ingegneria sismica
International Journal of Earthquake Engineering
Anno XLIII - Num. 2 - 2026

Research on adaptive resource scheduling algorithm and decision model for heterogeneous computing environment

Heping Xu[1] *and Liu Liu*[2,*]

[1] Computer Science Department of Army Academy of Artillery and Air Defense, Hefei 230051, Anhui, China
[2] School of Foreign Studies, Anhui Normal University, Wuhu241003, Anhui, China

SUMMARY: *In order to solve the problems of large differences in task types, strong fluctuations in resource states, and difficulty in balancing delay and utilization in static scheduling in heterogeneous computing environments, this paper proposes an adaptive resource scheduling algorithm and a dynamic resource decision-making model. Based on the unified representation of task-resource, this method integrates heterogeneous perception initialization, nonlinear search control and feedback-driven update into the same scheduling link, so that task mapping, resource reconfiguration and parameter correction can be continuously adjusted with the change of system state. Based on the extended CloudSim platform, comparative experiments were carried out on HPC2N, NASA and synthetic burst workloads. The results show that under the condition of 2000 tasks, the makespan of the proposed method is reduced to 1214 s, which is further decreased by 8.4% compared with the optimal comparison method. The resource utilization rate is increased to 85.6%, and the average response delay is controlled at 109 ms. Research shows that the model can effectively alleviate the node imbalance and task congestion problems in heterogeneous resource pools, and has good adaptability and practical value for online resource scheduling in complex computing scenarios.*

KEYWORDS: *heterogeneous computing environment; Resource scheduling; Dynamic decision model; Adaptive optimization*

1 Introduction

Heterogeneous computing environment has become an important running form of current high performance computing, cloud-edge collaboration and intelligent service platforms. With the simultaneous deployment of cpus, Gpus, FPgas and various specialized accelerators in the same resource pool, the execution ability of computing systems has been significantly expanded, but the mismatch problem between task types, resource characteristics and running constraints has also intensified. For large-scale concurrent tasks, the scheduling process is no longer just a simple "task assignment", but a comprehensive decision-making process involving computing capacity identification, resource status awareness, load balancing control and quality of service guarantee. Whether the scheduling strategy is reasonable directly affects the task completion time, system throughput, node energy consumption level and overall resource utilization [1-4]. Especially in the scenarios with frequent load fluctuations and prominent resource heterogeneity, the lack of targeted scheduling mechanism often leads

*17309666807@hust.edu.cn
https://doi.org/10.65102/is2026726
Published: 30 April, 2026

to the coexistence of high-performance nodes idling and low-fit tasks congestion, which weakens the actual computing benefits of heterogeneous platforms.

In recent years, around the problem of resource scheduling and load balancing, researchers have proposed many kinds of methods such as deep reinforcement learning, swarm intelligence optimization, graph modeling scheduling and multi-objective search [5-10]. These researches have made some progress in shortening makespan, improving response delay and improving local resource utilization, but they still face several prominent difficulties in complex heterogeneous environments: One problem is that it is difficult to fully describe the diversity of resources. Traditional static mapping methods often only focus on task length or node load, which is difficult to reflect the real differences between different processor architectures, bandwidth conditions and energy costs. The other problem lies in the lack of adaptability of the scheduling process to dynamic disturbances. When the task arrival rate, node availability or priority change, the existing models are prone to premature convergence, delayed decision or local optimal solidifications. Moreover, although most methods can optimize a single index, it is difficult to maintain a stable trade-off among execution delay, load balancing and energy overhead [11-14]. This indicates that the research of resource scheduling in heterogeneous computing environment still needs a computing framework with the ability of environment awareness, online adjustment and multi-objective coordination.

Based on this, this paper focuses on the main line of "task-resource-state-decision", and constructs an adaptive resource scheduling algorithm and dynamic decision-making model for heterogeneous computing environment. This method introduced resource heterogeneous representation, task priority dynamic update and feedback-driven decision adjustment process on top of the basic scheduling mechanism, so that the scheduler could complete adaptive mapping and iterative optimization according to the changes of real-time load, node performance and task requirements. The core content of this work is reflected in three aspects. First, a unified representation mechanism of heterogeneous resource capabilities and task computing requirements is established to improve the matching accuracy of task allocation. Secondly, an improved adaptive scheduling strategy for dynamic scenarios is designed to enhance the convergence stability and global search ability of the algorithm under complex load conditions. Thirdly, a dynamic resource decision model is constructed to integrate the execution delay, resource utilization and energy consumption constraints into the same optimization framework, so as to improve the comprehensive performance of scheduling results.

Focusing on the above research objectives, this paper further focuses on the following questions: can the resource representation and state awareness mechanism improve the effectiveness of task mapping under the condition of significant differences between heterogeneous nodes? In the scenario of dynamic task flow continuously arriving, whether the adaptive scheduling strategy can balance the search efficiency and scheduling quality? In the actual system with multi-objective constraints, whether the dynamic decision model can achieve a more stable coordination optimization among delay, load and energy consumption. The analysis and answer to these questions will provide more practical algorithm basis and modeling support for resource scheduling research in heterogeneous computing environment.

2 Related Research

Focusing on the problem of task scheduling and resource decision-making in heterogeneous computing environments, existing research has gradually expanded from traditional heuristic search to deep reinforcement learning, swarm intelligence optimization, and graph structure

modeling. Xue et al. proposed a hybrid scheduling algorithm based on deep reinforcement learning, which modeled the task assignment process as a continuous decision problem, and gave consideration to both execution efficiency and resource utilization in the edge computing scenario, reflecting a good online adjustment ability [1]. Zheng et al. further applied reinforcement learning to workload scheduling, improved node load distribution through state-action mapping, and obtained lower response delay under the condition of dynamic task flow [2]. The advantage of these methods is that they have the ability to interact with the environment. However, when the dimension of the state space is high, the training stability and convergence cost are still relatively prominent.

In terms of adaptive optimization and multi-objective scheduling, Nabi et al. proposed AdPSO algorithm to improve task scheduling performance in cloud environment through adaptive adjustment of particle swarm parameters, which has certain improvement effects on task completion time and resource balance [3]. Kruekaew and Kimpan combined artificial bee colony algorithm with reinforcement learning for multi-objective task scheduling in complex cloud environment, which enhanced load balancing and global search ability [4]. Mahmoud et al. introduced the decision tree mechanism to deal with the multi-objective scheduling problem, so that the task allocation process had a clearer discrimination path, and established a stable compromise relationship between resource allocation and task completion time [5]. Pradhan et al. combined deep reinforcement learning with parallel particle swarm optimization to make load balancing decisions in cloud environment, which improved the responsiveness of scheduling process to dynamic changes [6]. This path shows that it is difficult to adapt to complex disturbances in heterogeneous resource pools by relying solely on fixed strategies, and the adaptive adjustment mechanism of algorithms is becoming a research focus.

As the scheduling scenarios extend from traditional cloud platforms to resource-constrained, multi-tenant and edge-cloud collaborative systems, the research objects also emphasize more on heterogeneity and real-time performance. Aiming at the resource-constrained multi-tenant serverless environment, Mampage et al. use deep reinforcement learning to optimize application scheduling and improve task execution sequence and node utilization effect under the condition of resource competition [7]. Song et al. proposed a job scheduling algorithm based on reinforcement learning for heterogeneous computing environments, which enables tasks to be mapped more reasonably according to device characteristics and gives a more direct response to the execution differences under different processor architectures [8]. From the perspective of priority perception, Sharif et al. realized the collaborative optimization of task scheduling and resource allocation in the edge computing health monitoring system, so that high-priority tasks could obtain more stable execution guarantee [9]. Saif et al. adopted the multi-objective grey Wolf optimization algorithm to deal with the task scheduling problem in the cloud-fog collaborative environment, and incorporated delay, energy consumption and resource allocation into the unified optimization process [10]. These studies have made positive progress in multi-objective constraints and dynamic workload processing, but most of them still focus on single scenario verification, and the joint modeling among heterogeneous node ability differences, task type changes and online decision feedback is still insufficient.

Synthesizing the existing results, it can be found that although the current research has formed a rich accumulation of methods in task scheduling efficiency, load balancing and energy consumption control, there are still obvious differences in heterogeneous resource representation accuracy, dynamic disturbance adaptability and multi-objective collaborative optimization between different technical routes. Table 1 summarizes and compares the method types, core mechanisms, optimization objectives and limitations of related studies. It

can be seen from Table 1 that most of the existing methods can achieve improvement in local indicators, but the attention to the ability difference of heterogeneous nodes, online feedback update and integrated scheduling-decision modeling is still insufficient. In order to improve the task mapping accuracy and overall scheduling performance in complex computing environments, this paper further studies from three levels: resource heterogeneous characteristics characterization, adaptive scheduling strategy design and dynamic resource decision model construction.

Table 1: Comparative analysis of related studies

Reference	Method Type	Core Mechanism	Main Optimization Objectives	Remaining Problems
[1]	Deep Reinforcement Learning-Based Hybrid Scheduling	Models scheduling as a continuous decision-making process	Execution efficiency, resource utilization	High training cost in high-dimensional state spaces
[2]	Reinforcement Learning-Based Workload Scheduling	State-action mapping and load adjustment	Response latency, load distribution	Generalization ability is constrained by specific scenarios
[3]	Adaptive PSO	Dynamic adjustment of particle parameters	Makespan, resource balancing	Easily affected by initialization quality
[4]	Swarm Algorithm + Reinforcement Learning	Multi-objective search and policy updating	Load balancing, global optimization	Limited convergence speed in complex scenarios
[5]	Decision Tree Scheduling	Discriminative resource allocation path	Scheduling time, resource allocation	Insufficient dynamic feedback capability
[6]	DRL + Parallel PSO	Collaborative optimization of load balancing and scheduling strategy	Load control, scheduling efficiency	Relatively complex model structure
[7]	Multi-Tenant Serverless Scheduling	Policy learning under constrained resources	Node utilization, task ordering	Insufficient characterization of heterogeneous device differences
[8]	Reinforcement Learning Scheduling in Heterogeneous Environments	Job mapping for heterogeneous devices	Job execution efficiency	Multi-objective coordination still needs improvement
[9]	Priority-Based Resource Allocation	Task-priority-driven scheduling	Critical task assurance, resource allocation	Generality is biased toward specific applications
[10]	Multi-Objective Grey Wolf Optimization	Joint optimization in cloud-fog collaboration	Latency, energy consumption, load	Limited online adaptive capability

3 Modeling task scheduling and resource decision in heterogeneous computing environment

Resource scheduling in heterogeneous computing environment is not at the same level as task assignment in traditional homogeneous cluster. In the resource pool composed of CPU, GPU, FPGA and multi-class acceleration nodes, different tasks have different requirements on computing density, memory access bandwidth, parallel granularity and response time. If the fixed priority or static rotation method is still used to allocate the tasks, it is easy to cause high performance node waiting, low adaptation node congestion and local load accumulation. Based on this, this paper defines the scheduling problem as a joint optimization process for heterogeneous resource pools, which completes the task-resource mapping and dynamic decision update according to task characteristics, resource status and operation feedback under the condition of given task set and node set, so that the system achieves a more stable balance among execution delay, resource utilization and operation energy consumption.

Let the set of heterogeneous tasks be $\mathcal{T} = \{\tau_1, \tau_2, \cdots, \tau_n\}$ and the set of resource nodes be $\mathcal{R} = \{r_1, r_2, \cdots, r_m\}$. Task τ_i is described by computation length l_i, priority p_i, deadline d_i and resource requirement vector q_i. Node r_j is characterized by computing power, parallel acceleration ability, storage bandwidth and current load, which is denoted as resource capacity vector h_j. In order to avoid the bias caused by measuring the node ability only by a single main frequency or the number of cores, this paper introduces the resource adaptation coefficient θ_{ij} to describe the matching degree between task τ_i and node r_j. Thus, the expected execution time of task τ_i on node r_j can be expressed as follows.

$$P_{ij} = \frac{l_i}{\theta_{ij}(\mu_1 c_j + \mu_2 g_j + \mu_3 b_j)} \tag{1}$$

where c_j, g_j and b_j represent the general computing ability, accelerated computing ability and bandwidth support ability of node rj respectively, μ_1, μ_2, μ_3 are the corresponding weights, and $\mu_1 + \mu_2 + \mu_3 = 1$. Equation (1) essentially gives the task-resource execution time matrix in a heterogeneous environment, which is not only used to estimate the running time, but more importantly to provide a comparable unified scale for subsequent scheduling decisions. The related modeling process is shown in Figure 1.

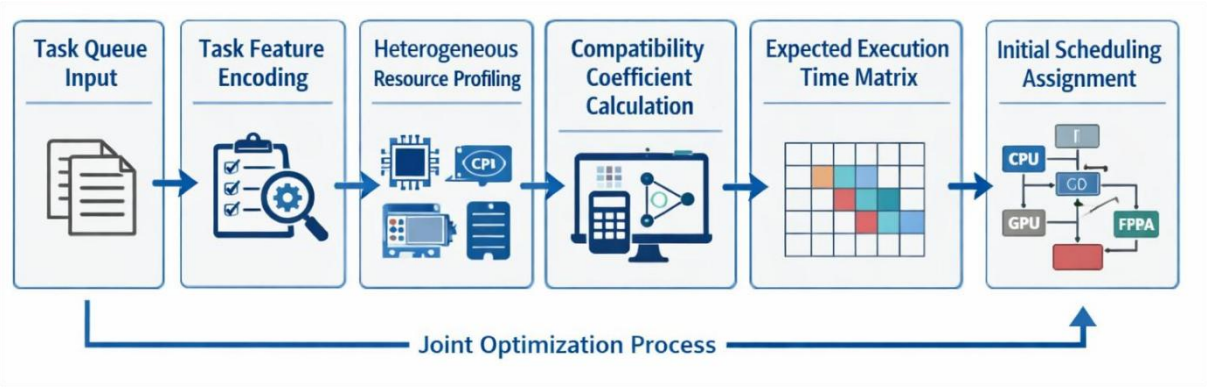

Figure 1: Modeling framework for task-resource mapping in heterogeneous computing environment

After obtaining the expected execution time matrix, the binary allocation variable x_{ij} is

further defined. When task τ_i is assigned to node r_j, $x_{ij} = 1$; Otherwise $x_{ij} = 0$. Considering that heterogeneous nodes already have queuing load at scheduling time t, the cumulative completion time of node r_j is written as follows.

$$C_j(t) = q_j(t) + \sum_{i=1}^{n} x_{ij} P_{ij} \tag{2}$$

Here, $q_j(t)$ represents the waiting queue overhead of node r_j at the current time. It can be seen from Equation (2) that the execution effect of the same task on different nodes is not only affected by the inherent ability of nodes, but also directly related to its instantaneous load state. In other words, heterogeneous scheduling is not a simple static matching, but a joint selection of resource capacity and operation state.

In the evaluation of scheduling performance, we keep the most representative two core indicators in heterogeneous computing environment, namely system total completion time and resource utilization efficiency. The total completion time is measured by the maximum node completion time and is defined as follows.

$$(3)M = \max_{1 \le j \le m} C_j(t) \tag{3}$$

This index reflects the final convergence time of task batch in the whole resource pool, and can directly describe the impact of scheduling strategy on global execution efficiency. At the same time, resource utilization should not be simply equated with the ratio of nodes powered on, but should describe the proportion of effective execution time in the whole scheduling cycle. Therefore, the system resource utilization is expressed as follows in this paper.

$$U = \frac{\sum_{j=1}^{m} \sum_{i=1}^{n} x_{ij} P_{ij}}{m \cdot M} \tag{4}$$

The larger U in Equation (4), the less idle waiting time in the same scheduling period, the more fully utilized the resource pool as a whole. If we only pursue the minimum M, the tasks may be excessively concentrated on a few high-performance nodes. If we only pursue U maximum, it may cause low-priority tasks to crowd out critical resources. To this end, this paper constructs the scheduling objective as a comprehensive optimization function considering delay, utilization and load balancing:

$$\min J = \omega_1 \widetilde{M} + \omega_2 \widetilde{B} - \omega_3 \widetilde{U} + \omega_4 \widetilde{E} \tag{5}$$

where $\widetilde{M}, \widetilde{B}, \widetilde{U}$ and $\widetilde{E}$ represent the normalized total completion time, load deviation, resource utilization and energy consumption overhead, respectively, $\omega_1, \omega_2, \omega_3, \omega_4$ are the target weights, and $\sum \omega_k = 1$ is satisfied. Compared with single metric optimization, this modeling approach is more consistent with the actual operation characteristics of heterogeneous platforms, because in real systems, high throughput, low latency and low energy consumption are not naturally established at the same time, and the scheduler must make a dynamic tradeoff between the conflicting goals.

A static objective function alone is not enough to describe the running process of continuously arriving tasks in heterogeneous environments. The reason is that the node availability, queue size, task priority, and resource contention are constantly changing, and the optimal mapping at one time does not guarantee that it will be valid at the next time. Based on

this understanding, this paper further models the resource decision process as a state transition system in discrete time. Let the system state at time t be s_t, which is composed of node load, task waiting length, resource adaptation distribution, energy consumption level and priority risk. The decision action is a_t, which represents the operation of task allocation, resource reconfiguration or execution order adjustment. Then the system evolution can be abstracted as follows.

$$s_{t+1} = f(s_t, a_t, \xi_t) \tag{6}$$

Here, ξ_t represents the task disturbance and environmental change information arriving from outside at time t. The significance of Equation (6) is that the heterogeneous resource scheduling is extended from "single allocation" to "continuous decision", so that the scheduler can revise the subsequent policy according to the feedback results, rather than remaining unchanged after the initial mapping. The overall structure of the dynamic resource decision model is shown in Figure 2.

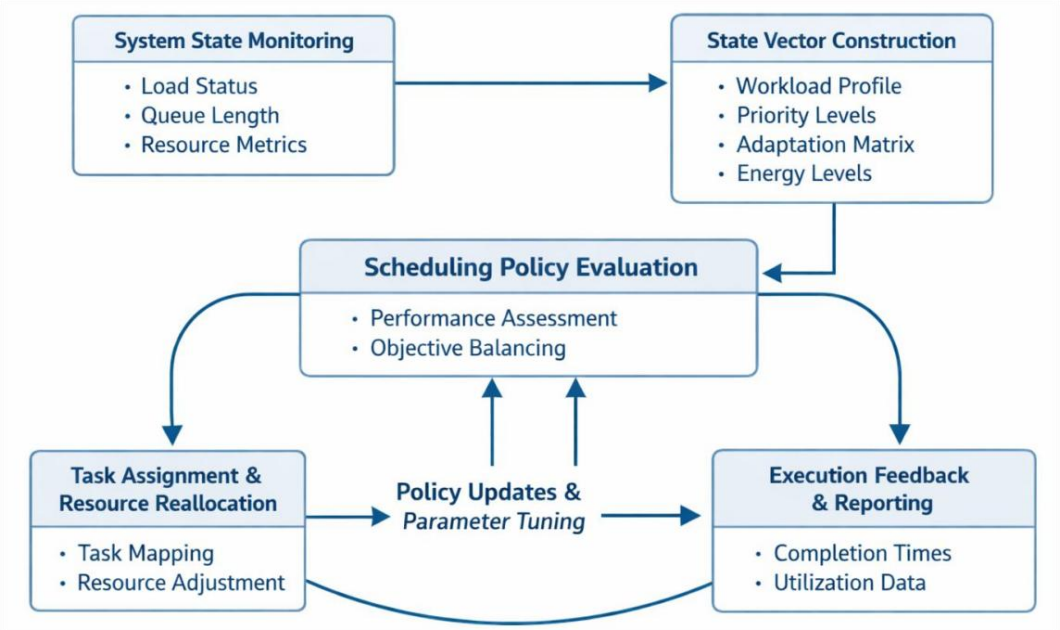

Figure 2: Dynamic resource decision model and feedback optimization closed loop

In the above model, resource decision is not only to determine which kind of node a task should be placed on, but also to adjust resources and modify policies during execution. For example, when the load of a certain kind of GPU node increases sharply and the waiting time of critical tasks increases rapidly, the decision module can judge whether to migrate some low-priority tasks to general-purpose computing nodes or temporarily increase the occupancy ratio of high-priority tasks on acceleration resources according to the state vector. In this way, the task scheduling model and the resource decision model form a unified computing link: the former is responsible for describing the task-resource mapping relationship and multi-objective optimization boundary, and the latter is responsible for continuously revising the decision direction under dynamic disturbance conditions.

4 Adaptive resource scheduling algorithm and dynamic decision model

4.1 Basic scheduling Algorithm and resource allocation mechanism

In heterogeneous computing environments, the basic role of scheduling algorithms is not only to "distribute" tasks, but to establish an executable, comparable, and iterable resource allocation process according to the corresponding relationship between task characteristics and node capabilities. If the initial scheduling phase only completes the mapping according to the arrival order or a single load index, although the implementation is simple, it is easy to ignore the differences between CPU, GPU and other acceleration nodes in parallel granularity, memory access efficiency and execution cost, which will lead to the increase of task queuing time and the imbalance of resource occupancy. Based on the above modeling, this paper constructs a basic scheduling algorithm for heterogeneous resource pools, which integrates task coding, resource representation, fitness evaluation and allocation update into the same calculation process. The overall structure is shown in Figure 3.

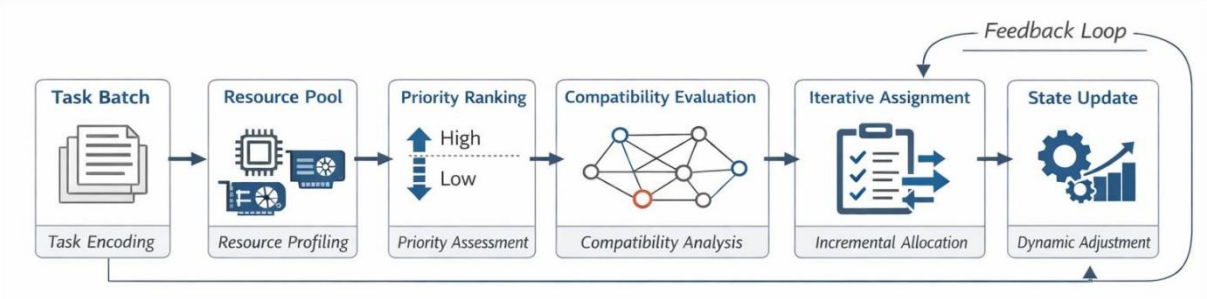

Figure 3: Framework of basic scheduling algorithm and resource allocation mechanism

The underlying scheduling algorithm starts with job batches $\mathcal{T} = \{\tau_1, \tau_2, \dots, \tau_n\}$ is the input, and each task is represented as a feature vector composed of computation length, data transmission scale, deadline and service level. Correspondingly, each node r_j in the resource pool is described by its processing capacity, memory bandwidth, current load, available queue length and unit energy consumption level. In order to match tasks and resources in a unified scale, this paper encodes a single candidate scheduling scheme into a one-dimensional mapping vector:

$$s = [s_1, s_2, \dots, s_n] \tag{7}$$

Here, $s_i = j$ means that task τ_i is assigned to node r_j. This encoding method avoids the redundant overhead caused by high-dimensional Boolean matrix in the early stage of search, and facilitates the subsequent rapid update and local correction in the scheduling space.

Before resource allocation starts, the system needs to sort the task execution urgency. Considering that not all tasks in a heterogeneous environment should be prioritized according to a single deadline or task length, this paper defines a task priority function:

$$\pi_i = \alpha \frac{l_i}{\max(l)} + \beta \frac{1}{d_i + \varepsilon} + \gamma p_i \tag{8}$$

where l_i is the task computation length, d_i is the remaining deadline, p_i is the service class weight, α,β,γ are the normalization coefficients, and $\alpha+\beta+\gamma=1$. The function of Equation (8) is

to include "computational burden", "deadline pressure" and "business importance" into the ranking basis at the same time, so that the basic scheduling stage has certain task differentiation ability, rather than treating all jobs as homogeneous objects.

After finishing the priority sorting, the scheduler further calculates the base fitness between task τ_i and resource node r_j. Because the computing advantages of heterogeneous nodes are not consistent, selecting nodes only based on main frequency or idle rate often leads to the misallocation of high bandwidth tasks to the devices with low memory capacity. To this end, this paper gives the resource adaptation score:

$$A_{ij} = \eta_1 \frac{\theta_{ij}}{\max(\theta)} + \eta_2 \frac{1 - q_j}{\max(q) + \varepsilon} + \eta_3 \frac{1 - e_j}{\max(e) + \varepsilon} \tag{9}$$

Here, θ_{ij} represents the structural matching coefficient between the task and the node, q_j is the current load level of the node, e_j is the energy consumption intensity per unit time, η_1, η_2, η_3 are the weight parameters. This formula shows that the basic resource allocation does not only focus on "who is fast", but also considers the current available state of the node and the execution cost. A higher adaptation score indicates that the node is more suitable as a candidate execution location for task τ_i.

Based on task priority and resource adaptation score, the initial scheduling mechanism of "ranking-driven-item insertion" is adopted in this paper. Specifically, the system first arranges the tasks according to π_i from high to low, and then selects the resource node that minimizes the overall cost for each task in turn. If the expected completion time after node r_j receives task τ_i is C_{ij}, the local selection criterion for the underlying schedule can be written as follows.

$$j^* = \arg\min_j \left(\lambda_1 C_{ij} - \lambda_2 A_{ij}\right) \tag{10}$$

Here, λ_1 and λ_2 are used to balance the completion time and the adaptation benefit. Equation (10) means that the task allocation is not only to find the shortest execution time node, but to establish a compromise between "complete as soon as possible" and "reasonable match". For the same type of highly parallel tasks, GPU nodes may be preferentially selected due to their high adaptation coefficient. For lightweight control tasks with tight deadlines, they may be scheduled to general-purpose nodes with more stable response.

After the resource allocation is completed, the system updates the waiting queue, available load and energy consumption estimation of nodes, and forms a new resource state vector. If a node is close to the congestion threshold due to continuous receiving of high-priority tasks, the basic scheduling mechanism will automatically reduce its adaptation score in the next round of task insertion, thereby suppressing the centralized occupation of resources. Although this update method still belongs to the base layer algorithm, it has the preliminary closed-loop characteristics: task input changes resource state, resource state in turn affects the subsequent task mapping, and the scheduling result is no longer a one-time static output, but a gradual evolution of the allocation process.

4.2 Improved adaptive scheduling strategy for heterogeneous environment

In this paper, we introduce heterogeneous sensing initialization, nonlinear adaptive control and feedback disturbance correction into the basic scheduling framework to improve the environmental adaptability and dynamic search stability of the scheduling strategy. The first

step in the improvement strategy is to reconstruct how the initial candidate assignments are generated. Although the basic algorithm uses task priority and adaptation score to generate the initial mapping, the difference between different candidate solutions is still limited, which is not conducive to the expansion of the subsequent scheduling search. In order to enhance the coverage of the initial solution, the task-resource matching relationship and node congestion risk are jointly encoded as heterogeneous guiding factors:

$$H_{ij} = \rho_1 \hat{\theta}_{ij} + \rho_2 (1 - \hat{q}_j) + \rho_3 (1 - \hat{e}_j) + \rho_4 \hat{m}_j \tag{11}$$

where $\hat{\theta}_{ij}$ is the normalized task-resource fitness, $\hat{q}_j$ is the node load level, $\hat{e}_j$ is the unit energy consumption intensity, $\hat{m}_j$ represents the available memory or bandwidth of the node, and $\rho_1 + \rho_2 + \rho_3 + \rho_4 = 1$. The function of Equation (11) is to compress the multi-dimensional ability differences of heterogeneous nodes into a unified scoring space, so that the initial candidate solutions no longer only focus on the "current fastest node", but take into account both resource availability and execution cost. Based on this bootstrap factor, the probability that task τ_i is assigned to node r_j in the initialization phase can be expressed as:

$$P_{ij}^{(0)} = \frac{\exp(H_{ij}/\tau)}{\sum_{k=1}^{m} \exp(H_{ik}/\tau)} \tag{12}$$

Here, τ is the temperature parameter. Compared with deterministic allocation, Equation (12) retains the tendency that better nodes are preferentially selected, and also retains the opportunity to enter the candidate solution set for suboptimal but potentially effective resource mapping, thereby improving the problem of too narrow distribution of initial solutions. Figure 4 shows the overall process of the improvement strategy.

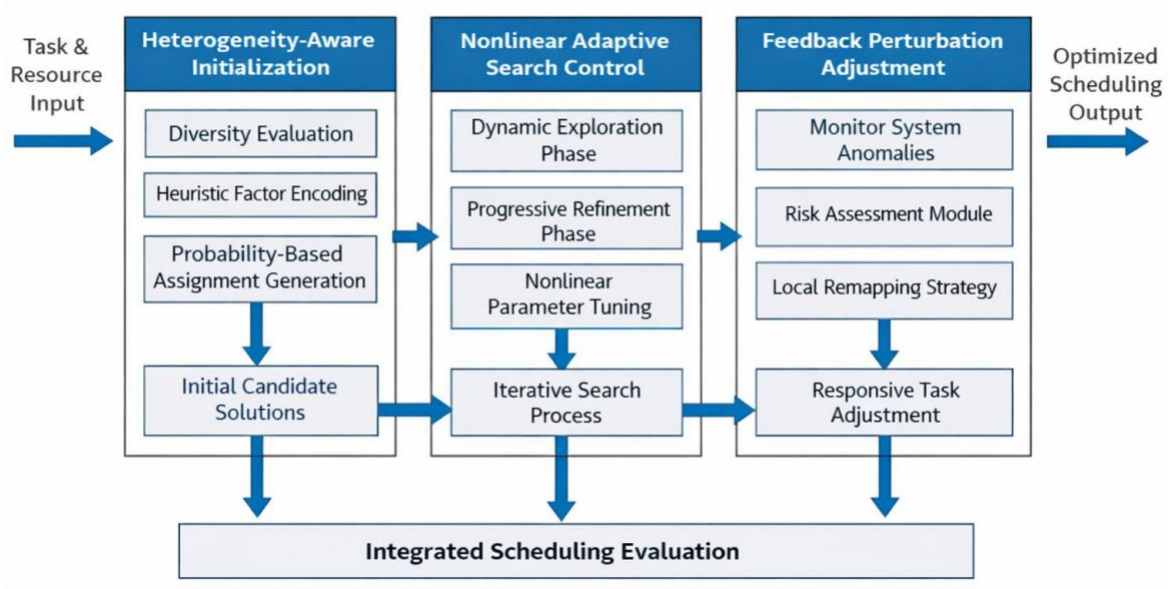

Figure 4: Process of improved adaptive scheduling policy for heterogeneous environment

After the candidate solution generation, the nonlinear adaptive control parameter is further used to adjust the advancing rhythm of the search phase. Although the traditional linear decay method is simple to implement, it often leads to the phenomenon of insufficient search in the early stage and too fast contraction in the later stage in heterogeneous scheduling problems. To this end, this paper defines the search control factor:

$$\alpha(t) = \alpha_{min} + (\alpha_{max} - \alpha_{min}) \exp\left(-\kappa \frac{t}{T}\right) \tag{13}$$

Here, t is the current iteration round, T is the maximum iteration number, and κ is the control coefficient. This parameter kept a high level in the early iteration, which made the algorithm more inclined to search wide-area across nodes and types of resources. As the iteration progresses, α(t) gradually decreases, and the search center turns to local refinement, so as to improve the convergence accuracy of high-quality scheduling solutions. Compared with the fixed update rule, this nonlinear control method is more suitable for heterogeneous scenarios, because the task distribution and resource state are not stable, the algorithm needs to retain enough exploration flexibility in the early stage, and strengthen the utilization of high-quality mapping regions in the middle and late stage.

Relying on control factor adjustment alone is still not sufficient to avoid local optima. In order to enhance the responsiveness to burst load and congested nodes, this paper introduces feedback disturbance correction mechanism. Suppose that the load deviation, queuing growth rate and energy consumption fluctuation of node r_j at time t together constitute the feedback risk term $F_j(t)$. When $F_j(t)$ exceeds the threshold, the system does not directly overturn the current scheduling solution, but performs a local remapping of the relevant task subset, whose update criterion is written as:

$$s_i^{t+1} = \arg\max_j \left[\alpha(t) H_{ij} + (1 - \alpha(t)) G_{ij}(t) - \beta F_j(t)\right] \tag{14}$$

Here, $G_{ij}(t)$ represents the real-time revenue estimation of task τ_i on node r_j, considering the execution delay, completion rate and resource idle recovery in recent rounds, and β is the risk penalty coefficient. Equation (14) means that the scheduler simultaneously considers three types of information: static heterogeneous adaptation, dynamic execution benefit and risk penalty when updating. If a node has strong computing power, but its queue growth is too fast or its energy consumption is abnormally high, the attraction of the node in the next round of allocation will be actively weakened. On the contrary, some nodes that are underutilized in the initial stage may re-enter the candidate core area due to low risk.

After the above improvements, the scheduling strategy presents a clearer two-layer feature. In the outer layer, heterogeneous guidance factors and nonlinear control parameters are used to maintain the search range to avoid excessive concentration in the early stage. In the inner layer, the feedback disturbance correction mechanism is used to adjust the local congestion, imbalance and energy consumption anomaly in a directional manner, reducing the probability of local optimal solidification. This method is different from the general heuristic "random perturbation". The modification in this paper is not an unconstrained jump, but is based on the joint drive of task characteristics, resource heterogeneity and operation feedback, so it has stronger interpretability.

4.3 Construction and optimization of dynamic resource decision model

In heterogeneous computing environments, task arrival frequency, node load level, accelerator availability, and energy consumption constraints are all in flux. An effective allocation result at one time may not be advantageous at the next. If the model still executes the same rules repeatedly according to static weights, it is easy to cause local overheating of high-performance nodes, accumulation of migration overhead and fluctuation of critical task response. Based on this realistic constraint, we further construct a dynamic resource decision model for heterogeneous resource pools, and organize the "state perception-action

generation-execution feedback-parameter correction" into a closed-loop optimization link, so that the scheduler has the ability of online adjustment.

In this model, the running state of the system is abstracted as a unified state vector, which is continuously updated by a rolling time window. Let the system state at time t be:

$$s_t = [\bar{q}_t, \bar{u}_t, \bar{b}_t, \bar{e}_t, \bar{r}_t, \bar{p}_t] \tag{15}$$

Here, $\bar{q}_t$ represents the normalized queue length, $\bar{u}_t$ represents resource utilization, $\bar{b}_t$ represents bandwidth occupancy level, $\bar{e}_t$ represents energy consumption per scheduling cycle, $\bar{r}_t$ represents the risk of deadline violation, and $\bar{p}_t$ corresponds to the pressure of high-priority tasks. Different from the above local score based on the characteristics of a single node, Equation (15) emphasizes the compressed expression of the global operating state, whose role is to provide continuous, comparable and easily updated input for the decision-making module. The overall structure of the model is shown in Figure 5.

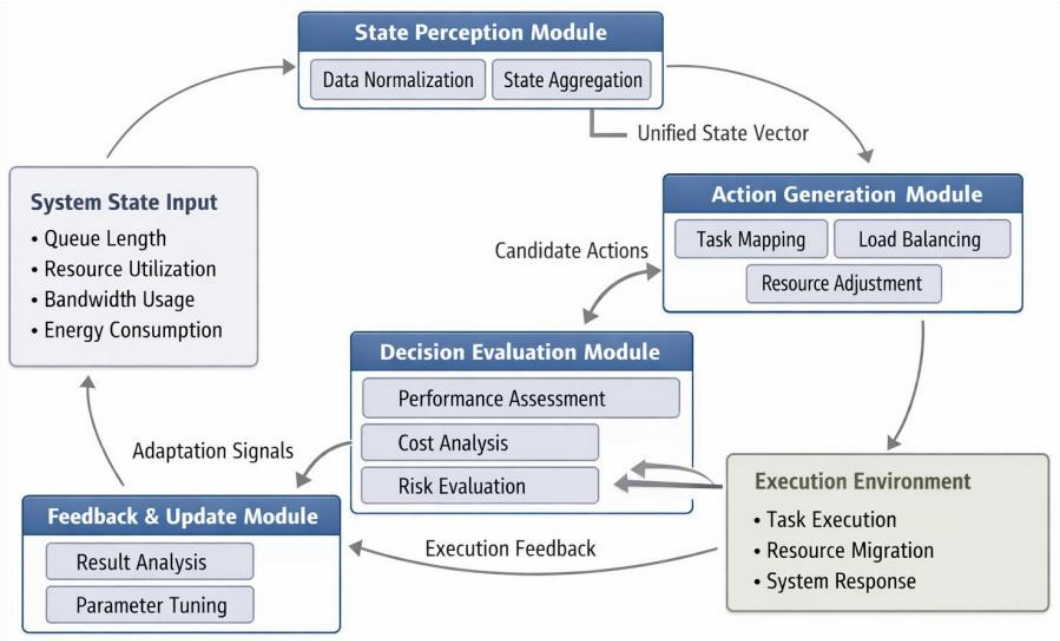

Figure 5: Dynamic resource decision model and feedback optimization framework

In the design of action space, we do not restrict the decision to a single "task placement", but incorporate task remapping, execution order adjustment, node load migration and local resource reconfiguration into the candidate action set. The reason for this is that fluctuations in performance in heterogeneous environments are often triggered not by a single new task but by a combination of queuing buildup, resource contention, and ill-timed migration. In order to compare the comprehensive benefits of different actions in the current state, this paper defines a decision scoring function:

$$F(s_t, a_t) = \lambda_1 \hat{U}(s_t, a_t) - \lambda_2 \hat{W}(s_t, a_t) - \lambda_3 \hat{E}(s_t, a_t) - \lambda_4 \hat{L}(s_t, a_t) \tag{16}$$

where $\hat{U}$ represents the expected resource utilization benefit after action execution, $\hat{W}$ represents the waiting delay cost, $\hat{E}$ represents the extra energy consumption overhead, $\hat{L}$ represents the load imbalance degree, and $\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1$. Equation (16) does not pursue the global optimum in an absolute sense, but unifies the three requirements of "fast", "stable" and "economical" into a comparable scale through multi-objective weighting, and

then selects the one with the highest score from the candidate actions as the current decision output. The components of the model and their roles are shown in Table 2.

Table 2: Main modules and optimization function of dynamic resource decision model

Module	Input Information	Core Processing	Output Function
State Perception Module	Node load, queue length, energy consumption, bandwidth utilization, task priority	Multi-source data normalization and state aggregation	Forms a global state vector
Candidate Action Generation Module	Current state, task queue, available node resources	Generates remapping, resequencing, migration, and local reconfiguration actions	Provides a set of candidate decisions
Decision Scoring Module	State vector and candidate actions	Calculates a comprehensive score based on utilization gain, latency cost, energy consumption, and load balancing	Outputs the optimal action
Execution Feedback Module	Actual completion time, resource occupancy, migration cost, violation rate	Evaluates action effectiveness and deviation	Forms feedback signals
Parameter Update Module	Current scoring results, actual feedback, historical policy weights	Adjusts decision parameters and action preferences	Improves adaptability of subsequent decisions

In order to avoid the model being solidified in a certain type of resource preference after a long time running, this paper further introduces a feedback-driven parameter update mechanism. Let g_t be the actual execution payoff at time t and $\hat{g}_t$ be the payoff prediction of the decision module, then the model parameter update is written as:

$$\Theta_{t+1} = (1 - \eta)\Theta_t + \eta(g_t - \hat{g}_t)\Psi_t \tag{17}$$

Here, Θ_t represents the current set of decision parameters, η is the update step size, and Ψ_t is the feature response vector corresponding to the current action. The function of this formula is not to make a complex derivation, but to use the prediction bias to correct the strength of the model's judgment of the resource state. If a certain type of action is better in the theoretical score, but causes high migration overhead or waiting time rebound in the actual execution, its corresponding parameters will be automatically weakened. On the other hand, if some low-frequency actions perform stably under specific load conditions, the model will gradually increase its probability of being selected. In this way, the decision model is transformed from "fixed rule scheduling" to "learnable online optimization process".

5 Experimental results and analysis

In order to test the effectiveness of the adaptive resource scheduling algorithm and dynamic resource decision model proposed in this paper in heterogeneous computing environment, the experiment is completed on the extended CloudSim Plus platform, and the heterogeneous

modeling of CPU, GPU and FPGA nodes is carried out. The simulation resource pool consisted of 60 computing nodes, including 24 CPU nodes, and the single node was configured as 32 cores@2.6 GHz and 64 GB RAM. There were 20 GPU nodes configured at 16 cores@2.4 GHz, 1 block of T4 level accelerator, and 128 GB RAM. There are 16 FPGA nodes configured as 12 cores@2.3 GHz, reconfigurable logic resources and 64 GB RAM. The task length is distributed from 1.0×10^3 to 1.2×10^4MI, and the data transmission scale, deadline and priority perturbation are introduced to approximate the real heterogeneous workload. Three kinds of task trajectories including HPC2N, NASA and synthetic burst load were selected for the experiment, and the task size was set to 500, 1000, 1500 and 2000, all of which were scheduled independently and non-preemptively. Each group of experiments was repeated 30 times and the average value was taken. The comparison algorithms include AdPSO, GA-GWO, RL-Scheduler and GNN-RL. The evaluation indicators are total completion time, resource utilization, average response delay, scheduling operation overhead and stability performance after ablation.

From the perspective of overall execution efficiency, the proposed method maintains a lower makespan under different task sizes. Figure 6 shows the trend of the total completion time of each algorithm under the condition of multi-task size. As the number of tasks increases from 500 to 2000, the completion time of all methods shows an upward trend, but the growth slope of the proposed method is significantly slower. Taking 2000 tasks as an example, the makespan of the proposed method is 1214 s, which is lower than that of GNN-RL (1326 s), RL-Scheduler (1378 s), GA-GWO (1431 s) and AdPSO (1498 s). Compared with the best performing baseline GNN-RL, the completion time is decreased by 8.4%. Compared with the traditional adaptive particle swarm optimization method, the reduction is 19.0%. This result shows that the heterogeneous sensing initialization and nonlinear control strategy proposed in Section 4.2 can still maintain high search effectiveness after the task scale is expanded, so that the scheduling process is not easy to lose the global coordination ability due to local congestion. Especially in the high-voltage scenario with more than 1500 tasks, the over-concentration phenomenon of high-load GPU nodes is suppressed more obviously by the proposed model, so it retains better convergence stability in the later stage execution.

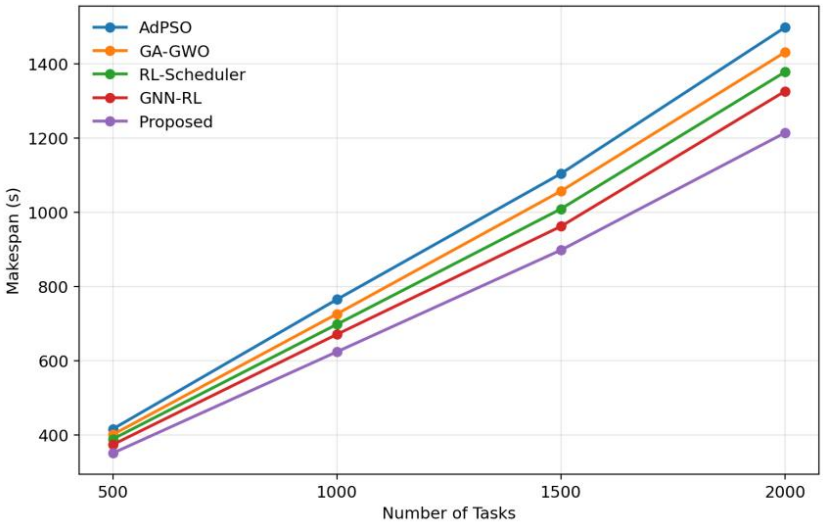

Figure 6: Comparison of makespan of each algorithm under different task scales

The comparison results of resource utilization are shown in Figure 7. It can be seen that the resource utilization of the proposed method under four groups of task sizes reaches 80.4%,

82.7%, 84.1%and 85.6%respectively, which is always higher than that of the other comparison algorithms. Taking 2000 task scenarios as an example, the proposed method outperforms GNN-RL by 4.4 percentage points and RL-Scheduler by 5.8 percentage points. This improvement does not rely solely on the high occupancy of a certain class of high-performance nodes, but under the role of dynamic resource decision model, general computing, accelerated computing and bandwidth resources are kept in a more reasonable coordination range. In other words, the proposed method does not "push" tasks to the fastest node, but continuously modifies the mapping direction according to the task structure and node status, so that different types of resources can maintain more sufficient effective busy time in the scheduling cycle. The utilization improvement reflected in Figure 7 also echoes the reduction in completion time in Figure 6, indicating that the proposed method does not sacrifice local resource balance for superficial time gains, but achieves more stable optimization results at the overall resource organization level.

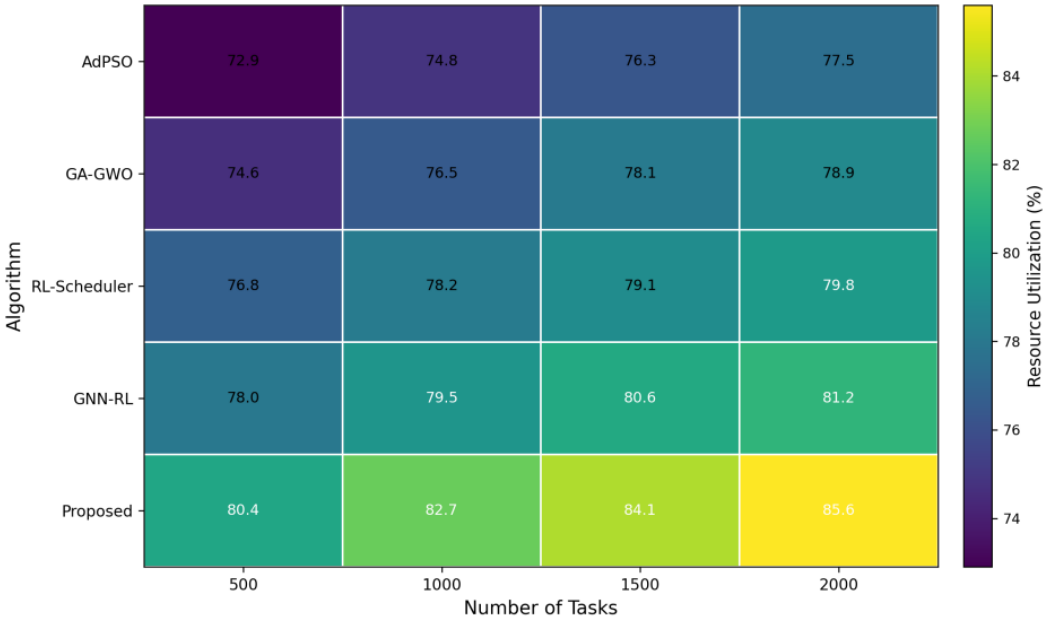

Figure 7: Comparison of resource utilization of each algorithm under different task scales

In order to further investigate the adaptability of the dynamic decision model to time-varying loads, the average response delay under three types of workload traces is counted, and the results are shown in Figure 8. The complexity of the task structure of HPC2N, NASA and synthetic burst workload increases in turn, which puts forward higher requirements for the online reaction ability of the scheduler. The average response delay of the proposed method in the three scenarios is 86 ms, 101 ms and 109 ms, respectively, while GNN-RL is 98 ms, 115 ms and 124 ms, respectively. In other words, the delay reduction of the proposed method on the three types of trajectories is maintained at about 12%, showing a consistent generalization ability. This result shows that the state vector modeling and feedback update mechanism in Section 4.3 indeed plays a role: when the workload changes from relatively smooth to burst mixed, the model can timely sense the queue growth, bandwidth occupancy and priority pressure changes, and adjust the task remapping strategy through the candidate action score, so there is no obvious delay instability. Although the response delay of all algorithms increases significantly under synthetic burst load, the increase of the proposed method is relatively smaller, which also verifies that the dynamic resource decision model has a stronger buffer ability for abnormal load disturbance.

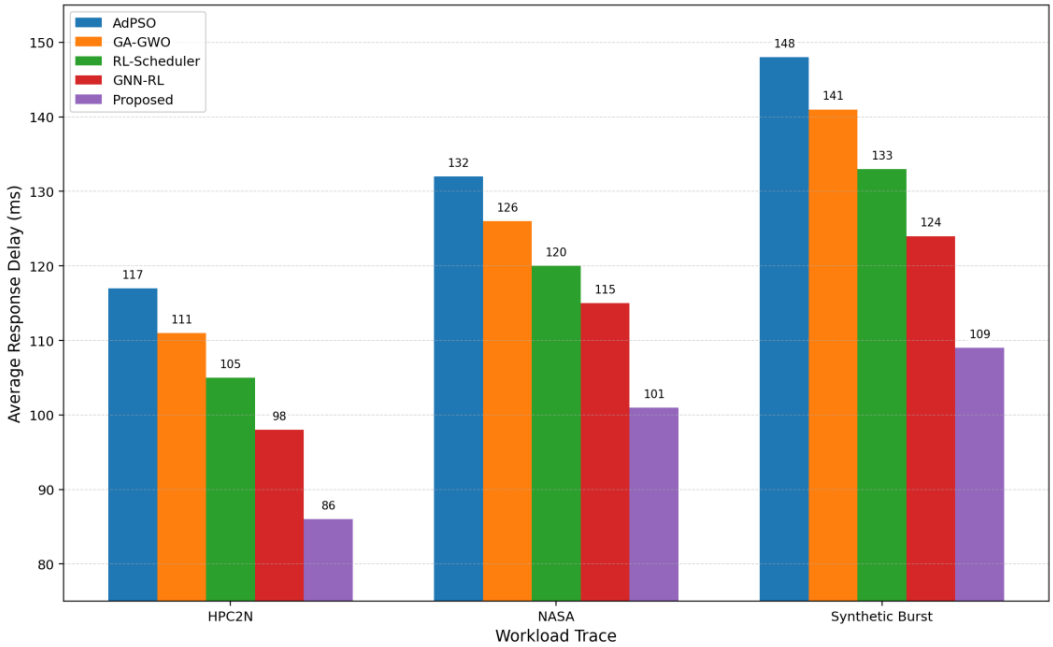

Figure 8: Comparison of average response delay under different workload traces

Besides the scheduling quality, the running cost of the algorithm itself also needs to be evaluated. Table 3 shows the average scheduling running time, approximate computation amount and normalized comprehensive score of each algorithm for 2000 task scenarios. It can be seen that the single run time of AdPSO is the shortest, but its scheduling result is significantly weaker than that of the proposed method. GNN-RL is more complex in policy modeling than RL-Scheduler, and thus has higher running time and floating point overhead. The average running time of the proposed method is 9.1 s, which is slightly higher than that of AdPSO and GA-GWO, but still lower than that of RL-Scheduler and GNN-RL. From the perspective of the normalized comprehensive score, the proposed method is 1.73×10^7, which is significantly lower than the 2.57×10^7 of GNN-RL. This shows that the proposed model retains the ability of emphasis degree while not introducing excessive additional reasoning burden, and its operating efficiency is in the acceptable interval, which has the potential to be further verified for online deployment.

Table 3: Comparison of operating efficiency

Algorithm	Average Runtime / s	Approximate FLOPs	Normalized composite score
AdPSO	7.9	1.5×10^6	1.19×10^7
GA-GWO	8.6	1.8×10^6	1.55×10^7
RL-Scheduler	9.8	2.1×10^6	2.06×10^7
GNN-RL	10.7	2.4×10^6	2.57×10^7
Proposed Method	9.1	1.9×10^6	1.73×10^7

In order to identify the actual contribution of each component module to the overall performance, this paper again conducts ablation experiments in a mixed load scenario of 2000 tasks. Table 4 shows that the model performance degrades to varying degrees after removing any of the critical modules. Among them, after removing the dynamic decision update, the makespan increases from 1214 s to 1361 s, the resource utilization decreases from 85.6% to 79.9%, the average response delay increases to 128 ms, and the SLA violation rate also reaches 6.1%. This shows that dynamic decision update is not an additive local modification,

but a core mechanism to ensure that the model can continuously revise the direction of resource allocation. After removing the feedback disturbance correction, the congestion diffusion of the system in the burst scenario is more difficult to be cut off in time, and the makespan and response delay also show a significant deterioration. In contrast, although the heterogeneous sensing initialization and the lack of nonlinear control parameters will not lead to the complete failure of the model, they will weaken the coverage ability of candidate solutions in the early stage and the convergence quality of search in the middle and late stages, and make the overall scheduling performance drop significantly. The complete model achieves the best results on all four indicators, indicating that the three types of improvements proposed in this paper are not isolated techniques, but play complementary roles in the unified scheduling of links: initialization is responsible for expanding the feasible search range, the control mechanism is responsible for coordinating the search rhythm, and the dynamic decision update is responsible for constantly correcting deviation during operation.

Table 4: Results of ablation experiments

Model Variant	Makespan / s	Resource Utilization / %	Average Response Latency / ms	SLA Violation Rate / %
Without Heterogeneity-Aware Initialization	1278	82.1	118	4.9
Without Nonlinear Control Parameters	1314	81.4	121	5.3
Without Feedback Perturbation Correction	1336	80.7	124	5.6
Without Dynamic Decision Update	1361	79.9	128	6.1
Full Model	1214	85.6	109	3.8

Based on the above results, it can be seen that the adaptive resource scheduling algorithm and dynamic resource decision-making model proposed in this paper show better comprehensive advantages in the dimensions of execution efficiency, resource utilization, online response and operation stability. Figures 6 to 8 show that the proposed method can still maintain low completion time and response delay, and maintain higher resource utilization level under the condition of task scale expansion and load fluctuation enhancement. Table 3 shows that its scheduling overhead is not significantly out of control due to the enhancement of model structure. Table 4 further verifies that heterogeneous sensing initialization, nonlinear control and dynamic decision update all have substantial contributions to the performance improvement. The experimental results show that the proposed method can effectively alleviate the problems of task congestion, node imbalance and policy rigidity in heterogeneous resource pools, and provide a more reliable implementation path for the subsequent online resource scheduling in heterogeneous computing platforms.

6 Discussion

The experimental results show that the adaptive resource scheduling algorithm and dynamic resource decision model proposed in this paper can be more stable than the comparison methods in heterogeneous computing environment, and its advantages are mainly reflected in three aspects: total completion time control, resource utilization improvement and response delay suppression. Compared with the scheduling methods that rely on static priorities or fixed search rules, the proposed model still maintains good convergence quality when

large-scale tasks continue to arrive, which indicates that the improvement strategy does not stop at the parameter repair at the local search level, but improves the direction and adaptability of scheduling search through heterogeneous perception initialization, nonlinear control update and feedback-driven correction. Especially in high load scenarios, the algorithm does not concentrate on a few high-performance nodes too early, but can adjust the task mapping in time according to the change of node status, so it shows better load distribution ability in complex resource pools. Compared with reinforcement learning or graph structure enhanced scheduling methods, the proposed method achieves better comprehensive performance while maintaining low computational overhead, which has strong engineering significance. Although some complex models can learn fine-grained scheduling policies in specific scenarios, the cost of model training and online inference is relatively high. When the system state changes rapidly, the policy update may not be able to keep up with the synchronization. The proposed method adopts a lightweight closed-loop structure of state awareness and decision making, and completes the dynamic modeling of heterogeneous resource states without significantly increasing the scheduling overhead, so it is more suitable for deployment in the computing platform that requires continuous online response. From the mechanism level, heterogeneous aware initialization expands the coverage of candidate solutions in the resource space and reduces the aggregation phenomenon caused by strong resource preference in the initial stage. The nonlinear control parameter improves the switching rhythm between global exploration and local exploitation in the search process, so that the algorithm retains enough search flexibility in the early stage and converges to the high-quality solution region faster in the later stage. The dynamic resource decision model further introduces execution feedback into the parameter update process, which enables the scheduler to respond more timely to congestion, energy consumption fluctuation and priority pressure. It is the connection of these links that makes the proposed method not only suitable for task assignment in cloud-edge-end collaborative computing, but also scalable to multiple accelerator clusters, intelligent manufacturing scheduling, dynamic load balancing and other similar scenarios.

7 Conclusions

In this paper, an adaptive resource scheduling algorithm and a dynamic resource decision model are proposed to solve the scheduling problem of complex task structure, diverse resource types and dynamic running state in heterogeneous computing environment. In this study, starting from the task-resource mapping modeling, a unified representation mechanism for heterogeneous nodes such as CPU, GPU and FPGA is constructed, and on this basis, a basic scheduling framework is designed, an improved adaptive scheduling strategy and a feedback-driven dynamic decision-making process are designed, so that resource allocation does not stop at static rule matching. And it can continuously complete online correction and optimization according to the change of system state. Compared with the traditional methods that only focus on a single completion time or local load, the proposed model emphasizes the collaborative balance among execution efficiency, resource utilization, delay control and operation stability. Experimental results show that the proposed method achieves better performance under different task sizes and multi-class workload trajectories, and is generally better than the comparison algorithms in terms of makespan, resource utilization and average response time. Moreover, ablation experiments further verify that heterogeneous sensing initialization, nonlinear control parameters and dynamic decision update have substantial contributions to performance improvement. This shows that the method framework constructed in this paper can effectively alleviate the problems of node imbalance, task

congestion and policy hardening in heterogeneous resource pools, and maintain good convergence quality and adaptability in complex scenarios. In addition to the algorithm performance, the proposed method also has certain engineering transferability. Its modeling method and decision-making link are relatively clear, and it can connect with the scheduling modules in cloud platforms, edge clusters and multi-accelerator computing systems, providing a realizable technical path for online resource management on heterogeneous infrastructure. Future research can further extend the model by combining network jitter, failure recovery and cross-node migration overhead in real platforms, so as to improve its robustness and generalization ability in actual deployment environments.

References

[1] Xue F, Hai Q, Dong T, et al. A deep reinforcement learning based hybrid algorithm for efficient resource scheduling in edge computing environment[J]. Information Sciences, 2022, 608: 362-374.

[2] Zheng T, Wan J, Zhang J, et al. Deep reinforcement learning-based workload scheduling for edge computing[J]. Journal of Cloud Computing, 2022, 11(1): 3.

[3] Nabi S, Ahmad M, Ibrahim M, et al. AdPSO: adaptive PSO-based task scheduling approach for cloud computing[J]. Sensors, 2022, 22(3): 920.

[4] Kruekaew B, Kimpan W. Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning[J]. Ieee Access, 2022, 10: 17803-17818.

[5] Mahmoud H, Thabet M, Khafagy M H, et al. Multiobjective task scheduling in cloud environment using decision tree algorithm[J]. IEEE access, 2022, 10: 36140-36151.

[6] Pradhan A, Bisoy S K, Kautish S, et al. Intelligent decision-making of load balancing using deep reinforcement learning and parallel PSO in cloud environment[J]. IEEE Access, 2022, 10: 76939-76952.

[7] Mampage A, Karunasekera S, Buyya R. Deep reinforcement learning for application scheduling in resource-constrained, multi-tenant serverless computing environments[J]. Future Generation Computer Systems, 2023, 143: 277-292.

[8] Song Y, Li C, Tian L, et al. A reinforcement learning based job scheduling algorithm for heterogeneous computing environment[J]. Computers and Electrical Engineering, 2023, 107: 108653.

[9] Sharif Z, Jung L T, Ayaz M, et al. Priority-based task scheduling and resource allocation in edge computing for health monitoring system[J]. Journal of King Saud University-Computer and Information Sciences, 2023, 35(2): 544-559.

[10] Saif F A, Latip R, Hanapi Z M, et al. Multi-objective grey wolf optimizer algorithm for task scheduling in cloud-fog computing[J]. IEEe Access, 2023, 11: 20635-20646.

[11] Wang Z, Goudarzi M, Gong M, et al. Deep reinforcement learning-based scheduling for optimizing system load and response time in edge and fog computing environments[J].

Future Generation Computer Systems, 2024, 152: 55-69.

[12] Zhou X, Yang J, Li Y, et al. Deep reinforcement learning-based resource scheduling for energy optimization and load balancing in SDN-driven edge computing[J]. Computer Communications, 2024, 226: 107925.

[13] Zhang Z, Xu C, Xu S, et al. Towards optimized scheduling and allocation of heterogeneous resource via graph-enhanced EPSO algorithm[J]. Journal of Cloud Computing, 2024, 13(1): 108.

[14] Pei X, Sun P, Hu Y, et al. Multi-resource interleaving for task scheduling in cloud-edge system by deep reinforcement learning[J]. Future Generation Computer Systems, 2024, 160: 522-536.

[15] Xing Y. Work scheduling in cloud network based on deep Q-LSTM models for efficient resource utilization[J]. Journal of Grid Computing, 2024, 22(1): 36.

[16] Zhang Z, Xu C, Liu K, et al. A resource optimization scheduling model and algorithm for heterogeneous computing clusters based on GNN and RL: Z. Zhang et al[J]. The Journal of Supercomputing, 2024, 80(16): 24138-24172.

[17] Zhou G, Tian W, Buyya R, et al. Deep reinforcement learning-based methods for resource scheduling in cloud computing: A review and future directions[J]. Artificial Intelligence Review, 2024, 57(5): 124.

[18] Devi N, Dalal S, Solanki K, et al. A systematic literature review for load balancing and task scheduling techniques in cloud computing[J]. Artificial Intelligence Review, 2024, 57(10): 276.

[19] Rana N, Jeribi F, Khan Z, et al. A systematic literature review on contemporary and future trends in virtual machine scheduling techniques in cloud and multi-access computing[J]. Frontiers in Computer Science, 2024, 6: 1288552.

[20] Behera I, Sobhanayak S. Task scheduling optimization in heterogeneous cloud computing environments: A hybrid GA-GWO approach[J]. Journal of Parallel and Distributed Computing, 2024, 183: 104766.