2025 International Conference on Intelligent Computing
July 26-29, Ningbo, China
https://www.ic-icc.cn/2025/index.php

The Task Scheduling of IMA based on The Multi-stage Q-learning Differential Evolution

Shuying Feng[1[0009-0006-5315-4814]], Lisong Wang[1(✉)[0000-0001-6482-3717]], Shaohan Liu[1[0000-0003-4290-6591]], Fengtao Xu[2[0009-0009-6308-4946]], Yizhuo Sun[2[0009-0000-7755-8561]]

[1] Nanjing University of Aeronautics and Astronautics, Nanjing, China
[2] China Academy of Launch Vehicle Technology, Beijing 100076, China
fengshuying@nuaa.edu.cn

Abstract. With the increasing complexity of the integrated modular avionics (IMA) and the growing demand for efficient operation in multi-task environments, IMA systems must not only utilize various resources efficiently but also consider communication latency, system safety and real-time responsiveness during task execution. Because of the number and complexity of tasks increasing, the system faces dual challenges: real-time task scheduling and resource utilization optimization. Therefore, we propose a task scheduling method based on a multi-stage Q-learning differential evolution algorithm. First, a bilevel scheduling model for IMA systems is constructed, which comprehensively considering key factors such as resource utilization, communication latency and safety. Second, an enhanced differential evolution algorithm is employed to optimize the model. Specifically, in the population initialization stage, the proposed algorithm uses a chaotic logistic map to ensure a uniform distribution of the initial population in the solution space. During the population evolution process, the fitness of each infeasible individual is corrected through the penalty function. Meanwhile, the proposed algorithm utilizes a Q-learning mechanism to dynamically adjust the evolutionary operators to improve their adaptability and employs a multi-stage constraint addition strategy to expand the search space of the algorithm. Finally, experimental results comparing the proposed algorithm with other algorithms demonstrate its superior performance, which indicates its effectiveness in solving task scheduling problems in integrated avionics systems.

Keywords: IMA Task Scheduling, Constraint Handling, Differential Evolution (DE), Q-learning Algorithm, Adaptive Operator.

1 Introduction

With the rapid development of aviation technology, traditional avionics system architectures can't meet the complex requirements of modernization. To enhance system operational efficiency and ensure flight safety and reliability, IMA system has emerged [1]. Currently, several advanced aircraft have adopted the IMA architecture, for instance, the Boeing 787, Airbus A350, and COMAC C919 [1,2]. In IMA system, task scheduling follows the bilevel scheduling model defined by the ARINC 653 standard.

This model includes both inter-partition scheduling and intra-partition scheduling and ensures that all applications execute within spatially and temporally separated partitions [3]. By this partitioning mechanism, the IMA architecture guarantees that tasks running in different partitions do not interfere with each other [4]. Compared to traditional federated and integrated avionics systems, the IMA architecture has the advantages of time-partitioning and resource sharing, which effectively enhance the operational efficiency and fault tolerance of the system. However, these advantages also make the system more complex and present new challenges for system design. In the design process of IMA architecture, the core challenge is how to effectively allocate tasks while ensuring system schedulability [5]. Kim et al. [6] solved the maximum task load boundaries for each IMA partition using linear programming. Davis et al. [7] used both greedy algorithms and exhaustive search algorithms to explore the entire solution space, obtaining a globally optimal set of parameters. Al-Sheikh et al. [8] proposed an optimal response algorithm based on game theory to obtain the optimal solution. Considering the optimization of the IMA task scheduling system is a complex non-linear and non-convex optimization problem [9], the traditional optimization algorithms used in the above studies are difficult to meet these multi-objective requirements at the same time.

The complexity of task scheduling in IMA system requires balancing multiple conflicting objectives and constraints. Single-objective optimization algorithms struggle with this challenge, so an appropriate multi-objective optimization algorithm is key to solving the IMA task scheduling problem. CHEN [10] improved the decomposition-based multi-objective evolutionary algorithm (MOEA/D) by introducing the constrained dominance principle to solve the IMA partition parameter configuration problem. However, this method relies on too many parameters. Chu [11] solved the IMA resource parameter configuration problem using a forward-checking algorithm and an NSGAII with an elite retention strategy. However, this approach is prone to uneven solution distribution and premature convergence. Aminifar et al. [12] addressed real-time system task allocation using a simulated annealing algorithm, but the model is easy to getting trapped in local optima. Shojafar et al. [13, 14] optimized job allocation in cloud computing using fuzzy theory and genetic algorithms, but their study did not address constraints. The DE algorithm [15] is widely used in multi-objective optimization because of its strong global search capability, fast convergence speed, ease of implementation and simple parameter settings. However, traditional DE algorithms have some limitations. For instance, they initialize populations randomly, which may cause the initial population to be concentrated in certain regions of the solution space. Additionally, they use fixed evolutionary operators, which can lead to slow and unstable convergence.

To address this, this paper proposes an improved DE-based solution for the task scheduling problem in IMA systems. The main contributions of this paper are summarized as follows:

- A bilevel scheduling model is constructed, including task scheduling and partition scheduling, which considers resource constraints, communication delays, and system safety.

- A multi-stage Q-learning DE algorithm is designed, which optimizes aspects such as population initialization, constraint handling, and adaptive adjustment of crossover and mutation operators.
- Simulation experiments validate the proposed algorithm's superiority in optimization performance, demonstrating its potential for solving complex system task scheduling problems.

2 System Model and Problem Formulation

2.1 The Bilevel Scheduling Model

The IMA system typically employs a distributed processing framework. It consists of multiple embedded processing nodes interconnected via the AFDX network. Each processing node can host multiple partitions, and each partition allocated one or more predefined real-time tasks, as illustrated in **Fig. 1**.

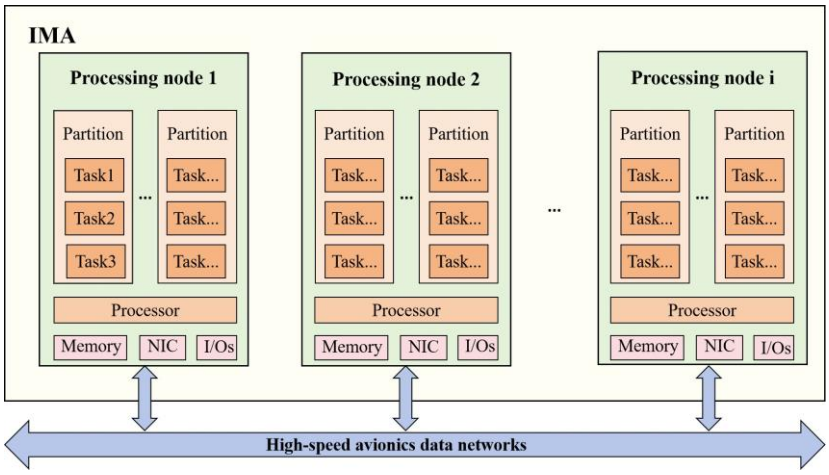

Fig. 1. IMA system framework.

We define a bilevel scheduling model as shown in **Fig. 2**. The first level scheduling divides tasks into task groups, and each group resides in a partition. The second level scheduling assigns partitions to nodes and establishes a scheduling model to optimize and obtain the final allocation results. We define a set of tasks as $T = \{t_1, t_2, \dots, t_{n^T}\}$. Each task can be represented as a triple $t_1 = (TM_i, C_i, D_i)$, where TM_i is the memory requirement, C_i is the maximum execution time, and D_i is the deadline. Task priorities are determined using the Earliest Deadline First (EDF) algorithm. The set of n^P partitions is defined as $\{p_1, p_2, \dots, p_{n^P}\}$. Each partition is characterized by $p_j = (PM_j, PT_j)$, where PM_j is the partition's memory size and PT_j is the partition frame time. A set of n^N nodes is represented as $\{n_1, n_2, \dots, n_{n^N}\}$. Each node is defined as $n_k = (NM_k, NT_k)$, where NM_k is the memory capacity and NT_k represents the main frame time.

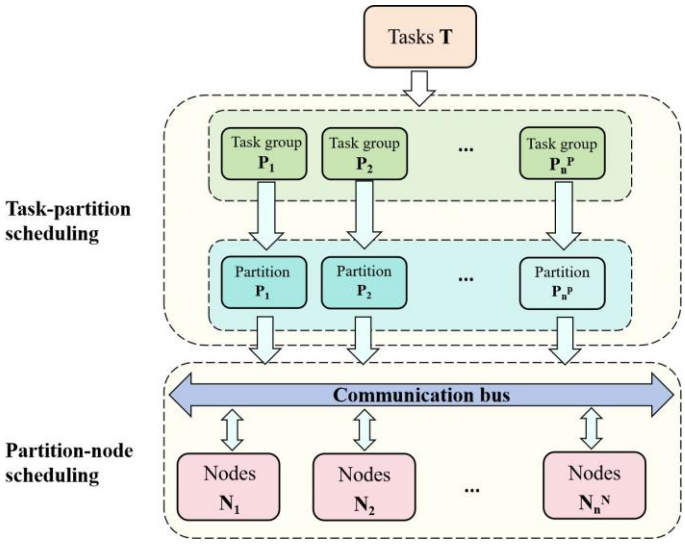

Fig. 2. The bilevel scheduling model.

2.2 Task Scheduling Problem Modeling

Task-to-Partition Scheduling. In this scheduling level, A^T is defined as an allocation matrix, where A_{ij}^T represents the allocation of task t_i to partition p_j. Considering safety factors in the scheduling process, some tasks cannot be assigned to the same partition. To address this, we define M^T as the task mutual exclusion matrix, where M_{ij}^T indicates that task t_i and task t_j are mutually exclusive.

To meet the different resource requirements of tasks, some constraints must be satisfied during the allocation process.

The schedulability constraint requires that the completion time of each task does not exceed its deadline, which can be expressed as:

$$T_i^{finish} = \sum_{m \in hp(i)} A_{mj}^T \cdot A_{ij}^T \cdot C_m + C_i \leq D_i \tag{1}$$

where $hp(i)$ denotes the set of tasks with higher priority than task t_i.

The memory constraint requires that the total memory occupied in the same partition does not exceed the partition's available memory, which can be expressed as:

$$\sum_{i=1}^{n^T} A_{ij}^T \cdot TM_i \leq PM_j \tag{2}$$

The CPU utilization constraint requires the total execution time of tasks assigned to the same partition does not exceed the partition's frame time. It can be expressed as:

$$\sum_{i=1}^{n^T} A_{ij}^T \cdot C_i \leq PT_j \tag{3}$$

For safety reasons, two mutually exclusive tasks cannot be assigned to the same partition. This constraint can be expressed as:

$$\sum_{i=1}^{n^T} \sum_{j=1}^{n^T} \sum_{k=1}^{n^P} A_{ik}^T \cdot A_{jk}^T \cdot M_{ij}^T = 0 \tag{4}$$

To evaluate the quality of the task scheduling results, this paper employs two optimization objectives. The maximum partition utilization is used to measure the load balancing between partitions. It consists of the partition memory utilization and partition CPU utilization, and can be expressed as:

$$P^{use} = \max_{1 \le j \le n^P} \left(\lambda_1 \cdot \frac{\sum_{i=1}^{n^T} A_{ij}^T \cdot C_i}{PT_j} + \lambda_2 \cdot \frac{\sum_{i=1}^{n^T} A_{ij}^T \cdot TM_i}{PM_j} \right) \tag{5}$$

where λ_1 and λ_2 represent the weights and the sum equal to 1.

The average completion time is defined as the mean of the completion times of all tasks. This objective can be expressed as:

$$FT^{avg} = \frac{1}{n^T} \sum_{i=1}^{n^T} T_i^{finish} \tag{6}$$

In summary, the optimization objectives for task-to-partition scheduling can be expressed as:

$$min\{P^{use}, FT^{avg}\} \tag{7}$$

Partition-to-Node Scheduling. In this scheduling level, the matrix A^P is used to represent the allocation of partitions on nodes. $A_{jk}^P = 1$ represents that p_j is allocated to n_k. Matrix M^P represents the partition mutual exclusion matrix, where $M_{ij}^T = 1$ indicates that p_i and p_j are mutually exclusive. In addition, a matrix E of size $n^P \times n^P$ represents the communication delay between partitions, where E_{jk} represents the communication delay caused by p_j and p_k on different processing nodes.

We also define the memory constraint, the CPU utilization constraint and the partition mutual exclusion constraint, which are similar to the task-to-partition scheduling.

$$\sum_{j=1}^{n^P} A_{jk}^P \cdot PM_j \le NM_k \tag{8}$$

$$\sum_{j=1}^{n^P} A_{jk}^P \cdot PT_j \le NT_k \tag{9}$$

$$\sum_{i=1}^{n^P} \sum_{j=1}^{n^P} \sum_{k=1}^{n^N} A_{ik}^P \cdot A_{jk}^P \cdot M_{ij}^P = 0 \tag{10}$$

The optimization goal of partition-node scheduling comprises two parts. The maximum node usage consists of node memory usage and node CPU usage.

$$N^{use} = \max_{1 \le j \le n^N} (\mu_1 \cdot \frac{\sum_{i=1}^{n^P} A_{ij}^P \cdot PT_i}{NT_j} + \mu_2 \cdot \frac{\sum_{i=1}^{n^P} A_{ij}^P \cdot PM_i}{NM_j}) \tag{11}$$

In the formula, μ_1 and μ_2 represent the weights and their sum is 1.

The communication delay refers to the time delay caused by data exchange between partitions located on different processing nodes in a multi-processor system or a multi-partition system.

$$Cost = \sum_{i=1}^{n^P} \sum_{j=1}^{n^P} E_{ij} - \sum_{i=1}^{n^P} \sum_{j=1}^{n^P} \sum_{k=1}^{n^N} A_{ik}^P \cdot A_{jk}^P \cdot E_{ij} \quad (12)$$

In summary, the optimization goal of partition-to-node scheduling can be expressed as:

$$min\{N^{use}, Cost\} \quad (13)$$

3 Proposed Algorithm

3.1 Overall Framework of MSQ-MODE

Algorithm 1: Procedure of the proposed MSQ-MODE

Input: population size: NP; maximum number of iterations: G; set of constraints: C.
Output: population: P.

Initialize the current constraint set $curC \leftarrow \emptyset$ and external archive $A \leftarrow \emptyset$;
for $i \leftarrow 1$ to NP do
    $F(i) = random(0,1)$; $CR(i) = random(0,1)$;
$P_1 \leftarrow$ Evolve P using DE for G generations;
for $i \leftarrow 1$ to $len(Constraints)$ do
    $ifr(i) \leftarrow$ Calculate the infeasibility rate if_r of constraint i on P;
$priorC \leftarrow$ Sort constraints in descending order according to if_r;
while $curC$ is not equal to C do
    if $|A| \geq NP$ then
        $P \leftarrow$ Select N individuals from A based on the SPEA2 strategy;
    if $|A| < NP$ then
        $remainP \leftarrow$ Generate $NP - |A|$ individuals according to ($x_i = \lfloor z_i \cdot n \rfloor$ (15);
        $P \leftarrow A \cup remainP$;
    $cons \leftarrow$ Select from $priorC$ based on the priority order;
    $curC = curC \cup cons$;
    while the transformation conditions are not satisfied do
        Calculate r_k, rz_k and rn_k according to ($r_k \equiv max\{rz_k, rn_k\} \leq \epsilon$(20), ($rz_k = \max\limits_{i=1,\dots,m} \{\frac{|z_i^k - z_i^{k-l}|}{max\{|z_i^{k-l}|, \Delta\}}\}$ (21) and ($rn_k = \max\limits_{i=1,\dots,m} \{\frac{|n_i^k - n_i^{k-l}|}{max\{|n_i^{k-l}|, \Delta\}}\}$ (22);
        for $i \leftarrow 1$ to NP do
            Calculate $F(i)$ and $CR(i)$;
            $o \leftarrow$ Generate offspring based on the DE/rand/1 strategy and binary crossover strategy;
            Calculate the fitness of o based on ($F_i^T = \{P_i^{use} + P_i^T, FT_i^{avg} + P_i^T\}$ (18) or ($F_i^P = \{N_i^{use} + P_i^P, Cost_i + P_i^P\}$(19);
            Update P based on the dominance relationship between x and o, and update the rewards and actions;
            Update the Q-table;
        $A \leftarrow$ Add non-dominated feasible solutions in P;
return P;

Algorithm 1 provides the pseudocode for MSQ-MODE, and the steps are as follows: In lines 1-3, the relevant parameters are initialized. In lines 4-7, the priority of constraints is determined. In lines 9-13, the initial population for each stage is initialized. In lines 14-15, constraints are added based on priority and the stage is determined. In line 17, the maximum rate of change between the ideal and worst points over the past l generations is calculated. In lines 18-23, the population evolves. In line 24, the external archive is updated based on the non-dominated feasible solutions of the population. Finally, the final P is output.

3.2 Population Initialization based on Chaotic Logistic Mapping

The chaotic logistic map [16] can enable the generated initial population to cover the solution space more extensively and improve the global search ability. The chaotic logistic map is generally described by the following mathematical formulation:

$$z_{i+1} = \theta \cdot z_i \cdot (1 - z_i) \tag{14}$$

where $\theta \in [1,4]$.

Each individual represents a scheduling scheme, which is encoded using the real coded method shown in **Fig. 3**.

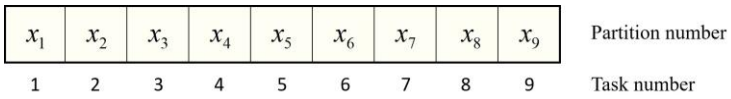

Fig. 3. Real coded.

The population initialization combined with the chaotic logistic map can be represented as:

$$x_i = \lfloor z_i \cdot n \rfloor \tag{15}$$

where x_i represents the value of the i-th position of individual x, i is the number of tasks or partitions to be scheduled, and n represents the number of partitions or nodes.

3.3 Fitness Handling Mechanism based on Penalty Function

In this section, a penalty function is used to adjust the fitness of individuals failing to satisfy the constraints. This adjustment puts them at a disadvantage during the selection process, and prevents them from negatively affecting the optimization. In the task-to-partition scheduling, the penalty value of the i-th individual's fitness can be defined as:

$$\begin{aligned} P_i^T = \quad & 10 \cdot (\textstyle\sum_{i=1}^{n^T} max(0, T_i^{finish} - D_i) \\ & + \textstyle\sum_{j=1}^{n^P} max(0, \sum_{i=1}^{n^T} A_{ij}^T \cdot TM_i - PM_j) \\ & + \textstyle\sum_{j=1}^{n^P} max(0, \sum_{i=1}^{n^T} A_{ij}^T \cdot C_i - PT_j) \\ & + \textstyle\sum_{i=1}^{n^T} \sum_{j=1}^{n^T} max(0, \sum_{k=1}^{n^P} A_{ik}^T \cdot A_{jk}^T \cdot M_{ij}^T)) \end{aligned} \tag{16}$$

Similarly, the penalty function for partition-to-node scheduling is defined as:

$$\begin{aligned} P_i^P = \quad & 10 \cdot (\sum_{j=1}^{n^N} max(0, \sum_{i=1}^{n^P} A_{ij}^P \cdot PM_i - NM_j) \\ & + \sum_{j=1}^{n^N} max(0, \sum_{i=1}^{n^P} A_{ij}^P \cdot PT_i - NT_j) \\ & + \sum_{i=1}^{n^P} \sum_{j=1}^{n^P} max(0, \sum_{k=1}^{n^N} A_{ik}^P \cdot A_{jk}^P \cdot M_{ij}^P)) \end{aligned} \tag{17}$$

Based on the penalty function, the fitness value of individual i can be modified as:

$$F_i^T = \{P_i^{use} + P_i^T, FT_i^{avg} + P_i^T\} \tag{18}$$

$$F_i^P = \{N_i^{use} + P_i^P, Cost_i + P_i^P\} \tag{19}$$

3.4 The Adaptive Operators based on Q-learning Algorithm

In this paper, each individual independently maintains and updates the operators using the Q-learning algorithm. We define an individual as an agent. Similar to [17], the states are set according to the dominance relationship between offspring and parents: the offspring dominates the parent; the parent dominates the offspring; the offspring and the parent do not dominate each other. Here, different adjustment schemes of the operators are set as actions, mainly including: $f = -0.1, c_r = 0.1$; $f = 0.1, c_r = 0.1$; $f = 0, c_r = 0$. These actions are transformed into a feedback mechanism to update the operators: $F = F + f$, $CR = CR + c_r$. If the offspring dominates the parent, it indicates that the current local search direction is effective, so the reward is 1. If the parent dominates the offspring, it suggests that the current local search direction may be ineffective, thus the reward is -1. For other situations, the reward is 0. The update of the Q-table uses the Bellman equation. The agent uses the ϵ-greedy strategy to select the action and receives the corresponding reward to update the Q-table.

3.5 Constraint-handling based on Multi-stage Constraint Addition

In multi-objective optimization problems, there are usually multiple constraints, imposing constraints too early may limit the exploration of the solution space. To ensure that the algorithm has good global search ability and convergence, we propose a multi-stage constraint addition method, which enables the algorithm to smoothly transition from the initial stage with loose constraints to the later stage with strict constraints, thereby improving the quality of the final solution. The specific steps are as follows:

Determining the priority of constraints. Firstly, we randomly generate an initial population. Then let the population evolve to the approximate unconstrained Pareto Front. Next, we calculate the infeasibility rate of the population on each constraint and sort the constraints in descending order based on the infeasibility rates.

Adding constraints stage by stage. To increase the diversity of solutions and find potential effective solution regions, we do not add any constraints in the initial stage. In the subsequent stages, one constraint will be gradually added in each round according to the priority level. As the number of constraints gradually increases, the search scope gradually narrows, and finally focuses on a solution space that satisfies all constraints.

To improve the stability and diversity of the constraint addition process, we use an external archive to store the non-dominated feasible solutions from each stage. At each stage, the initial solution set is obtained from the external archive.

Stage transition conditions. We introduce the switching strategy of PPS [18] to dynamically determine whether to add new constraints based on the optimization status of the population. The strategy is as follows:

$$r_k \equiv max\{rz_k, rn_k\} \leq \epsilon \tag{20}$$

$$rz_k = \max_{i=1,\dots,m} \{\frac{|z_i^k - z_i^{k-l}|}{max\{|z_i^{k-l}|, \Delta\}}\} \tag{21}$$

$$rn_k = \max_{i=1,\dots,m} \{\frac{|n_i^k - n_i^{k-l}|}{max\{|n_i^{k-l}|, \Delta\}}\} \tag{22}$$

where r_k denotes the maximum rate of change between the ideal and nadir points over the past l generations. z_i^k and n_i^k represent the ideal point and the nadir point in the k-th generation, respectively. ϵ is set to $1e-3$ and Δ is set to $1e-6$.

4 Experimental Results

4.1 Experiment Settings

The basic configuration information of the IMA system in the experiment is shown in **Table 1** and **Table 2**. In the task-to-partition scheduling, we set that t_2 and t_5, t_3 and t_{10}, t_9 and t_{16}, t_1 and t_{11}, t_7 and t_{10} are mutually exclusive (i.e., the corresponding positions in M^T are set to 1, and all other positions set to 0). Similarly, in the partition-to-node scheduling, we set that p_1 and p_4, p_3 and p_7 are mutually exclusive. The communication delays between partitions are shown in **Table 3**.

Table 1. Task parameters.

Task	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
TM	13	8	13	10	19	26	26	44	24
C	2	2	4	1	3	2	4	7	2
D	32	30	40	25	35	34	42	55	36
Task	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}	t_{18}
TM	48	56	26	48	50	59	43	46	49
C	3	4	1	4	5	2	4	2	3
D	39	44	26	43	45	38	41	37	50

Table 2. Partition parameters and node parameters.

Partition	p_1	p_2	p_3	p_4	p_5	p_6	p_7	Node	n_1	n_2	n_3
PM	200	120	250	190	130	110	140	NM	1000	900	500
PT	50	35	65	30	98	32	41	NT	300	200	100

Table 3. Partition Communication Delay.

Partition	p_1	p_2	p_3	p_4	p_5	p_6	p_7
p_1	0	45	23	67	12	9	34
p_2	45	0	12	43	56	78	90
p_3	23	12	0	34	67	21	89
p_4	67	43	65	0	65	32	54
p_5	12	56	67	65	0	56	87
p_6	89	78	21	32	56	0	43
p_7	34	90	89	54	87	43	0

NSGAII [19], CMOEA/D [20], CCMO [21] and PPS [18] are selected as the comparison algorithms for MSQ-MODE. NSGAII and CMOEA/D are very representative algorithms, and CCMO and PPS are multi-stage constrained MOEAs. We set the crossover and mutation operators for NSGAII, CMOEA/D and CCMO to 0.9 and 0.1, respectively. The population size is set to 100 and the maximum number of iterations set to 150. In CMOEA/D, the number of neighbors is set to 5, and the evolutionary process uses the DE component (DE/rand/1 and binary crossover). The remaining parameters of these algorithms align with those specified in the original papers. To account for the randomness of the algorithms, each algorithm is run 30 times.

Since the true Pareto front of the IMA task scheduling problem is difficult to obtain directly, the IGD index cannot be used for this problem. We adopt the hypervolume (HV) as the performance index. The larger the HV value, the better the comprehensive performance of the algorithm.

4.2 Result Analysis

Effectiveness test. Table 4 lists the max, min, mean and std HV values for each algorithm obtained in the bilevel scheduling model. From **Table 4**, MSQ-MODE achieves the best mean HV values in both task-to-partition scheduling and partition-to-node scheduling. This indicates that the algorithm is capable of consistently producing high-quality solution sets over multiple runs, demonstrating strong global search ability and consistency. Additionally, MSQ-MODE also performs best in terms of both the max and min HV values, indicating that the algorithm can find high quality solutions and reflect its strong ability to explore the potential solution space. In contrast, in the task-partition scheduling, the minimum HV values of CCMO and PPS are 0, and in the partition-node scheduling, the minimum HV values of CMOEA/D and PPS are 0, which indicates that they failed to find any feasible solutions during the evolutionary process.

As we can see from **Fig. 4** and **Table 4**, our algorithm has the smallest std for HV, indicating that it is more stable compared to other algorithms. This is because our algorithm introduces a penalty function in the fitness evaluation, reducing ineffective searches within the population and guiding the evolution direction towards the feasible solution set, ensuring convergence stability. Moreover, our algorithm uses the Q-learning mechanism to ensure that individuals evolve in favorable directions, which not only

improves the average HV value but also reduces the variance in convergence paths across different experiments, thereby lowering the standard deviation. In addition, our algorithm employs a multi-stage constraint-handling strategy, and through gradual convergence, it reduces fluctuations caused by randomness, leading to more consistent solution quality across different experiments and ultimately resulting in a smaller standard deviation.

Significance test. Table 4 lists the P-values and test results of the Wilcoxon signed-rank test comparing MSQ-MODE with NSGAII, CMOEA/D, CCMO and PPS at a significance level of 0.05. In the table, the symbols "+", "≈", and "−" represent that the performance of MSQ-MODE is significantly better than that of the competitors, similar to that of the competitors, and significantly worse than that of the competitors, respectively. **Table 4** shows that, from a statistical perspective, the MSQ-MODE algorithm significantly outperforms other comparison algorithms. Notably, the Wilcoxon signed-rank test, which is specifically designed to compare median differences between two related samples, furnishes robust and incontrovertible evidence. This evidence strongly supports the assertion that MSQ-MODE not only attains an exceptionally high optimization quality in the task scheduling problem but also exhibits strikingly significant differences in its optimization outcomes. Therefore, the test results further confirm that the MSQ-MODE algorithm has superior performance and higher stability compared to other comparison algorithms in solving this problem. This strongly validates its applicability and reliability in practical applications.

Table 4. The min, max, mean, std of the HV values for different algorithms and the P-values and test results of the Wilcoxon signed-rank test comparing different algorithms at a significance level of 0.05.

Scheduling model	Algorithm	Min	Max	Mean	Std	P-value	R
Task-to-partition	MSQ-MODE	1.1123	1.2057	1.1598	0.0221	-	-
	NSGAII	0.7386	1.1928	0.9329	0.0828	3.72529E-09	+
	CMOEA/D	1.0109	1.1807	1.0925	0.0491	1.30385E-07	+
	CCMO	0.0000	1.1807	0.8579	0.4768	1.21836E-05	+
	PPS	0.0000	0.8935	0.3617	0.3472	1.86265E-09	+
Partition-to-node	MSQ-MODE	0.1865	0.1882	0.1879	0.0004	-	-
	NSGAII	0.1784	0.1874	0.1847	0.0026	1.86265E-09	+
	CMOEA/D	0.0000	0.0122	0.0076	0.0047	1.86265E-09	+
	CCMO	0.1241	0.1807	0.1544	0.0136	1.73255E-09	+
	PPS	0.0000	0.0122	0.0065	0.0054	1.86265E-09	+

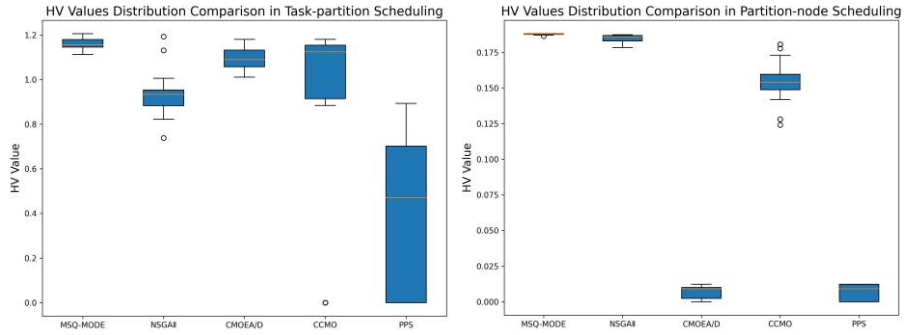

Fig. 4. HV values distribution of the bilevel scheduling model.

5 Conclusion

In this article, the task scheduling problem in integrated electronic systems is studied. The core of this problem is to optimize the task scheduling scheme while satisfying the constraints of resources, communication delays, and safety. This paper proposes a two-layer scheduling model based on task scheduling, which is used to optimize task-partition scheduling and partition-node scheduling. Subsequently, a multi-stage Q-learning differential evolution algorithm (MSQ-MODE) is proposed. This algorithm combines chaotic initialization, an adaptive penalty function, a dynamic adjustment mechanism based on Q-learning, and a multi-stage constraint addition strategy. Compared with other heuristic algorithms, MSQ-MODE has stronger adaptability and better optimization performance. In the future, combining machine learning techniques to automatically learn and optimize task scheduling strategies will become an important research direction.

References

1. Gaska T, Watkin C, Chen Y: Integrated modular avionics-past, present, and future. IEEE Aerospace and Electronic Systems Magazine. 2015 Sep;30(9):12-23.
2. RTCA (Firm). SC-200: Integrated Modular Avionics (IMA) Development: Guidance and Certification Considerations. RTCA; 2005.
3. Kim D, Lee YH, Younis M: Software architecture supporting integrated real-time systems. Journal of Systems and Software. 2003 Jan 15;65(1):71-86.
4. Watkins CB, Walter R: Transitioning from federated avionics architectures to integrated modular avionics. In 2007 IEEE/AIAA 26th Digital Avionics Systems Conference. In IEEE, Oct. 2007.
5. Annighoefer B, Nil C, Sebald J, Thielecke F. Structured and symmetric IMA architecture optimization: Use case ariane launcher. In2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC) 2015 Sep 13 (pp. 6B3-1). IEEE.
6. Kim JE, Abdelzaher T, Sha L: Schedulability bound for integrated modular avionics partitions. In2015 Design, Automation & Test in Europe Conference & Exhibition (DATE) 2015 Mar 9, pp. 37-42. IEEE.

7. Davis R, Burns A. An investigation into server parameter selection for hierarchical fixed priority pre-emptive systems. In16th International Conference on Real-Time and Network Systems (RTNS 2008) 2008 Oct 16.
8. Al Sheikh A, Brun O, Hladik PE, Prabhu BJ. A best-response algorithm for multiprocessor periodic scheduling. In 2011 23rd Euromicro conference on real-time systems. IEEE, 2011, pp. 228–237.
9. Yoon MK, Kim JE, Bradford R, Sha L. Holistic design parameter optimization of multiple periodic resources in hierarchical scheduling. In 2013 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2013, pp. 1313–1318.
10. Chen H, Zhang W, Lyu Y. Research on partition parameter design method for integrated modular avionics based on MOEA/D-ADV. IEEE Access, vol. 8, pp. 117 278–117 297, 2020.
11. Chu J, Zhao T, Jiao J, Chen Z. Optimal design of configuration scheme for integrated modular avionics systems with functional redundancy requirements. EEE Systems Journal, vol. 15, no. 2, pp. 2665–2676, 2020.
12. Aminifar A, Eles P, Peng Z. Jfair: A scheduling algorithm to stabilize control applications. In21st IEEE Real-Time and Embedded Technology and Applications Symposium. IEEE, 2015, pp. 63–72.
13. Shojafar M, Javanmardi S, Abolfazli S, Cordeschi N. FUGE: A joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method. Cluster Computing, vol. 18, pp. 829–844, 2015.
14. Javanmardi S, Shojafar M, Amendola D, Cordeschi N, Liu H, Abraham A. Hybrid job scheduling algorithm for cloud computing environment. In Proceedings of the fifth international conference on innovations in bio-inspired computing and applications IBICA 2014. Springer, pp. 43–52.
15. Storn R, Price K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization, vol. 11, pp. 341–359, 1997.
16. Yu Y, Gao S, Cheng S, Wang Y, Song S, Yuan F. CBSO: a memetic brain storm optimization with chaotic local search. Memetic Computing, vol. 10, pp. 353–367, 2018.
17. Yu X, Xu P, Wang F, Wang X. Reinforcement learning-based differential evolution algorithm for constrained multi-objective optimization problems. Engineering Applications of Artificial Intelligence, vol. 131, p.107817, 2024.
18. Fan Z, Li W, Cai X, Li H, Wei C, Zhang Q, Deb K, Goodman E. Push and pull search for solving constrained multi-objective optimization problems. Swarm and evolutionary computation, vol. 44, pp. 665–679, 2019.
19. Deb K, Pratap A, Agarwal S, Meyarivan TA. A fast and elitist multi-objective genetic algorithm: NSGA-II. EEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182–197, 2002.
20. Jain H, Deb K. An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach. IEEE Transactions on Evolutionary Computation, vol. 18, pp. 602–622, 2014.
21. Tian Y, Zhang T, Xiao J, Zhang X, Jin Y. A coevolutionary framework for constrained multi objective optimization problems. IEEE Transactions on Evolutionary Computation, vol. 25, no. 1, pp. 102–116, 2020.