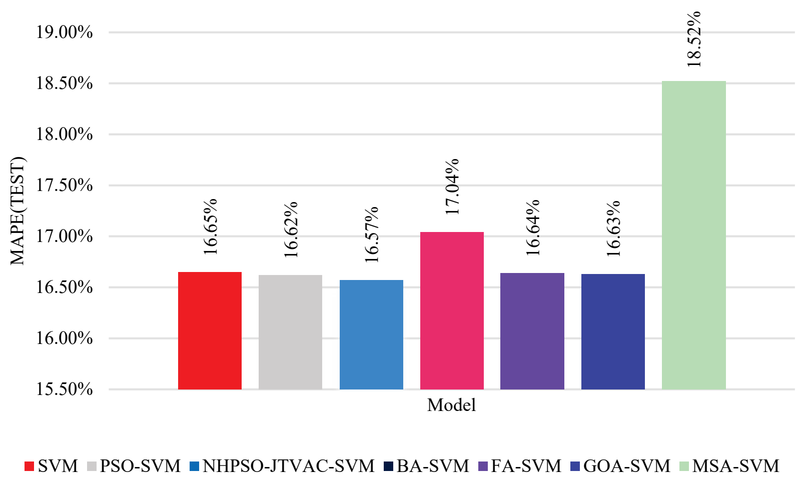


Figure 6.6 Test MAPE of each (meta-heuristic)-SVM model in block pre-painting process performance dataset.

Model	**Dataset**	***C***	***ε***	***σ***
PSO-SVM	Assembly process performance data	6.80	0.0172	6.8432
	Pre-outfitting process performance data	604.69	0.0157	0.0864
	Pre-painting process performance data	5.10	0.1801	0.2664
NHPSO-JTVAC-SVM	Assembly process performance data	5.87	0.0010	7.6115
	Pre-outfitting process performance data	997.94	0.0192	0.0666
	Pre-painting process performance data	2.57	0.0511	0.3200
BA-SVM	Assembly process performance data	113.65	0.1403	1.2904
	Pre-outfitting process performance data	201.55	0.3233	0.0509
	Pre-painting process performance data	819.10	0.6975	0.0775
FA-SVM	Assembly process performance data	8.85	0.0768	2.0664
	Pre-outfitting process performance data	334.07	0.0116	0.1026
	Pre-painting process performance data	603.08	0.1559	0.0160
GOA-SVM	Assembly process performance data	6.23	0.3650	6.3992
	Pre-outfitting process performance data	143.44	0.3556	0.0853
	Pre-painting process performance data	862.85	0.0763	0.0144
MSA-SVM	Assembly process performance data	311.58	0.7076	0.0890
	Pre-outfitting process performance data	98.14	0.5700	0.1525
	Pre-painting process performance data	198.83	1.0000	2.3679

Table 6.2 Optimal values of the three SVM parameters (C, ε, and σ).

Dataset		**SVM**	**PSO-SVM**	**NHPSO-JTVAC-SVM**	**BA-SVM**	**FA-SVM**	**GOA-SVM**	**MSA-SVM**
Assembly process performance data	MAPE (%)	13.40	11.81	**11.79**	12.55	12.20	12.14	12.54
	MAE	0.99	0.89	**0.89**	0.95	0.92	0.91	0.93
	RMSE	1.25	1.23	1.23	1.29	1.22	1.18	**1.17**
Pre-outfitting process performance data	MAPE (%)	21.24	17.98	**17.86**	29.99	38.94	25.58	23.67
	MAE	1.05	0.96	**0.96**	1.29	1.35	1.09	1.01
	RMSE	1.95	1.91	1.91	2.19	2.08	1.92	**1.83**
Pre-painting process performance data	MAPE (%)	16.65	16.62	**16.57**	17.04	16.64	16.63	18.52
	MAE	1.86	1.86	**1.85**	1.86	1.86	1.86	1.98
	RMSE	2.45	2.43	2.43	**2.42**	2.42	2.43	2.62

Table 6.3 Test errors of each (Meta-heuristic)-SVM model.

Table 6.4 lists the average MAPE values based on three datasets and obtained using SVM, PSO-SVM, NHPSO-JTVAC-SVM, BA-SVM, FA-SVM, GOA-SVM and MSA-SVM. The average MAPE for the NHPSO-JTVAC-SVM model was 15.41%, which was the smallest among the AI models.

	SVM	**PSO-SVM**	**NHPSO-JTVAC-SVM**	**BA-SVM**	**FA-SVM**	**GOA-SVM**	**MSA-SVM**
MAPE (%)	17.10	15.47	**15.41**	19.86	16.27	18.12	18.24

Table 6.4 Average MAPE of SVM, PSO-SVM, NHPSO-JTVAC-SVM, BA-SVM, FA-SVM, GOA-SVM, and MSA-SVM.

6.2 Results Based on Hybrid ANN-based models

In the same way, this study conducted prediction experiments using test data to verify the proposed hybrid ANN-based models. This study compared ANN, PSO-ANN, NHPSO-JTVAC-ANN, BA-ANN, FA-ANN, GOA-ANN, and MSA-ANN. The training set error (MAPE) in the iterative process of the integrated models are shown graphically in Figures 6.7, 6.8 and 6.9.

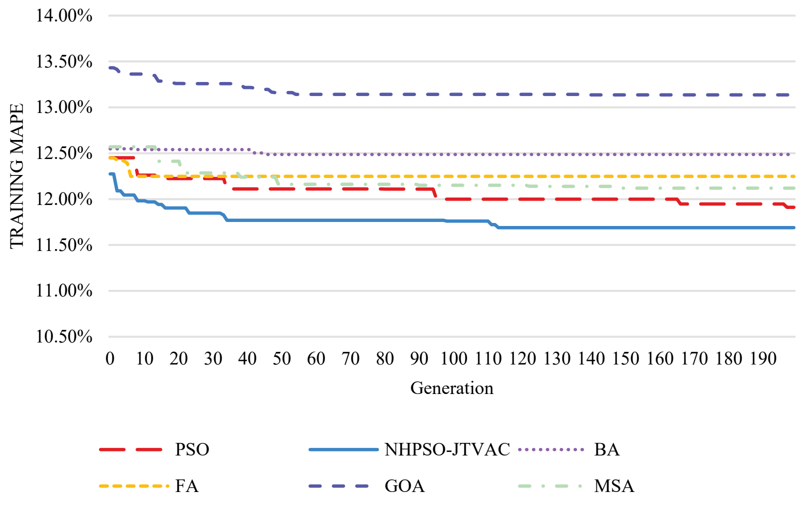


Figure 6.7 Generation of the optimization ANN models in block assembly process performance dataset.

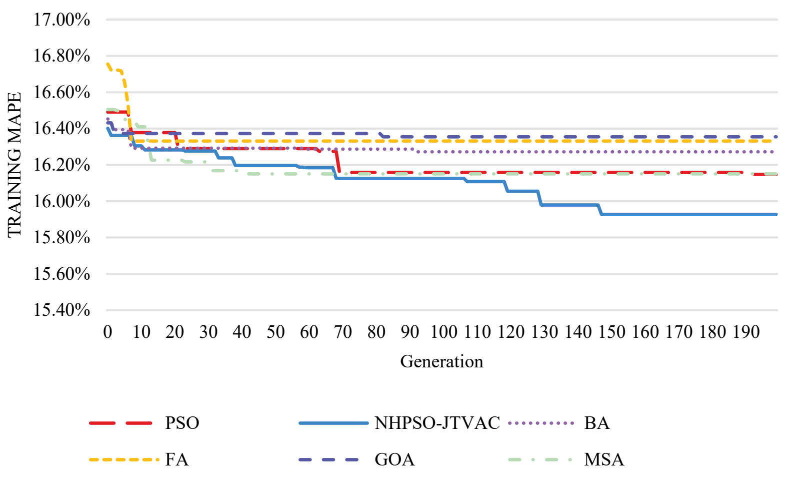


Figure 6.8 Generation of the optimization ANN models in block pre-outfitting process performance dataset.

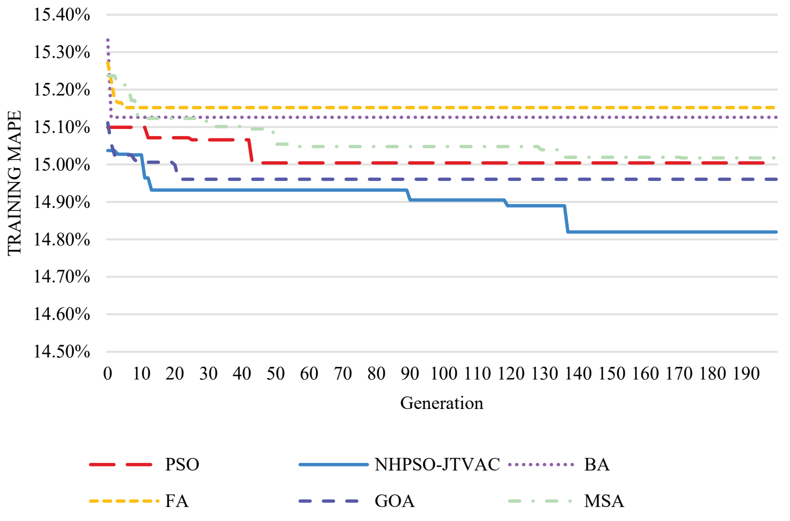


Figure 6.9 Generation of the optimization ANN models in block pre-painting process performance dataset.

Firstly, in the assembly process performance data, NHPSO-JTVAC had the best search performance with the optimal fitness value of 11.69%; PSO ranked second with the optimal fitness value of 11.91%; The optimal fitness values for MSA and FA were 12.12% and 12.25%, followed closely behind. But here, BA and GOA seem to fall into the problem of falling into a local optimum too early.

Secondly, in the pre-outfitting process performance data, NHPSO-JTVAC had the best search performance with the optimal fitness value of 15.93%. The second-best was PSO, with the optimal fitness value of 16.15%. It is worth noting that the search performance of MSA was similar to PSO.

Finally, in the pre-painting process performance data, it can be clearly seen that NHPSO-JTVAC also had the best search performance, and the optimal fitness value is 14.82%. But the second-best was GOA, with the optimal fitness value of 14.96%.

In addition, Table 6.5 presents the test accuracy of these ANN models based on the MAE, RMSE, and MAPE. The test error of MAPE, which this study set as the fitness function of the optimization process, is shown graphically in Figures 6.10, 6.11, and 6.12. The results indicated that most proposed hybrid ANN-based models had better test errors than the standard ANN model.

Furthermore, from the results, it can be found that the test error MAPE of the NHPSO-JTVAC-ANN model was the smallest in these models. In the block assembly process performance dataset, the MAPE of the NHPSO-JTVAC-ANN algorithm was 12.80%. In the pre-outfitting process performance dataset, the MAPE was 16.03%. Besides, in the pre-painting process performance dataset, the MAPE was 16.45%. In addition, the NHPSO-JTVAC-ANN model was greatly better than the ANN model.

Because the fitness function used in this study was the MAPE, it can be observed that PSO-ANN, MSA-ANN was better than NHPSO-JTVAC-ANN on other performance metrics such as MAE and RMSE.

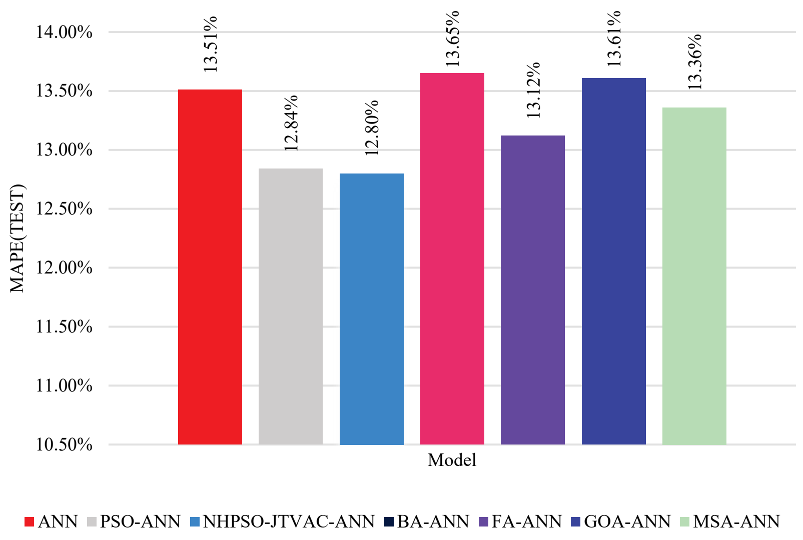


Figure 6.10 Test MAPE of each (meta-heuristic)-ANN model in block assembly process performance dataset.

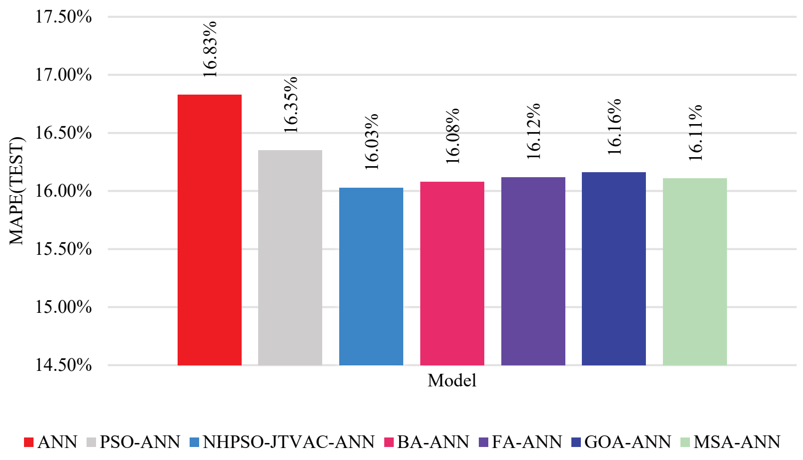


Figure 6.11 Test MAPE of each (meta-heuristic)-ANN model in block pre-outfitting process performance dataset.

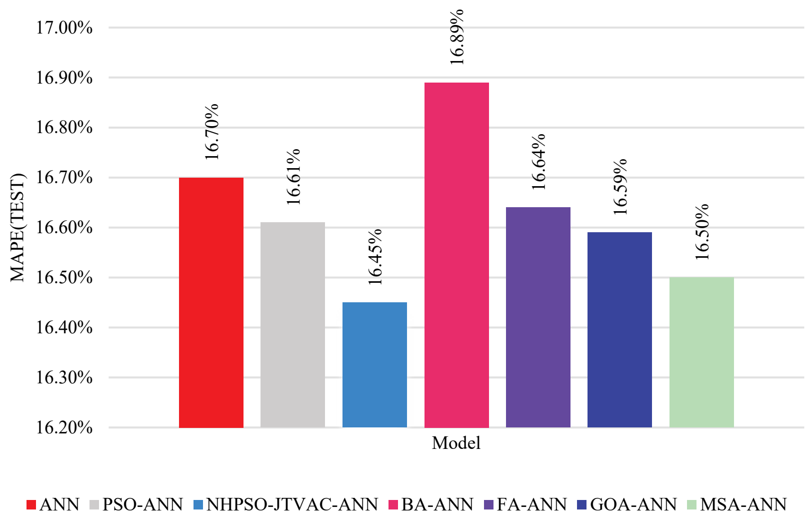


Figure 6.12 Test MAPE of each (meta-heuristic)-ANN model in block pre-painting process performance dataset.

Dataset		**ANN**	**PSO-ANN**	**NHPSO-JTVAC-ANN**	**BA-ANN**	**FA-ANN**	**GOA-ANN**	**MSA-ANN**
Assembly process performance data	MAPE (%)	13.51	12.84	**12.80**	13.65	13.12	13.61	13.36
	MAE	1.05	**0.97**	0.98	1.04	1.00	1.05	1.01
	RMSE	1.35	1.25	**1.25**	1.33	1.27	1.34	1.27
Pre-outfitting process performance data	MAPE (%)	16.83	16.35	**16.03**	16.08	16.12	16.16	16.11
	MAE	1.00	1.01	**0.99**	0.99	0.99	0.99	1.00
	RMSE	2.04	2.06	**2.03**	2.04	2.04	2.03	2.05
Pre-painting process performance data	MAPE (%)	16.70	16.61	**16.45**	16.89	16.64	16.59	16.50
	MAE	1.95	1.92	**1.90**	1.93	1.90	1.90	1.90
	RMSE	2.63	2.58	2.54	2.56	2.54	2.54	**2.52**

Table 6.5 Test errors of each (meta-heuristic)-ANN model.

Table 6.6 lists the average MAPE values based on three datasets obtained using ANN, PSO-ANN, NHPSO-JTVAC-ANN, BA-ANN, FA-ANN, GOA-ANN and MSA-ANN. It can be said that the average MAPE for the NHPSO-JTVAC-ANN model was 15.09%, which was the smallest among the AI models.

	ANN	PSO-ANN	NHPSO-JTVAC-ANN	BA-ANN	FA-ANN	GOA-ANN	MSA-ANN
MAPE (%)	15.68	15.27	**15.09**	15.54	15.29	15.45	15.32

Table 6.6 Average MAPE of ANN, PSO-ANN, NHPSO-JTVAC-ANN, BA-ANN, FA-ANN, GOA-ANN, and MSA-ANN.

6.3 Overall Results

As mentioned in Sections 6.1 and 6.2: in the hybrid SVM-based models, the MAPE error of NHPSO-JTVAC-SVM was the smallest; In the hybrid ANN-based models, the MAPE error of NHPSO-JTVAC-ANN was the smallest. So, this study compared the learning performance of NHPSO-JTVAC-SVM and NHPSO-JTVAC-ANN.

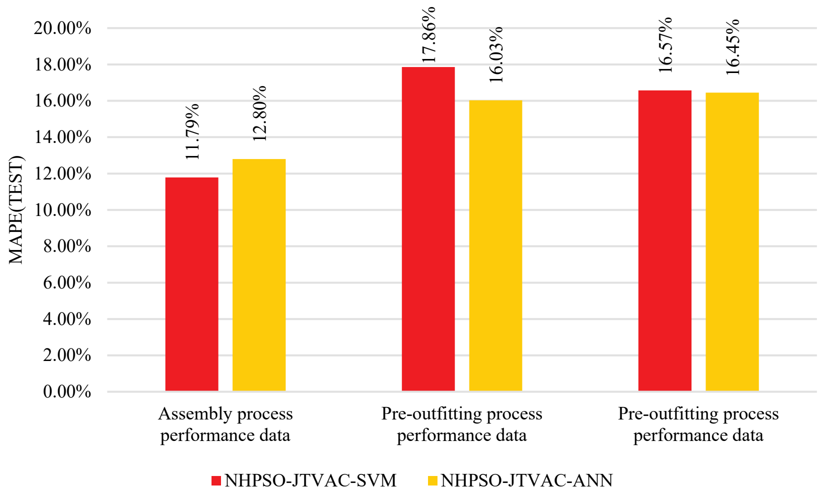


Figure 6.13 Comparison of test MAPE between NHPSO-JTVAC-SVM and NHPSO-JTVAC-ANN.

Figure 6.13 shows Test MAPE errors of NHPSO-JTVAC-SVM and NHPSO-JTVAC-ANN in three datasets. In the block assembly process performance data set, the test error MAPE of the NHPSO-JTVAC-SVM model was 11.79%, which was better than NHPSO-JTVAC-ANN. In turn, the

NHPSO-JTVAC-ANN had better learning performance than NHPSO-JTVAC-SVM in the block pre-outfitting and pre-painting performance data sets.

As a result, it seems more reasonable to select NHPSO-JTVAC-SVM as the prediction model for the block assembly process performance data set. Conversely, it appears more sensible for the block pre-outfitting and pre-painting process performance data sets to choose NHPSO-JTVAC-ANN as the prediction model.

Chapter 7. Conclusions and Future Works

It can be seen that many researchers have applied ML to predict lead time. However, due to the enormous variability and uncertainty of the manufacturing industry, the prediction will be inaccurate if the forecasting model was not optimized, thereby limiting the model's applicability in lead-time prediction.

Based on the analysis of the parameter performance of SVM and ANN, this study proposes hybrid SVM-based, ANN-based lead time prediction models. It fully utilizes the global search feature of the meta-heuristic algorithms to optimize the parameters of SVM and ANN, which overcomes the blindness of ML parameter selection. Compared with commonly used methods, the parameter selection in this paper provides clearer theoretical guidance. Additionally, in the searching process of parameters, it can be concluded that the NHPSO-JTVAC algorithm is superior in terms of performance.

Furthermore, the experimental results indicated that the NHPSO-JTVAC-SVM, NHPSO-JTVAC-ANN prediction model has good prediction accuracy. Overall, this study confirms the usefulness and value of the optimized predictive model for shipbuilding production.

It can be noted that the fitness function used in this thesis was the MAPE. Although the test error MAPE of the AI models based on NHPSO-JTVAC was better than other models, other performance metrics such as RMSE were worse than those of other models based on BA, MSA etc. To optimize the model further, we may develop an optimization algorithm that considers multi-fitness functions, an important aspect of future research.

Bibliography

1. Tatsiopoulos, I.; Kingsman, B. Lead time management. *European Journal of Operational Research* **1983**, *14*, 351-358.

2. Öztürk, A.; Kayalıgil, S.; Özdemirel, N.E. Manufacturing lead time estimation using data mining. *European Journal of Operational Research* **2006**, *173*, 683-700.

3. Zijm, W.H.; Buitenhek, R. Capacity planning and lead time management. *International Journal of Production Economics* **1996**, *46*, 165-179.

4. Brown, S.D.; Khan, H.; Salley, R.S.; Zhu, W. *Lead Time Estimation Using Artificial Intelligence*; LMI Tysons Corner United States: 2020.

5. CAPITANIO, F. Planning and control system in the shipbuilding. The integrated production plan at Fincantieri SpA. **2020**.

6. Berlec, T.; Govekar, E.; Grum, J.; Potocnik, P.; Starbek, M. Predicting order lead times. *STROJNISKI VESTNIK* **2008**, *54*, 308.

7. Lingitz, L.; Gallina, V.; Ansari, F.; Gyulai, D.; Pfeiffer, A.; Sihn, W.; Monostori, L. Lead time prediction using machine learning algorithms: A case study by a semiconductor manufacturer. *Procedia Cirp* **2018**, *72*, 1051-1056.

8. Poh, C.Q.; Ubeynarayana, C.U.; Goh, Y.M. Safety leading indicators for construction sites: A machine learning approach. *Automation in construction* **2018**, *93*, 375-386.

9. Gyulai, D.; Pfeiffer, A.; Nick, G.; Gallina, V.; Sihn, W.; Monostori, L. Lead time prediction in a flow-shop environment with analytical and machine learning approaches. *IFAC-PapersOnLine* **2018**, *51*, 1029-1034.

10. Jeong, J.H.; Woo, J.H.; Park, J. Machine Learning Methodology for Management of Shipbuilding Master Data. *International Journal of Naval Architecture and Ocean Engineering* **2020**, *12*, 428-439.

11. Ahmad, A.S.; Hassan, M.Y.; Abdullah, M.P.; Rahman, H.A.; Hussin, F.; Abdullah, H.; Saidur, R. A review on applications of ANN and SVM for building electrical energy consumption forecasting. *Renewable and Sustainable Energy Reviews* **2014**, *33*, 102-109.

12. Duan, K.; Keerthi, S.S.; Poo, A.N. Evaluation of simple performance measures for tuning SVM hyperparameters. *Neurocomputing* **2003**, *51*, 41-59.

13. Lee, A.; Geem, Z.W.; Suh, K.-D. Determination of optimal initial weights of an artificial neural network by using the harmony search algorithm: application to breakwater armor stones. *Applied Sciences* **2016**, *6*, 164.

14. Yu, T.; Cai, H. The Prediction of the Man-Hour in Aircraft Assembly Based on Support Vector Machine Particle Swarm Optimization. *Journal of Aerospace Technology and Management* **2015**, *7*, 19-30.

15. Wan, A.; Fang, J. Risk prediction of expressway PPP project based on PSO-SVM algorithm. In *ICCREM 2020: Intelligent Construction and Sustainable Buildings*; American Society of Civil Engineers Reston, VA: 2020; pp. 55-63.

16. Lv, Y.-j.; Wang, J.-w.; Wang, J.; Xiong, C.; Zou, L.; Li, L.; Li, D.-w. Steel corrosion prediction based on support vector machines. *Chaos, Solitons & Fractals* **2020**, *136*, 109807.

17. Pham, A.-D.; Hoang, N.-D.; Nguyen, Q.-T. Predicting compressive strength of high-performance concrete using metaheuristic-optimized least squares support vector regression. *Journal of Computing in Civil Engineering* **2016**, *30*, 06015002.

18. Sahoo, A.; Samantaray, S.; Ghose, D.K. Prediction of Flood in Barak River using Hybrid Machine Learning Approaches: A Case Study. *Journal of the Geological Society of India* **2021**, *97*, 186-198.

19. Tavakkoli, A.; Rezaeenour, J.; Hadavandi, E. A novel forecasting model based on support vector regression and bat meta-heuristic (Bat–SVR): case study in printed circuit board industry. *International Journal of Information Technology & Decision Making* **2015**, *14*, 195-215.

20. Barman, M.; Choudhury, N.B.D. Hybrid GOA-SVR technique for short term load forecasting during periods with substantial weather changes in North-East India. *Procedia computer science* **2018**, *143*, 124-132.

21. Mohamad, E.T.; Armaghani, D.J.; Momeni, E.; Abad, S.V.A.N.K. Prediction of the unconfined compressive strength of soft rocks: a PSO-based ANN approach. *Bulletin of Engineering Geology and the Environment* **2015**, *74*, 745-757.

22. Mostafaeipour, A.; Goli, A.; Qolipour, M. Prediction of air travel demand using a hybrid artificial neural network (ANN) with Bat and Firefly algorithms: a case study. *The Journal of Supercomputing* **2018**, *74*, 5461-5484.

23. Hussin, S.; Malek, M.A.; Jaddi, N.; Hamid, Z. Hybrid metaheuristic of artificial neural network—Bat algorithm in forecasting electricity production and water consumption at Sultan Azlan shah Hydropower plant. In Proceedings of the 2016 IEEE International Conference on Power and Energy (PECon), 2016; pp. 28-31.

24. Moayedi, H.; Gör, M.; Lyu, Z.; Bui, D.T. Herding Behaviors of grasshopper and Harris hawk for hybridizing the neural network in predicting the soil compression coefficient. *Measurement* **2020**, *152*, 107389.

25. Cortes, C.; Vapnik, V. Support-vector networks. *Machine learning* **1995**, *20*, 273-297.

26. Thissen, U.; Van Brakel, R.; De Weijer, A.; Melssen, W.; Buydens, L. Using support vector machines for time series prediction. *Chemometrics and intelligent laboratory systems* **2003**, *69*, 35-49.

27. Zhang, M.-G. Short-term load forecasting based on support vector machines regression. In Proceedings of the 2005 International Conference on Machine

Learning and Cybernetics, 2005; pp. 4310-4314.

28. Astudillo, G.; Carrasco, R.; Fernández-Campusano, C.; Chacón, M. Copper Price Prediction Using Support Vector Regression Technique. *Applied Sciences* **2020**, *10*, 6648.

29. Cao, B.; Liu, X.; Chen, W.; Yang, K.; Tan, P. Skid-proof operation of wheel loader based on model prediction and electro-hydraulic proportional control technology. *IEEE Access* **2019**, *8*, 81-92.

30. Vapnik, V.; Izmailov, R. Knowledge transfer in SVM and neural networks. *Annals of Mathematics and Artificial Intelligence* **2017**, *81*, 3-19.

31. Cao, L.-J.; Tay, F.E.H. Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on neural networks* **2003**, *14*, 1506-1518.

32. Smola, A.J.; Schölkopf, B. A tutorial on support vector regression. *Statistics and computing* **2004**, *14*, 199-222.

33. Vapnik, V. *The nature of statistical learning theory*; Springer science & business media: New York, NY, USA, 2013.

34. Yaseen, Z.M.; Kisi, O.; Demir, V. Enhancing long-term streamflow forecasting and predicting using periodicity data component: application of artificial intelligence. *Water resources management* **2016**, *30*, 4125-4151.

35. Sapankevych, N.I.; Sankar, R. Time series prediction using support vector machines: a survey. *IEEE Computational Intelligence Magazine* **2009**, *4*, 24-38.

36. Chang, C.-C.; Lin, C.-J. LIBSVM: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)* **2011**, *2*, 1-27.

37. Tbarki, K.; Said, S.B.; Ksantini, R.; Lachiri, Z. RBF kernel based SVM classification for landmine detection and discrimination. In Proceedings of the 2016 International Image Processing, Applications and Systems (IPAS), 2016; pp. 1-6.

38. Al Azies, H.; Trishnanti, D.; PH, E.M. Comparison of Kernel Support Vector Machine (SVM) in Classification of Human Development Index (HDI). *IPTEK Journal of Proceedings Series* **2019**, 53-57.

39. Thomas, S.; Pillai, G.; Pal, K. Prediction of peak ground acceleration using ϵ-SVR, ν-SVR and Ls-SVR algorithm. *Geomatics, Natural Hazards and Risk* **2017**, *8*, 177-193.

40. Patle, A.; Chouhan, D.S. SVM kernel functions for classification. In Proceedings of the 2013 International Conference on Advances in Technology and Engineering (ICATE), 2013; pp. 1-9.

41. Du, J.; Liu, Y.; Yu, Y.; Yan, W. A prediction of precipitation data based on support

vector machine and particle swarm optimization (PSO-SVM) algorithms. *Algorithms* **2017**, *10*, 57.

42. Zhang, Q.; Yu, H.; Barbiero, M.; Wang, B.; Gu, M. Artificial neural networks enabled by nanophotonics. *Light: Science & Applications* **2019**, *8*, 1-14.

43. Sheela, K.G.; Deepa, S.N. Review on methods to fix number of hidden neurons in neural networks. *Mathematical Problems in Engineering* **2013**, *2013*.

44. Eseye, A.T.; Zhang, J.; Zheng, D.; Li, H.; Jingfu, G. A double-stage hierarchical hybrid PSO-ANN model for short-term wind power prediction. In Proceedings of the 2017 IEEE 2nd international conference on cloud computing and big data analysis (ICCCBDA), 2017; pp. 489-493.

45. Boob, D.; Dey, S.S.; Lan, G. Complexity of training relu neural network. *Discrete Optimization* **2020**, 100620.

46. Pandya, V.; Areibi, S.; Moussa, M. A Handel-C implementation of the back-propagation algorithm on field programmable gate arrays. In Proceedings of the 2005 International Conference on Reconfigurable Computing and FPGAs (ReConFig'05), 2005; pp. 8 pp.-6.

47. Yang, T.; Asanjan, A.A.; Faridzad, M.; Hayatbini, N.; Gao, X.; Sorooshian, S. An enhanced artificial neural network with a shuffled complex evolutionary global optimization with principal component analysis. *Information Sciences* **2017**, *418*, 302-316.

48. Khashei, M.; Bijari, M. An artificial neural network (p, d, q) model for timeseries forecasting. *Expert Systems with applications* **2010**, *37*, 479-489.

49. Kartheeswaran, S.; Durairaj, D.D.C. A data-parallelism approach for PSO-ANN based medical image reconstruction on a multi-core system. *Informatics in Medicine Unlocked* **2017**, *8*, 21-31.

50. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the Proceedings of ICNN'95-international conference on neural networks, 1995; pp. 1942-1948.

51. Shi, Y.; Eberhart, R.C. Empirical study of particle swarm optimization. In Proceedings of the Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406), 1999; pp. 1945-1950.

52. Shi, Y.; Eberhart, R. A modified particle swarm optimizer. In Proceedings of the 1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360), 1998; pp. 69-73.

53. Ratnaweera, A.; Halgamuge, S.K.; Watson, H.C. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on evolutionary computation* **2004**, *8*, 240-255.

54. Ghasemi, M.; Aghaei, J.; Hadipour, M. New self-organising hierarchical PSO with jumping time-varying acceleration coefficients. *Electronics Letters* **2017**, *53*, 1360-1362.

55. Yang, X.-S. A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization (NICSO 2010)*; Springer: 2010; pp. 65-74.

56. Komarasamy, G.; Wahi, A. An optimized K-means clustering technique using bat algorithm. *European Journal of Scientific Research* **2012**, *84*, 263-273.

57. Yang, X.-S. *Nature-inspired metaheuristic algorithms*; Luniver press: 2010.

58. Saremi, S.; Mirjalili, S.; Lewis, A. Grasshopper optimisation algorithm: theory and application. *Advances in Engineering Software* **2017**, *105*, 30-47.

59. Wang, G.-G. Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems. *Memetic Computing* **2018**, *10*, 151-164.

60. Fushiki, T. Estimation of prediction error by using K-fold cross-validation. *Statistics and Computing* **2011**, *21*, 137-146.

61. Liu, T.; Yin, S. An improved particle swarm optimization algorithm used for BP neural network and multimedia course-ware evaluation. *Multimedia Tools and Applications* **2017**, *76*, 11961-11974.

62. Hasanipanah, M.; Armaghani, D.J.; Khamesi, H.; Amnieh, H.B.; Ghoraba, S. Several non-linear models in estimating air-overpressure resulting from mine blasting. *Engineering with Computers* **2016**, *32*, 441-455.

63. Amiri, M.; Amnieh, H.B.; Hasanipanah, M.; Khanli, L.M. A new combination of artificial neural network and K-nearest neighbors models to predict blast-induced ground vibration and air-overpressure. *Engineering with Computers* **2016**, *32*, 631-644.

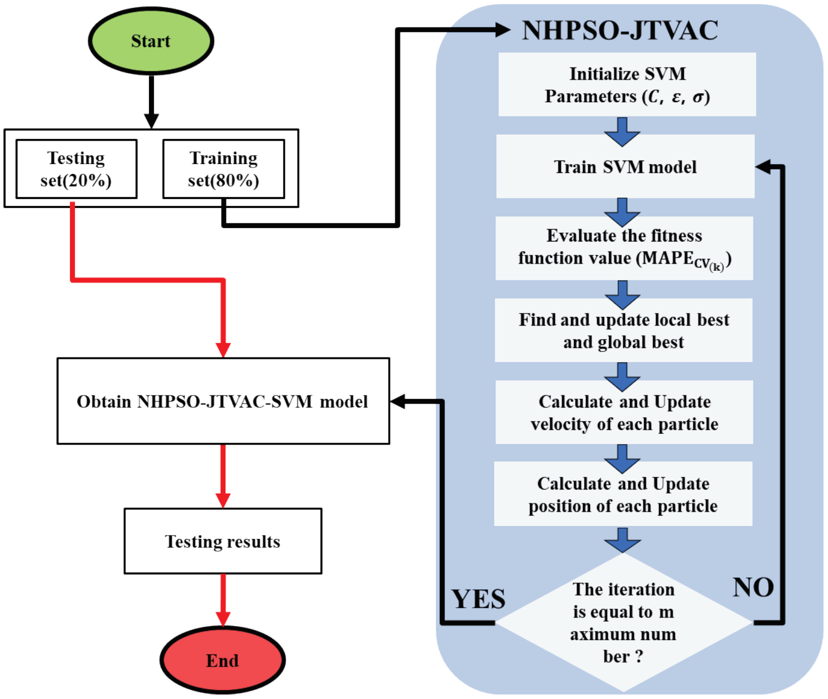
64. Ghasemi, E. Particle swarm optimization approach for forecasting backbreak induced by bench blasting. *Neural Computing and Applications* **2017**, *28*, 1855-1862.

65. Hasanipanah, M.; Noorian-Bidgoli, M.; Armaghani, D.J.; Khamesi, H. Feasibility of PSO-ANN model for predicting surface settlement caused by tunneling. *Engineering with Computers* **2016**, *32*, 705-715.

66. Cai, J.; Luo, J.; Wang, S.; Yang, S. Feature selection in machine learning: A new perspective. *Neurocomputing* **2018**, *300*, 70-79.

67. Lewis, C.D. *Industrial and business forecasting methods: A practical guide to exponential smoothing and curve fitting*; Butterworth-Heinemann: Oxford, UK, 1982.

Appendix A



Start
Testing set(20%)
Training set(80%)
NHPSO-JTVAC
Initialize SVM Parameters (C, ε, σ)
Train SVM model
Evaluate the fitness function value ($MAPE_{CV_{(k)}}$)
Find and update local best and global best
Calculate and Update velocity of each particle
Calculate and Update position of each particle
The iteration is equal to m aximum num ber ?
YES
NO
Obtain NHPSO-JTVAC-SVM model
Testing results
End

Figure A.1 Flow chart of NHPSO-JTVAC within SVM.

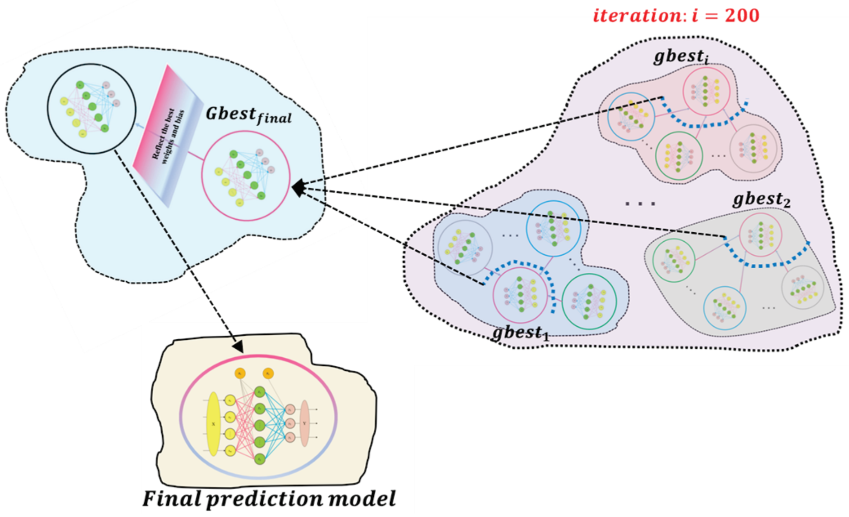


Figure A.2 Sketch map of NHPSO-JTVAC within ANN.

초 록

조선 산업에서 각 공정은 리드 타임을 가진다. 리드 타임이란 공정 시작과 종료 간에 시간으로, 고효율의 생산계획과 체계적 생산관리를 위해 매우 중요한 지표이다. 특히, 생산 계획 단계에서 정확한 리드타임 예측은 납기 준수를 위한 계획 수립을 위해 매우 중요하다. 그러나 기존의 예측 방법은 과거 데이터의 평균값을 사용했기 때문에 정확도가 매우 떨어졌다.

따라서 본 연구에서는 리드 타임과 다른 영향 요인 간의 복잡한 관계를 이해하기 위해 예측 분야에서 자주 적용되는 머신 러닝 (ML) 모델인 서포트 벡터 머신 (SVM) 및 인공 신경망 (ANN) 적용을 제안한다.

또한, 기계학습 모델 예측 정확도를 향상시키기 위해 메타 휴리스틱 알고리즘을 적용하여 모델의 파라미터를 최적화하고자 한다. 본 연구는 meta-heuristics-ANN, meta-heuristics-SVM 모델을 포함하는 하이브리드 모델을 구축한다. 더불어, 본 연구는 메타 휴리스틱 알고리즘 기반으로 최적화된 기계학습 모델의 성능을 서로 비교한다.

연구 결과를 통해, ML 모델의 파라미터를 탐색하는 과정에서 particle swam optimization (PSO)의 enhanced 버전인 NHPSO-JTVAC 알고리즘이 탐색 성능 면에서 다른 알고리즘보다 우수하다는 것을 알 수 있다. 뿐만 아니라 테스트 결과를 살펴보면 NHPSO-JTVAC에 기반한 하이브리드 모델이 조선소 세 개의 블록 공정 데이터에서 (각각 11.79%, 16.03% 및 16.45%) 가장 작은 MAPE 테스트 오차임을 알 수 있다. 이것은 NHPSO-JTVAC를 기반으로 구축된 모델이 예측 정확도 측면에서 의미 있는 향상을 더 달성할 수 있음을 보여준다.

전반적으로 NHPSO-JTVAC-SVM, NHPSO-JTVAC-ANN 모델은 조선소 블록 공정의 리드 타임을 예측하는 데 적합하다는 것을

확인할 수 있다.

주요어: 머신 러닝; 인공 지능 최적화 기법; 서포트 벡터 머신; 인공 신경망; 생산 리드 타임; 하이브리드 모델

학 번: 2019-25206