International Journal of Multiphysics
Volume 19, No. 1, 2025
ISSN: 1750-9548

Optimal Scheduling Strategy of Multi-physics Coupled Computing Resources Based on Machine Learning

Cheng Cheng [1,2,*], Zhiqing Lu [1,2], Kang Chen [1,2], Wei Guo [1,2], Defeng Liu [1,2] and Chong Huang [1,2]

China Ship Scientific Research Center; Wuxi 214082, China[1]

TAIHU Laboratory of Deepsea Technological Science, Wuxi 214082, China[2]

*Corresponding author

Abstract

Multi-physics coupled simulations imposes stringent demands on high-performance computing(HPC) resources, particularly regarding physical memory consumption. The inappropriate allocation of HPC resources may result in computational task failures or inefficient resource utilization. To enhance the utilization rate of the scarce large-memory computing resources in HPC clusters, this study explores the prediction methods for resource requirements and computation time of multi-physics coupling calculations, as well as rational allocation strategies for computing resources.By analyzing the characteristics of multi-physics coupling computational tasks and utilizing actual data collected from fluid-structure-acoustic(FSA) coupling computations, this study establishes an effective resource prediction model for FSA coupling calculations based on various machine learning algorithms. Subsequently, based on the forecasting results and HPC node configurations, an optimal allocation model for computing nodes is developed using an improved selection and elimination method based on weight estimation. This approach achieves a balanced optimization between computation time and resource allocation.Experimental results indicate that the proposed methods can effectively predict the resource requirements and computation time for multi-physics coupling calculations and provide optimal strategies for resource allocation, thereby resulting in an average decrease of 17.5% in computation time and an improvement of 20.4% in resource utilization efficiency.

Keywords: Computing resource prediction, machine learning, multi-physics coupling, resource allocation, multi-objective decision-making.

1. Introduction

The development of ships, aircraft, rockets, and other products is a highly complex system engineering task, involving multiple disciplines such as structure, fluid dynamics, acoustics, and thermodynamics. Each sub-system from these disciplines is often designed and simulated using different CAD/CAE software. However, with the growing complexity of technology and intensifying competition in these fields, traditional development methods that rely on ground testing and single-discipline simulation analysis are increasingly unable to meet the demands for high-precision and rapid development of complex products. Ignoring the coupling effects between multiple disciplines may lead to the loss of the optimal overall performance design.

Multi-physics coupled simulation based on unified modeling can ensure data consistency and transfer accuracy between different physical fields and enable the unified solution of different discipline performances, making it an ideal method for multi-physics coupled calculation. Yet, this approach involves building multi-physics solution matrices within the same software, causing an exponential increase in computational degrees of freedom and demanding substantial computational resources. Generally, parallel computing with large-memory computing nodes is used. Moreover, during the coupling process, physical field interactions may cause geometric

deformation (e.g., changes in the flow field due to structural vibration), requiring dynamic mesh reconstruction and synchronous updating of the coupling interface. This places extremely high demands on memory bandwidth and the real-time communication efficiency of computing nodes.

Currently, when requesting multi-physics coupled computing resources through high-performance scheduling systems, reliance on engineers' personal experience is common. To prevent computational task failures, excessive resources are often requested, making it difficult to access scarce large-memory computing nodes and potentially leading to project delays. Additionally, as the number of distributed computing nodes increases, so does the communication latency between them. Simply increasing the number of computing nodes may result in cross-node communication overhead that offsets parallel computing benefits. Therefore, accurately predicting multi-physics coupled computing resource requirements and computation time, and implementing reasonable resource allocation is of utmost importance.

To address these issues, this paper proposes a machine-learning-based method for predicting multi-physics coupled high-performance computing resource requirements and computation time, as well as an improved selection and elimination algorithm based on weight estimation for generating optimal high-performance computing resource allocation strategies, the overall technical approach is shown in Figure 1.

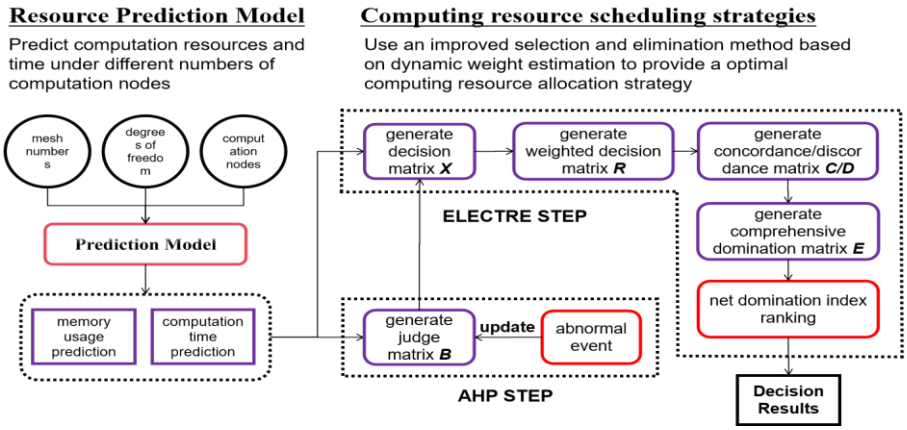


Figure 1. Overall technical approach

The effectiveness of the proposed method is verified using FSA coupled simulation calculations in COMSOL software[1], a multi-physics coupled simulation software commonly used in the shipping industry. The main contributions of this paper are two-fold:

1)A multi-physics coupled simulation computing resource prediction model is proposed. Based on machine learning algorithms and trained with COMSOL multi-physics coupled calculation example data, this model can quickly and accurately predict the actual computing resources and computation time required for computing tasks under different numbers of computing nodes.

2)An improved selection and elimination method based on weight estimation is proposed to obtain multi-physics coupled computing resource scheduling strategies. This strategy balances the conflict between physical memory usage and computation time, providing the optimal computing resource allocation strategy.

2. Related Work

The prediction of multi-physics coupled computing resource requirements and computation time is influenced by multiple factors, such as the number of computing nodes, grid cells in the model, degrees of freedom, solver algorithm types, and computational accuracy. It can be abstracted as a multi-input and multi-output predictive regression problem.

Most current research on multi-input and multi-output predictive regression models is based on neural networks, which perform well for tasks with large datasets. For example, LSTM has been used for precipitation prediction in [2]. Dynamic recurrent neural networks have been applied for structural response prediction in [3]. A CNN-Transformer hybrid framework has been proposed for predicting diesel vehicle ammonia emissions in [4].

Additionally, BP neural networks have been utilized to achieve accurate construction cost prediction for large-scale building projects in [5]. However, neural-network-based methods have significant limitations when the dataset is small. Even with data augmentation and network-structure adjustments to optimize the model, these limitations persist. Multi-physics coupled computing tasks, due to their long computation time and high resource demands, cannot generate large amounts of data in the short term. Thus, neural networks are not well-suited for predicting resource usage in multi-physics coupled computing task. For multi-physics coupled computing tasks, the ideal scenario is to complete the computation as quickly as possible using the fewest computing resources. Yet these two objectives are conflicting. A single node's memory is limited, while multi-physics coupled computing requires massive memory, typically necessitating parallel computing across multiple nodes. Reducing computation time requires more computing nodes. These conflicting goals lead to a multi-objective optimization problem. Multi-objective optimization problems are common in engineering and scientific research. They involve simultaneously optimizing two or more conflicting and related objective functions. Finding a solution that optimizes all functions simultaneously is usually impossible. Instead, researchers aim to find a set of trade-off solutions, known as the Pareto optimal set. Decision-makers then select the best solution based on the Pareto set and frontier, considering practical needs and preferences. Selection methods for the best solution have become a research hotspot. For instance, in some studies, multi-objective problems were transformed into single-objective ones by assigning weights to each goal in [6]. In others, solution quality was evaluated by calculating distances between reference points and solutions in [7]. Many scholars have also explored specific multi-objective programming problems. For example, a green credit multi-objective optimization model with risk minimization and environmental benefit maximization as goals has been developed in [8]. The NSGA-II algorithm has been used to study hydrogen fuel cell performance in [9]. Power grid investment planning decisions have been analyzed with an improved ELECTRE method in [10]. However, most research focuses on finance, electricity, and water resources, with limited attention to multi-physics coupled computing resource allocation.

3. Preliminary Knowledge

3.1. Machine Learning

Machine Learning(ML), a subfield of Artificial Intelligence(AI), enables computer systems to automatically learn patterns from data and make predictions or decisions without explicit programming. Its primary objective is to extract patterns from data and generalize them to new data.

3.1.1. Common Machine Learning Algorithms

(1) Supervised Learning

Supervised learning involves training models with labeled datasets and is suitable for classification and regression problems.Typical algorithms include linear regression, decision trees(see [11]), and support vector machines(see [12]) et al.

The random forest regression algorithm(see [13]), a tree-based ensemble learning method, makes predictions by constructing multiple decision trees and aggregating their results, as shown in Figure 2.

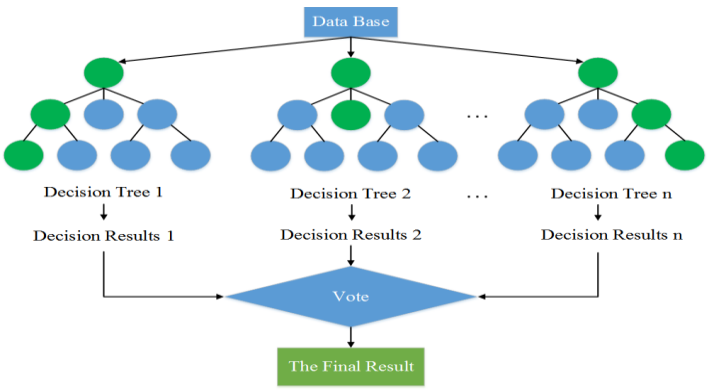


Figure 2. Schematic diagram of the Random Forest algorithm

Support Vector Regression(see [14]) is a regression algorithm based on Support Vector Machines. Its core idea is to find an optimal hyperplane in a high-dimensional feature space that minimizes the sum of distances from all sample points to the hyperplane, as shown in Figure 3.

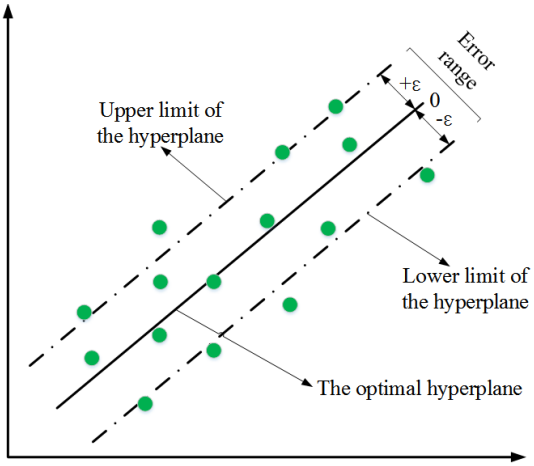


Figure 3. Schematic diagram of the Support Vector Regression algorithm

Gradient Boosting Regression(see [15]) is an ensemble-based regression algorithm. It combines multiple weak learners to form a strong learner, enhancing prediction accuracy. In each iteration, it fits the residuals of the previous prediction to gradually reduce the prediction error of the overall model, as shown in Figure 4.

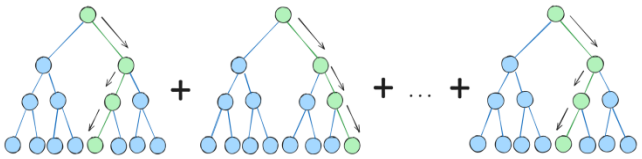


Figure 4. Schematic diagram of the Gradient Boosting Regression algorithm

(2)Unsupervised Learning

Unsupervised learning analyzes the internal structure of unlabeled data and is commonly used for clustering and dimensionality reduction. Typical algorithms include K-means clustering(see [16]) and Principal Component Analysis (PCA)(see [17]) et al.

3.1.2. Common Evaluation Metrics for Machine Learning Algorithms

In machine learning, evaluation metrics are crucial tools for assessing the predictive performance of models. By quantifying the discrepancy between predicted and actual values, these metrics help researchers evaluate the accuracy, robustness, and generalization ability of models. The commonly used evaluation metrics in machine learning are as follows:

(1) Mean Squared Error (MSE)

MSE is one of the most commonly used evaluation metrics for regression tasks. It calculates the average of the squares of the differences between predicted and actual values, with the formula as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (1)$$

In the formula, y_i represents the actual value,$\hat{y}_i$ denotes the predicted value, and n is the number of samples.

MSE is more sensitive to larger errors, making it suitable for scenarios where significant deviations from predictions need to be penalized.

(2) Root Mean Squared Error (RMSE)

RMSE is the square root of MSE, with the formula as follows:

$$RMSE = \sqrt{MSE} \tag{2}$$

It retains the sensitivity of MSE to larger errors and has the same dimension as the original data, making it easier to interpret. RMSE is widely used in regression tasks, especially when a clear understanding of errors is needed.

(3)Mean Absolute Error (MAE)

MAE calculates the average of the absolute differences between predicted and actual values, with the formula as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3}$$

In the formula, y_i represents the actual value,$\hat{y}_i$ denotes the predicted value, and n is the number of samples.

Unlike MSE and RMSE, MAE is less sensitive to outliers, making it suitable for datasets with anomalies.

(4)Mean Absolute Percentage Error (MAPE)

MAPE expresses prediction errors in percentages, with the formula as follows:

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{4}$$

In the formula, y_i represents the actual value,$\hat{y}_i$ denotes the predicted value, and n is the number of samples.

It is useful for comparing errors across different data scales but can become unstable when actual values are close to zero.

(5)Mean Squared Logarithmic Error (MSLE)

MSLE evaluates model performance by calculating the square of the difference between the logarithms of predicted and actual values, with the formula as follows:

$$MSLE = \frac{1}{n}\sum_{i=1}^{n}\left(\ln(y_{pred_i}+1) - \log(y_{true_i}+1)\right)^2 \tag{5}$$

In the formula,$y_{\mathrm{pred_i}}$ represents the predicted value of the i-th sample,$y_{\mathrm{true_i}}$ denotes the actual value of the i-th sample,n is the number of samples.

It is suitable for regression tasks with non-negative target values across a wide range.

3.2. Multi-Objective Optimization Methods

3.2.1. Analytic Hierarchy Process (AHP)

AHP(see [18]) is a decision-making method based on hierarchical structures, suitable for complex problems with multiple objectives and criteria. It determines relative weights through pairwise comparisons to achieve a global optimal solution. The implementation steps are as follows:

(1)Hierarchy Model Construction: Decompose the decision problem into a goal layer(such as resource scheduling optimization)、criterion layer(such as computation time and memory utilization) and alternative layer, forming a tree-like hierarchy.

(2)Judgment Matrix Generation: Use the 1-9 scale method(as shown in Table 1) to compare elements within the same level, constructing a judgment matrix $B = (b_{ij})_{n\times n}$, where b_{ij}represents the relative importance of element i compared to element j.

Table 1. Ratio Scale

Value	Description
1	Equally important
3	Moderately more important
5	Significantly more important
7	Much more important
9	Extremely more important

2,4,6,8	Intermediate values

(3)Weight Calculation and Consistency Check: Solve the judgment matrix to obtain the maximum eigenvalue λ_{max} and eigenvector w, normalize to get the weight vector. Consistency of the matrix is verified using the Consistency Ratio (CR), which must be less than 0.1 to ensure the judgment matrix is acceptable. The calculation formula is as follows:

$$\begin{cases} CR = \frac{CI}{RI} \\ CI = \frac{\lambda_{max} - n}{n-1} \end{cases} \tag{6}$$

(4)Global Weight Synthesis: Aggregate weights across levels to compute the overall priority vector for alternatives, completing the consistency check.

(5)Decision Making: Select the optimal alternative or rank alternatives based on the overall priority results.

3.2.2. Elimination Et Choice Translating Reality (ELECTRE) Method

ELECTRE(see [19]) is a multicriteria decision-making method that screens and ranks alternatives through concordance and discordance matrices. The steps are as follows:

(1)Decision Matrix Construction: Form an $m \times n$ decision matrix X based on n criteria, where x_{ij} represents the matrix element, indicating the score of the i-th alternative on the j-th criterion; m is the number of alternatives.

(2)Data Standardization: Normalize the decision matrix X using the range method to eliminate dimensional differences,an m×n order standardized matrix X′ is obtained. where x'_{ij} is the matrix element, representing the normalized value of the j-th criterion for the i-th alternative. The normalization formula is as follows:

$$x'_{ij} = \frac{\max\limits_{i\epsilon(1,m)} x_{ij} - x_{ij}}{\max\limits_{i\epsilon(1,m)} x_{ij} - \min\limits_{i\epsilon(1,m)} x_{ij}} \tag{7}$$

(3)Weighted Decision Matrix: Incorporate the weights of n criteria into the standardized matrix to obtain a $m \times n$ order weighted decision matrix R. r_{ij} is the element of matrix R, represents the weighted value of the j-th criterion for the i-th alternative. The formula is as follows:

$$r_{ij} = x'_{ij}\omega_j \tag{8}$$

In the formula,ω_j represents the weight of the j-th criterion.

(4)Concordance and Discordance Sets: Define concordance sets $H_{kl} = \{j | r_{kj} \geq r_{lj}\}$ and discordance sets $B_{kl} = \{j | r_{kj} < r_{lj}\}$ based on alternative pairs (k, l).

(5)Concordance Matrix C: Based on the concordance set H_{kl}, construct an $m \times m$ order concordance matrix C to characterize the proportion of criteria weights where alternative k is better than alternative l. Here,c_{kl} is the element of this matrix, representing the concordance index of alternative k over alternative l. The calculation formula for c_{kl} is as follows:

$$c_{kl} = \frac{\sum_{j\in H_{kl}} \omega j}{\sum_{j\in J} \omega j} \tag{9}$$

The greater the value of c_{kl}, the more significantly the criteria weight of alternative k exceeds that of alternative l.

(6)Discordance Matrix D: Based on the discordant set B_{kl}, construct an $m \times m$ order discordant matrix D to reflect the maximum local differences where alternative k is inferior to alternative l. Here,d_{kl} is the element of this matrix, represents the discordant index of alternative k with respect to alternative l. The calculation formula for d_{kl} is as follows:

$$d_{kl} = \frac{\max_{j \in B_{kl}} |r_{kj} - r_{lj}|}{\max_{j \in J} |r_{kj} - r_{lj}|} \tag{10}$$

The larger the d_{kl} value is, the more inferior the criterion scores of alternative k are to those of alternative l.

(7)Comprehensive Domination Matrix E: Based on the concordance matrix C and the discordance matrix D, the $m \times m$ order comprehensive dominance matrix E is determined. This matrix reflects the relative advantages and disadvantages of different alternatives under multi-criteria decision making. e_{kl} is the element of matrix E, represents the dominance index of alternative k over alternative l. The calculation formula for e_{kl} is as follows:

$$e_{kl} = c_{kl} - d_{kl} \tag{11}$$

The larger the e_{kl} value, the greater the dominance strength of alternative k over alternative l, indicating that alternative k is superior to alternative l in the evaluation.

(8)Net Domination Index Calculation: Based on the comprehensive dominance matrix, the net dominance index ξ_k for each alternative is calculated using the following formula:

$$\xi_k = \sum_{i=l \text{⍰} i \neq k}^{m} (e_{ki} - e_{ik}), k = 1,2, \dots, m \tag{12}$$

The ξ_k value reflects the overall evaluation score of an alternative under criterion integration. A larger ξ_k indicates a better overall evaluation.

(9)Net Domination Ranking: Sort the alternatives in descending order of ξ_k. Based on the ranking results, carry out multi-physics coupled simulation computation resource scheduling strategy optimization.

4. Machine Learning-Based Multi-Physics Coupled Simulation Resource Prediction Model

4.1. Data Collection and Preprocessing

Submit different multi-physics coupled models to a high-performance scheduling system. By selecting various numbers of large-memory nodes (96 CPU cores and 1.5TB memory per node), obtain model computation results under different resource conditions. The dataset includes 30 multi-physics coupled 3D models. Each model series has a mesh quantity ranging from 5 million to 50 million and a computing node count from 5 to 25. Data is stored in Excel, totaling 1,500 groups, with 70% for training and 30% for testing. The random shuffling rate is 42%. Some samples in the dataset are shown in Table 2.

Table 2. Some Samples of the Collected Data

Model Number	Mesh Count (in ten thousands)	Degrees of Freedom (in ten thousands)	Number of Compute Nodes (in units)	Computation Time (in seconds)	Physical Memory (in GB)	Virtual Memory (in GB)
1	7371	11800	14	10674	1047	1110
1	5407	7476	12	7920	524	590
2	4479	14000	22	13950	1016	1152
2	3134	9811	12	12757	926	1046
3	2168	6943	7	10615	1046	1206
3	1943	5883	9	8237	693	831
4	4589	7060	12	8572	625	681
4	4589	7060	11	7681	701	723
5	2214	5063	9	5104	453	521
5	2214	5063	7	5638	677	762
6	3596	5116	20	8016	600	630
6	2870	3997	11	8429	601	629
7	3323	6533	20	10107	718	805
7	1665	3360	7	7778	912	944
8	3959	5891	20	9854	663	695
8	1754	2536	7	7171	538	573
9	1595	3356	9	6149	579	606

9	1190	2811	9	4330	348	431
10	2082	5411	20	5619	397	419
10	1188	2804	11	4754	447	477

In Table 2, the first column represents different multi-physics coupling models. The second and third columns show the mesh count and degrees of freedom for these models under different meshing methods. The node count indicates the actual number of nodes used when submitting jobs. The mesh count, degrees of freedom, and node count serve as inputs for the computational resource prediction model, while computation time and physical memory usage are the outputs.

The distribution of data based on model mesh and degrees of freedom is illustrated in Figures 5 and 6. It is evident that the mesh count and degrees of freedom in multi-physics coupling tasks are significantly large. Over 75% of the data have a mesh count exceeding 15 million, and over 80% have degrees of freedom exceeding 25 million.

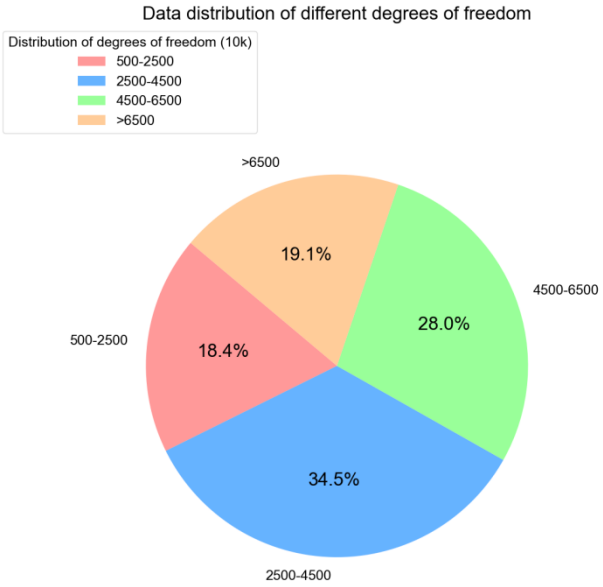


Figure 5. Data distribution of different grid quantitties

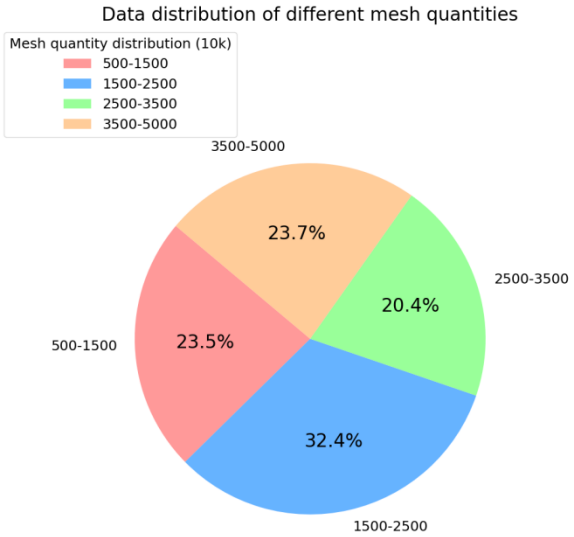


Figure 6. Data distribution of different degrees of freedom

4.2. Feature Selection and Extraction

The computational resource requirements and computation time of multi-physics coupling computational models are mainly related to the number of computational nodes allocated, the mesh count of the model, the number of

degrees of freedom, the solution algorithm, and the computational accuracy. Considering that the default MUMPS(see [20]) solver is usually chosen in practical computations, which has high robustness, is effective in solving multi-physics coupling problems, and typically does not require adjustment of computational accuracy. Therefore, the cluster node count, mesh count, and degrees of freedom of the multi-physics coupling model are selected as the input features of the prediction model, while the output features of the model are computation time, physical memory usage, and virtual memory usage.

4.3. Prediction Model Construction

The multi-physics coupling computational resource prediction problem has the following characteristics:

1)Low-dimensional input features: The number of features such as node count, mesh count, and degrees of freedom is limited, making it suitable for lightweight machine learning models;

2)Limited data volume: Since multi-physics coupling computational tasks require a large amount of computational resources and longer computation time, the number of datasets is limited, and a large amount of data cannot be generated for training deep learning algorithms;

3)Significant nonlinear relationships: Computational resource requirements and computation time change non-linearly with the increase in node count in a segmented manner (such as the diminishing marginal returns caused by communication overhead).

In view of the above characteristics, machine learning regression algorithms are particularly suitable for nonlinear regression problems with limited datasets. Random forest regression reduces variance by integrating multiple decision trees, has strong nonlinear relationship modeling capabilities, and supports feature importance analysis. Support vector regression (SVR) maps data to high-dimensional spaces using kernel functions to handle nonlinear regression and is suitable for small-sample scenarios. Gradient boosting regression (GBR) optimizes predictions by iteratively fitting residuals and is robust to outliers. Therefore, these three machine learning regression algorithms were chosen to construct the prediction models for computational resource usage and computation time.

(1)FSA Coupling Simulation Computational Resource Prediction Model Based on Random Forest Regression

The Random Forest Regressor class from the scikit-learn package(see [21]) was used to construct the model, with the following parameters needing adjustment:

- n_estimators: the number of decision trees. Increasing the number of trees can enhance model performance but will increase training time and memory consumption.
- max_depth: the maximum depth of the tree. It controls tree complexity and prevents over-fitting. It can be 'None' (unrestricted depth) or a positive integer.
- min_samples_split: the minimum number of samples required to split an internal node. It controls tree growth and prevents over-fitting.

The actual training parameters are shown in Table 3.

Table 3. Random Forest Regressor Algorithm Training Parameters

Training Parameters	Value
n_estimators	Range（50,50,1000）
max_depth	None，range（10,10,100）
min_samples_split	Range（2,2,10）
error_function	MSE,MAE,MSLE

During training, a combination of Bayesian optimization and cross-validation was used to determine the optimal parameters, and the optimal parameters obtained are shown in Table 4.

Table 4. Optimal Training Parameters of Random Forest Regressor Algorithm

Optimal Training Parameters	Value
n_estimators	800
max_depth	None
min_samples_split	4
error_function	MAE

Feature importance analysis was conducted on the training results, and the analysis results are shown in Figure 7. As can be seen from the figure, the number of degrees of freedom is the most important feature affecting the prediction results. this is due to the fact that it directly determines the dimension of the equation system during CAE solution, thereby influencing the solution time and resource requirements.

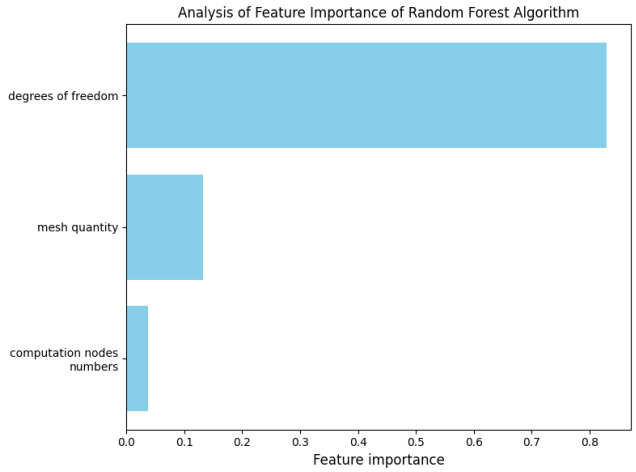


Figure 7. Feature Importance Analysis

The prediction errors of the random forest regression algorithm under optimal training parameters are shown in Table 5.

Table 5. Prediction Error of Random Forest Regressor Algorithm

Prediction Item	Error Function	
	MAE	MAPE
Computation Time	625.7	4.8%
Physical Memory Usage	75.4	6.4%
Virtual Memory Usage	84.9	4.8%

As shown in the table, the MAPE values for the three prediction items are all within 7%. Figure 8 presents a comparison of the predicted and actual values of 30 randomly selected test samples from the test set.

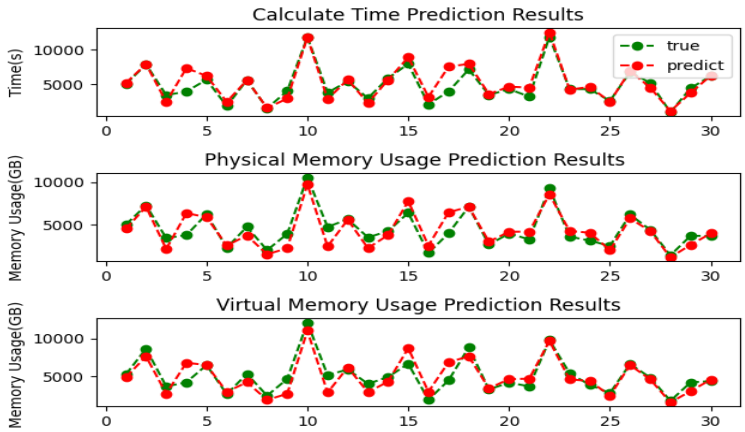


Figure 8. Comparison of predicted results and actual results by the Random Forest algorithm

(2)SVR Based FSA Coupling Simulation Resource Prediction Model

This study aims to predict computational resource requirements and computation time by finding a decision boundary that minimizes the distance between predicted and actual values. The SVR class from the scikit-learn package was used to construct the model. The parameters to be adjusted are as follows:

➢ C: Balances model complexity and training set fit. A large C leads to over-fitting, while a small C causes underfitting.

➢ ε: Defines the maximum acceptable error between predicted and actual values. A proper ε enhances the model's generalization ability.

➢ Kernel function and its parameters: Different kernel functions suit different data distributions. Common choices include linear, polynomial, and RBF kernels.

The actual training parameters are shown in Table 6.

Table 6. SVR Algorithm Training Parameters

Training Parameters	Value
C	Range（0.05,0.1,1）
ε	Range（0.01,0.1,1）
kernel	Linear,rbf
error_function	MSE,MAE,MSLE

During training, a combination of Bayesian optimization and cross-validation was used to determine the optimal parameters, and the optimal parameters obtained are shown in Table 7.

Table 7. Optimal Training Parameters of SVR

Optimal Training Parameters	Value
C	1
ε	1
kernel	Linear
error_function	MAE

The prediction error of the SVR algorithm under the optimal training parameters are as shown in Table 8.

Table 8. Prediction Error of SVR Algorithm

Prediction Item	Error Function	
	MAE	MAPE
Computation Time	938.7	15.0%
Physical Memory Usage	928.5	11.5%
Virtual Memory Usage	1017.8	9.5%

As shown in the table, the MAPE values for the three prediction items are all within 15%. Figure 9 presents a comparison of the predicted and actual values of 30 randomly selected test samples from the test set.

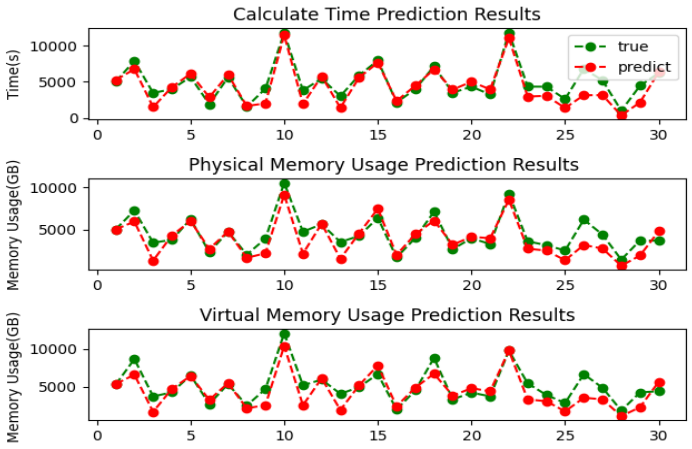


Figure 9. Comparison of SVR Algorithm Predicted Values and Actual Values.

(3)GBR Based FSA Coupling Simulation Computational Resource Prediction Model

The GBR class from the scikit-learn package was used to construct the model. The parameters to be adjusted are as follows:

- n_estimators: the number of iterations. Increasing this value can make the model more complex but may also increase the risk of over-fitting.
- learning_rate: determines the contribution of each new learner in the iteration. A smaller learning rate can make the model more robust but may increase training time and the number of iterations needed.
- max_depth: the maximum depth of each weak learner, which controls model complexity and prevents over-fitting.
- min_samples_split: the minimum number of samples required to split a node, which controls decision tree growth.

The actual training parameters are shown in Table 9.

Table 9. GBR Algorithm Training Parameters

Training Parameters	Value
n_estimators	Range（50,50,1000）
max_depth	None，range（10,10,100）
min_samples_split	Range（2,2,10）
Learning_rate	[0.01,0.05,0.1]
error_function	MSE,MAE,MSLE

During training, a combination of Bayesian optimization and cross-validation was used to determine the optimal parameters, and the optimal parameters obtained are shown in Table 10.

Table 10. Optimal Training Parameters of GBR

Optimal Training Parameters	Value
n_estimators	50
max_depth	None
min_samples_split	10
Learning_rate	0.05
error_function	MAE

The prediction error of the GBR algorithm under the optimal training parameters are as shown in Table 11.

Table 11. Prediction Error of GBR Algorithm

Prediction Item	Error Function	
	MAE	MAPE
Computation Time	652.3	5.7%
Physical Memory Usage	770.5	7.7%
Virtual Memory Usage	854.2	6.5%

As shown in the table, the MAPE for the three prediction items are all within 8%. Figure 10 presents a comparison of the predicted and actual values of 30 randomly selected test samples from the test set.

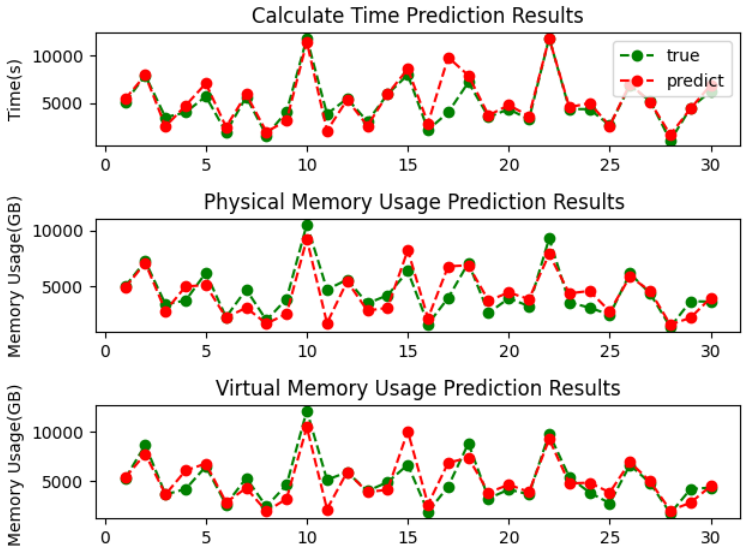
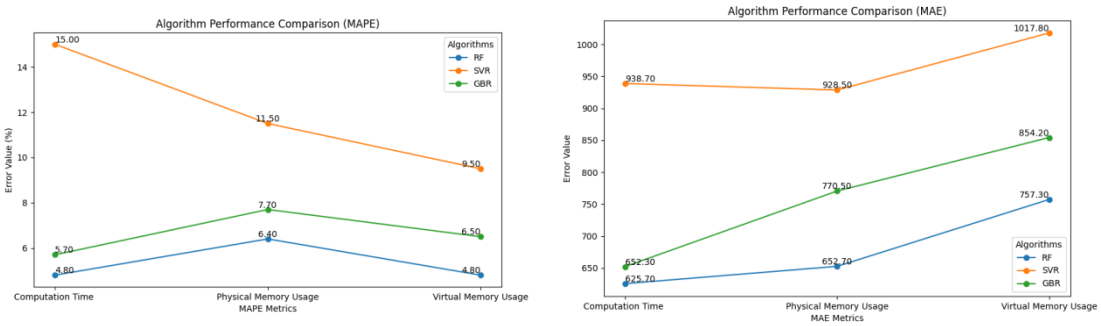


Figure 10. Comparison of GBR Algorithm Predicted Values and Actual Values.

4.4. Model Training Results Analysis

By comparing the error results of the three algorithms, it can be seen from Figure 11 that the Random Forest Regressor algorithm performs the best. Therefore, the Random Forest Regressor algorithm will be used as the predictive model for subsequent tasks.



(a) (b)

Figure 11. Compare of three algorithms : (**a**) MAPE; (**b**) MAE.

5. Multi-physics Coupling Simulation Computing Resource Scheduling Strategy based on the Improved AHP-ELECTRE model

5.1. Improvement strategies of the AHP-ELECTRE fusion model

Aiming at the problem of resource scheduling in multi-physics coupled simulation computing, a hybrid decision-making model based on dynamic weight estimation is proposed. The decision-making steps of this model are as follows:

(1) Problem modeling

For the multi-physics coupling simulation computing resource scheduling problem in this study, its decision-making goal is to optimize the computing time and physical memory utilization rate on the premise of being able to complete the multi-physics coupling computing tasks normally. The reason why the usage of virtual memory is not considered here is that virtual memory can generally be converted from the disk space of the computing node, and it is usually not a factor restricting whether the multi-physics coupling computing task can be completed normally. Based on this, the mathematical expression of the objective function of the optimal scheduling strategy for multi-physics coupling simulation computing is as follows:

$$\begin{cases} \min f_1 = t \\ \max f_2 = \frac{u_p}{u_t} \\ s.t. \begin{cases} t \geq 0 \\ 0 \leq u_p \leq 1536 \end{cases} \end{cases} \tag{13}$$

Among them, $f_1$represents the objective function of computing time, and t represents the computing time, with the unit of seconds; f_2 represents the objective function of physical memory usage, u_p represents the predicted physical memory usage of a single node, with the unit of GB, and u_t represents the maximum physical memory usage of a single node, which is 1536GB in the experimental environment of this paper.

(2) Dynamic weight correction of AHP

The weights of the traditional AHP are static values, assuming that the decision-making environment is stable and the target priority is fixed. However, in dynamic scenarios such as multi-physics coupled computing resource scheduling, task types, environmental loads, and user preferences may change in real time, and static weights are difficult to adapt to time-varying demands. For example:

● Task type changes: Steady-state simulation and transient simulation have different priorities for 'computing time' and 'memory usage rate'

● Resource fluctuation: When cluster nodes fail or new nodes are added, the weights need to be dynamically adjusted to optimize resource allocation

● User preference migration: The requirements of different projects for time sensitivity or cost constraints may be dynamically adjusted according to users' preference

Static weights in these scenarios can lead to the rigidity of the scheduling strategy and prevent the realization of global optimality.

The dynamic weight correction mechanism aims to dynamically update the judgment matrix and weight vector of AHP by providing real-time feedback data or environmental change signals, thereby enhancing the adaptability of the decision-making model. The core idea is to extend AHP from offline weight calculation to online dynamic optimization, which specifically includes the following steps:

1) Dynamic signal perception

Define the dynamic events that trigger weight correction, for example:

● Task type switching (such as from fluid-structure coupling to thermal coupling)

● The resource usage rate exceeds the threshold (such as the memory usage rate persistently being higher than 95%).

● The user manually adjusts the priority (such as temporarily requesting to shorten the calculation cycle)

2) Judgment matrix online updates

Adjust the element values of the criterion layer judgment matrix B according to dynamic events. For example, when the task transitions to transient simulation, users pay more attention to the computing time, and the importance ratio of 'computing time' relative to 'physical memory usage rate' needs to be increased.

Suppose the original judgment matrix is:

$$\mathbf{B} = \begin{bmatrix} 1 & b_{12} \\ b_{21} & 1 \end{bmatrix} \tag{14}$$

After the dynamic event is triggered, it is updated in the following steps:

● Rule-driven: Preset Events-Weight mapping table (for example, changing $b_{12} = 1/2$ to $b_{12} = 1/3$ in a transient task)

● Data-driven: Train regression models based on historical task data to predict the optimal b_{ij} under new events

3) Weight recalculation and consistency check

The updated judgment matrix needs to recalculate the weight vector $W = [w_1, w_2]$, and verify the consistency ratio (CR). If CR exceeds the limit, iterative optimization or manual intervention will be triggered. The eigenvector method is adopted to solve: $B \cdot W = \lambda_{max} \cdot W$, and the dynamic weights are obtained through normalization.

4) Smooth transition of weights

To avoid strategy oscillations caused by sudden changes in weights, the exponential smoothing method is introduced to gradually adjust the weights:

$$W_t = \alpha \cdot W_{new} + (1 - \alpha) \cdot W_{t-1} \tag{15}$$

Among them, $\alpha \in (0,1)$ is the smoothing coefficient, which controls the rate of change of the weight.

5.2. The generation process of resource scheduling strategies through multi-physics coupling simulation calculation

Taking the two criteria (calculation time f_1 and memory usage rate f_2 as examples, the mathematical process of dynamic weight correction is demonstrated:

1) Initial judgment matrix and static weights

The initial matrix is:

$$\mathbf{B}_0 = \begin{bmatrix} 1 & 2/3 \\ 3/2 & 1 \end{bmatrix} \tag{16}$$

It is calculated by the eigenvector method that $W_0 = [0.4, 0.6]$.

2) Dynamic events trigger matrix updates

Suppose a 'memory resource shortage' event occurs, it is necessary to enhance the importance of memory usage. Update the judgment matrix to:

$$\mathbf{B}_1 = \begin{bmatrix} 1 & 1/2 \\ 2 & 1 \end{bmatrix} \tag{17}$$

Resolve the eigenvector to obtain $W_1 = [0.33, 0.67]$.

3) Consistency check and smooth transition

Calculate $\lambda_{max} = 2.0$, $CI = 0$, $CR = 0$ (meeting the consistency requirements). If the smoothing coefficient $\alpha = 0.7$, then the updated weight $W_t = 0.7 \cdot [0.33, 0.67] + 0.3 \cdot [0.4, 0.6] = [0.35, 0.65]$.

4) Embed the ELECTRE decision-making process

Substitute the dynamic weight W_t into the weighted decision matrix R of ELECTRE, update the harmony matrix C, disharmony matrix D and comprehensive dominance matrix E, and finally generate the scheduling strategy adapted to the new weight.

6. Results and Discussion

6.1. Experimental environment

In this experiment, the computing resource prediction model of multi-physics coupling simulation generated in Section 3.4 is used to predict the required computing time and resource usage under different node numbers, and the optimal computing resource scheduling strategy is obtained by screening according to the prediction results. The experiment adopted Excel 2016 software and the Python3.7 environment to implement the solution in a 64-bit Windows7 computer. The overall running time of the program was within 1 second.

6.2. Experimental process

15 different models were randomly selected. The multi-physics coupling simulation computing resource prediction model provided the computing time and resource usage of these 15 models under different numbers of computing nodes and stored them in the form of Excel table data. Due to space limitations, the prediction results of one of the models are listed in Table 12.

Table 12. The predictions of one model

Number of mesh ($\times 10^4$)	Number of freedom ($\times 10^4$)	Number of Compute Nodes	Calculation time (s)	Physical memory usage (GB)	Physical memory usage rate (%)
3176	5974	5	13250	1189	77.4%
3176	5974	7	11023	1142	74.3%
3176	5974	9	8532	1101	71.7%
3176	5974	11	8400	1006	65.5%
3176	5974	13	7864	843	54.9%
3176	5974	15	7067	801	52.1%
3176	5974	17	7028	782	50.9%
3176	5974	19	7123	763	49.7%
3176	5974	21	7236	725	47.2%
3176	5974	23	7889	710	46.2%
3176	5974	25	7927	689	44.9%

It can be seen from the data in the table that the computing time and the physical memory usage rate are contradictory. When the number of nodes involved in the computing task increases, the computing time will decrease, but the physical memory usage rate is constantly declining. Moreover, when there are too many nodes for parallel computing, the communication overhead between nodes will instead increase the computing time.

Suppose that when submitting the assignment, more attention is chosen to be paid to the reduction of calculation time, and an initial judgment matrix is formed based on the ratio scale:

$$B_0 = \begin{bmatrix} 1 & 3/2 \\ 2/3 & 1 \end{bmatrix} \tag{18}$$

The initial weight $W_0 = [0.6,0.4]$ is calculated by the feature vector method.

Substitute the initial weight W_0 into the weighted decision matrix R of ELECTRE, calculate the harmony matrix C, disharmony matrix D and comprehensive dominance matrix E, thereby obtaining the net dominance index ξ_k.

All the schemes were ranked according to the net dominance index ξ_k, and the final ranking results are shown in Table 13 as follows:

Table 13. The decision result of multi-physics coupling simulation computing resource scheduling focusing on computing time

Compute node usage	Calculation time (s)	Physical memory usage rate (%)	ξ_k	Rank
5	13250	77.4%	-6.54	9
7	11023	74.3%	-3.72	8
9	8532	71.7%	4.13	3
11	8400	65.5%	1.44	6
13	7864	54.9%	1.79	5
15	7067	52.1%	11.02	1
17	7028	50.9%	10.17	2
19	7123	49.7%	3.95	4
21	7236	47.2%	-1.01	7
23	7889	46.2%	-8.53	10
25	7927	44.9%	-12.70	11

It can be seen from the results in the table that ξ_k is the highest when the node usage is 15, indicating that choosing the number of nodes at 15 in this task is the optimal strategy.

Then submit another assignment. The prediction results of this assignment are shown in Table 14 as follows:

Table 14. The predictions of one model

Number of mesh ($\times 10^4$)	Number of freedom ($\times 10^4$)	Number of Compute Nodes	Calculation time (s)	Physical memory usage (GB)	Physical memory usage rate (%)
2214	5063	5	11050	1032	67.2%
2214	5063	6	10024	982	63.9%
2214	5063	7	9038	904	58.9%
2214	5063	8	8263	763	49.7%
2214	5063	9	7104	641	41.7%
2214	5063	10	6552	502	32.7%

Since 15 computing nodes have been used in this computing task and only 10 nodes are available on the entire scheduling platform, the platform system determines that a "memory resource shortage" event has occurred. It is necessary to enhance the importance of memory usage rate and update the judgment matrix as follows:

$$B_1 = \begin{bmatrix} 1 & 1/2 \\ 2 & 1 \end{bmatrix} \tag{19}$$

Resolving the feature vector yields $W_1 = [0.33,0.67]$.

Set the smoothing coefficient $\alpha = 0.6$, then the updated weight is: $W_t = 0.6 \cdot [0.33,0.67] + 0.4 \cdot [0.6,0.4] = [0.44,0.56]$.

Substitute the updated weight W_t into the weighted decision matrix R of ELECTRE, update the harmony matrix C, disharmony matrix D and comprehensive dominance matrix E, thereby obtaining the net dominance index ξ'_k.

All the schemes were ranked according to ξ'_k, and the final ranking results are shown in Table 15 as follows:

Table 15. The decision result of multi-physics coupling simulation computing resource scheduling focusing on the utilization rate of computing resources

Compute node usage	Calculation time (s)	Physical memory usage rate (%)	ξ'_k	Rank
5	11050	67.2%	0.95	3
6	10024	63.9%	1.40	1
7	9038	58.9%	1.17	2
8	8263	49.7%	-1.10	5
9	7104	41.7%	-0.43	4
10	6552	32.7%	-1.99	6

It can be seen from the results in the table that ξ'_k is the highest when the node usage is 6, indicating that the optimal strategy can be obtained when the number of nodes selected in this scheme is 6.

6.3. Analysis of Practical application effects

In order to verify the effectiveness of the computing resource scheduling strategy, the multi-physics coupling computing tasks with mesh numbers greater than 15 million on the high-performance platform within three months before and after the application of the resource scheduling strategy were counted and the computing time and resource usage of each task were recorded. The statistical data are shown in Table 16.

Table 16. The practical application situation of computing resource scheduling decision-making

Date	Number of computing tasks	Average number of mesh ($\times 10^4$)	Average number of freedom ($\times 10^4$)	Average calculation time (s)	Average physical memory usage rate (%)
2024.7-2024.9	172	2227	5189	6475	33.8%
2024.10-2024.12	113	2491	6538	5341	54.2%

The table shows the number of computing tasks in the first three months (2024.7.1-2024.9.30) and the last three months (2024.10.1-2024.12.31) of using the computing resource scheduling strategy, as well as the computing time and resource utilization in each time period. 200 computing tasks were randomly selected from each of the two time periods, and their computing times and physical memory usage rates are plotted as shown in Figures 12 and 13.

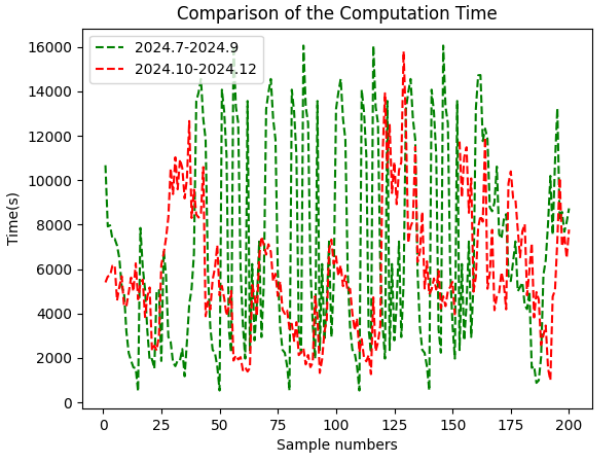


Figure 12. Comparison of the computation time

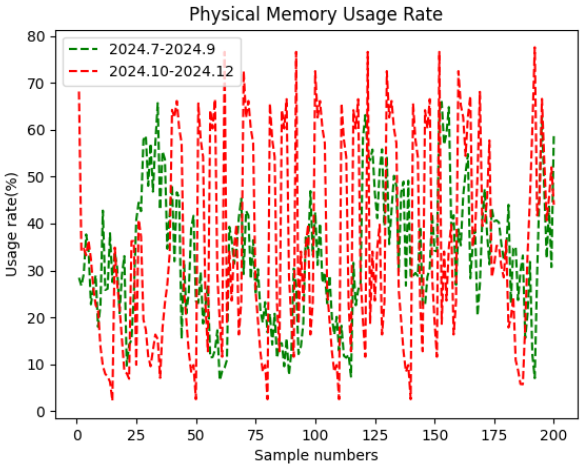


Figure 13. Comparison of physical memory usage

It can be seen from the data in the figure that after using the computing resource scheduling strategy, the average computing time of the high-performance computing platform has decreased by 17.5%, and the physical memory usage rate has increased by 20.4%, reflecting that the resources of large-memory computing nodes have been utilized more fully.

7. Conclusions

This paper aims to reduce computing waiting and improve the utilization rate of scarce large memory computing resources in high-performance clusters. For the first time, it combines the AHP dynamic weight correction mechanism with the ELECTRE method, solving the problem of rigid weights in time-varying task scenarios of traditional methods. For small sample data, a cascading model of "random forest prediction + improved ELECTRE scheduling" was proposed, which reduces the computational overhead while ensuring accuracy and realizes the optimal allocation strategy of comprehensive performance of computing time and physical memory usage rate. Through practical verification, the method proposed in this paper is applicable to the multi-physics coupling simulation software based on unified modeling, achieving the reduction of the average computing time on the high-performance computing platform and the improvement of the utilization rate of large-memory computing nodes. It can provide decision-making references for personnel related to multi-physics coupling simulation when performing computing tasks.

Refrences

[1] Liu H, Meng F, Yan Z, et al. Dynamic simulation of photothermal environment in solar greenhouse based on COMSOL multiple physical fields[J].Agriculture, 2025, 15(2):187-187

[2] Zhang Liting, Li Pengfei, Pang Wenjing, et al. Precipitation prediction based on seasonal auto regressive integral moving average and deep learning long and short term memory neural network [J]. Science Technology and Engineering, 2002,22(09):3453-3463

[3] Sun Zuoyu. Response prediction of semi-active control structures based on dynamic recurrent neural networks [J]. Journal of Vibration Engineering, 2000, 15(03) :123-128

[4] Bai Xiaoxin, Guo Xiangyang, Wu Chunling, et al. Research on Ammonia emission prediction Method for Diesel vehicles based on CNN-Transformer fusion framework [J]. China Environmental Science,2020, 19(6):1-10

[5] Tian Yuhui. Research on construction cost prediction method of large construction projects based on BP neural network [J]. China Construction Metal Structure, 2024, 23 (10) :16-18

[6] DEB K , SUNDAR J. Reference point based multi-objective optimization using evolutionary algorithms [C]//Proceedings of the 8th Annual Conference on Geneticand Evolutionary Computation. New York: ACM, 2006:635-642

[7] MIETTINEN K.Nonlinear multiobjective optimization[J]. International Series in Operations Research and Management Science,1998,12(1):115-129

[8] Sun R. Research on allocation optimization of green credit projects under multi-objective genetic algorithm [D]. Beijing: Beijing jiaotong university, 2023

[9] Liu Y. Research on hydrogen fuel cell performance based on NSGA-Ⅱ multi-objective optimization algorithm [J]. Energy and Energy Conservation, 2024, 23 (8):14-17

[10] Cheng Zhiyu, Zhu Xiaohu, Li Jianqing. Multi-criteria Fusion decision-making method for power network planning investment based on improved ELECTRE method[J]. Power of China, 2022, 55(11):59-65

[11] Loh W. Classification and regression trees [J]. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2011, 1(1): 14-23

[12] Si J, Wang Y, Guan Y, et al. Load forecasting based on multi-core learning Support Vector Machine (SVM)[J].Journal of Physics, 2024, 2876(1): 012035-012035.

[13] Breiman, Leo. Random Forests[J]. Machine Learning, 2001, 45(1):5-32

[14] Cortes, Corinna, and Vladimir V.. Support-Vector Networks[J]. Machine Learning, 1995, 20(3):73-297

[15] Jerome H., Friedman. Greedy function approximation：a gradient boosting machine[J].The Annals of Statistics, 2001, 29(5):1189-1232

[16] JAIN A., DUBES R. Algorithms for Clustering Data[M]. New Jersey: Prentice Hall, 1988: 89-105.

[17] SMITH R., JOHNSON L., WILLIAMS T., et al. Principal Component Analysis in Modern Statistics[J]. Journal of Data Science, 2023, 10(2): 123-135

[18] Darko, Amos. Review of application of analytic hierarchy process (AHP) in construction[J]. International Journal of Construction Management, 2019, 19(5):436-452

[19] Roy, Bernard. The outranking approach and the foundations of ELECTRE methods[J]. Theory and Decision, 1991, 31(1): 49-73

[20] Amestoy P, Duff I, Excellent J. Multifrontal parallel distributed symmetric and unsymmetric solvers [J]. Computer Methods in Applied Mechanics and Engineering, 2000, 184(2): 501-520

[21] Kramer, Oliver, and Oliver Kramer. Scikit-learn[J]. Machine Learning for Evolution Strategies, 2016, 1(1):45-53