Received April 18, 2019, accepted May 8, 2019, date of publication May 15, 2019, date of current version June 6, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2917043

Improvement of the Dynamic Priority Scheduling Algorithm Based on a Heapsort

SHANSHAN MENG[1], QIANG ZHU[2], AND FEI XIA[1], (Member, IEEE)
[1]School of Automation Engineering, Shanghai University of Electric Power, Shanghai 200090, China
[2]Shanghai Automation Instrumentation Co., Ltd., Shanghai 200230, China

Corresponding author: Fei Xia (xiafeiblue@163.com)

This work was supported in part by the National Key R&D Program of China under Grant 2017YFE0100900, and in part by the National Natural Science Foundation of China under Grant 71690234 and Grant 71871160.

ABSTRACT The earliest deadline first (EDF) scheduling algorithm is a typical representative of the dynamic priority scheduling algorithm. However, once the system is overloaded, the deadline miss rate increases and the scheduling performance deteriorates sharply, which causes a reduction in system resource utilization. To overcome this problem, we proposed an improved dynamic priority scheduling algorithm based on heap sorting. The task deadline, task value, energy consumption, and other parameters were introduced. The fuzzy analytic hierarchy process (FAHP) and the value density method were then used to determine the comprehensive priority of tasks. A heapsort algorithm with a lower time complexity was used to sort the comprehensive priority index so as to reduce the sorting overhead of the system. The system sorting overhead was then introduced to improve the decision condition of the priority scheduling subset and expand the schedulable range of the algorithm. The experimental results showed that the improved method reduced the deadline miss rate by an average of 0.1789, which improved the scheduling performance of the algorithm. The optimized scheduling algorithm can be applied to industrial control to improve system efficiency, and reasonable resource scheduling can reduce data center costs.

INDEX TERMS Real-time operating system, EDF, FAHP, heapsort, value density.

I. INTRODUCTION

In recent years, the rapid development of network technology has led to the widespread adoption of the real-time operating system in various fields. The accuracy of real-time system calculation depends not only on the logic of the algorithm but also on the time it takes to obtain results. If the system cannot meet the constraints, it can make mistakes [1]–[3]. Therefore, the primary task of a real-time operating system is to use all resources to control the tasks in real time.

Real-time tasks have their own task value, energy consumption [4], task deadline [5], [6], execution time and other reference factors. The choice of a real-time scheduling strategy is critical to the real-time performance of the system [7]. Among them, the Rate Monotonic (RM) algorithm and the Earliest Deadline First (EDF) scheduling algorithm are typical representatives of a priority scheduling algorithms [3]. In the dynamic priority scheduling strategy, when the task workload is less than 1, the performance of the EDF scheduling algorithm is superior to that of other algorithms. However, the difference in task priorities can lead to an overload of task scheduling. In this case, the EDF algorithm tends to exhibit poor scheduling performance, most tasks tend to miss their deadlines after overload [8]. Many tasks are aborted due to the influence of other tasks in the process of execution, which leads to a decrease in the task success rate and a waste of system resources [9]. Therefore, how to improve the scheduling performance in an overload scenario has become an important issue.

The scheduling algorithm proposed by Anderson *et al.* [10] has been shown to reduce the task delay rate and achieve good scheduling performance. Similarly, Barbieru and Pop [11] proposed a real-time job scheduling algorithm to solve the uncertainty of a long task completion time. The algorithm proposed by Wang *et al.* [12] effectively improved the success rate of tasks and reduced the delay time of soft real-time tasks by setting a task preemption threshold. Moreover, the improved algorithm proposed by Dong and Chen [13] has been found to effectively schedule dynamic real-time tasks in heterogeneous systems and respond to

The associate editor coordinating the review of this manuscript and approving it for publication was Amjad Mehmood.

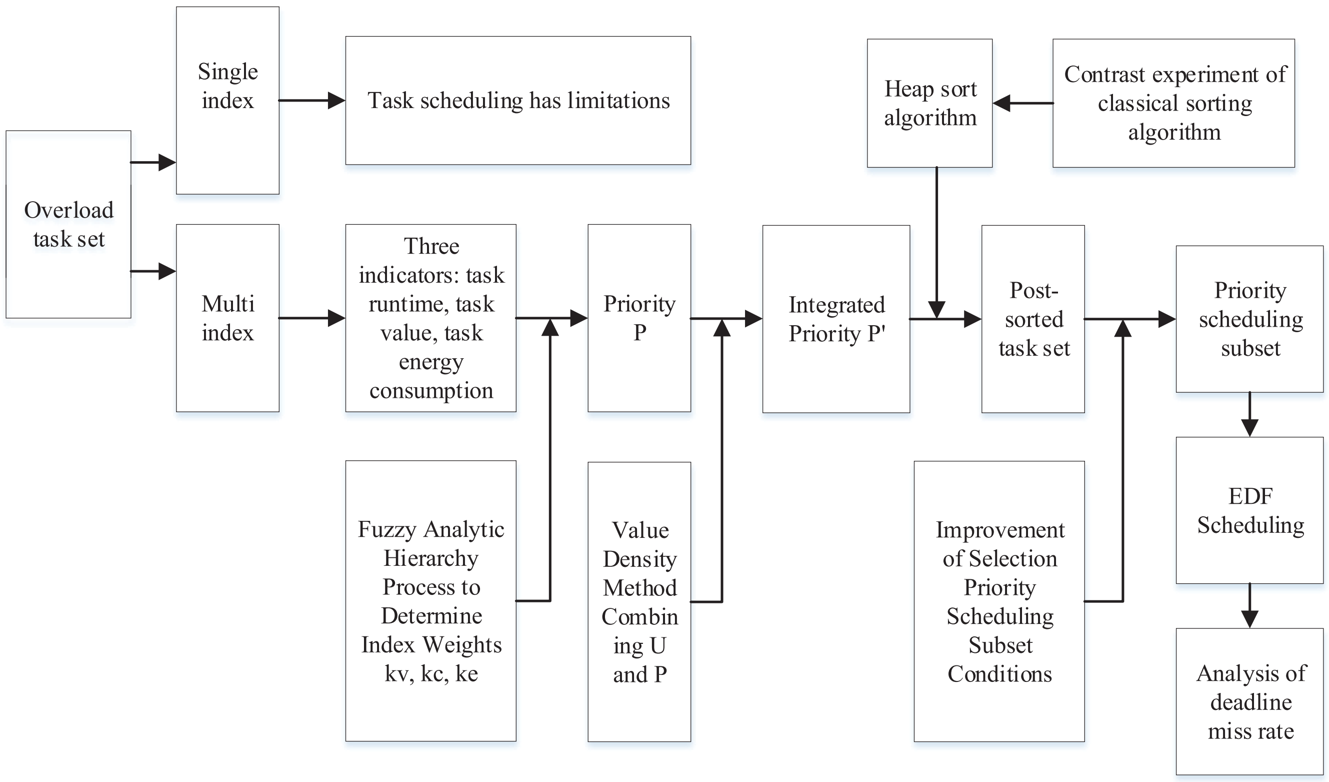
Single index
Task scheduling has limitations
Heap sort algorithm
Contrast experiment of classical sorting algorithm
Overload task set
Multi index
Three indicators: task runtime, task value, task energy consumption
Priority P
Integrated Priority P'
Post-sorted task set
Priority scheduling subset
Fuzzy Analytic Hierarchy Process to Determine Index Weights kv, kc, ke
Value Density Method Combining U and P
Improvement of Selection Priority Scheduling Subset Conditions
EDF Scheduling
Analysis of deadline miss rate

FIGURE 1. **Theoretical framework of the improved task scheduling algorithm.**

task requirements. Palopoli *et al.* [14] proposed a scheduling algorithm based on a discrete-time Markov chain of wireless state that could reduce the deadline miss rate of periodic tasks in soft real-time systems.Likewise, He *et al.* [15] proposed an adaptive multi-objective task scheduling (AMTS) strategy based to reduce the task deadline miss rate. Biondi and Sun [16] verified that the response time analysis technique was invalid and the global scheduling performance was lower than the partition scheduling. The strategy of assigning task priority dynamically, which was proposed by Wu [17], improved the system revenue and reduced the energy consumption of tasks. Sang *et al.* [18] introduced a value factor and task deadline, which ensured the priority scheduling of mission-critical tasks and reduced the failure rate of task deadlines.

In summary, real-time system scheduling research improved the success rate of task scheduling in terms of task priority, schedulable decision conditions, preemptive threshold scheduling, or hierarchical framework scheduling. However, the system overhead generated by the scheduling algorithm and sorting algorithm were neglected, as was the influence of different load values on task priority. In addition, when task priority was determined, no explicit source of weight data was given. Therefore, on the basis of the EDF scheduling algorithm, we first optimized the weight by using the Fuzzy Analytic Hierarchy Process (FAHP), which made the experimental results more reliable. Then, a heapsort was introduced to solve the problem of high system overhead in the priority index sorting part. The overloaded tasks were optimized by combining the task load value with the task priority, which increased the number of tasks scheduled and improved the utilization rate of system resources. Finally, the system overhead generated by the scheduling and sorting algorithms was introduced to improve the decision condition of the priority scheduling subset and enlarge the schedulable range of the algorithm.

II. TASK SCHEDULING MODEL

The flowchart of the improved task scheduling algorithm in this paper is shown in Fig. 1. It mainly includes the following six steps.

(1) Determine whether the system scheduling task set is overloaded. If the load of task set was less than 1, the EDF algorithm was called to execute; otherwise, the improved algorithm was used to schedule.

(2) Determine the weights of the indicators using the FAHP and then obtain the priority of the tasks P_i.Since the task scheduling was limited only by considering a single indicator, we introduced three indicators: task running time, task value, and task energy consumption.

(3) Calculate the comprehensive priority P_i', using the value density method, which combines the task load

value with the priority P_i. It not only guaranteed the priority scheduling of high-priority tasks but also increased the number of tasks to be processed and reduced the deadline miss rate of the algorithm.

(4) Sort the comprehensive priority $P_i{}'$using the heapsort algorithm because the time complexity of the heapsort algorithm was low, and the system consumption produced by the sorting algorithm during actual scheduling was reduced.

(5) Improve the conditions for selecting the priority scheduling subset. Considering the actual situation, the system sorting overhead $\Delta t'$ was introduced to ensure that the CPU utilization of all tasks in the subset was less than $1 - \frac{\Delta t'}{T}$.

(6) Schedule tasks in the priority scheduling subset using the EDF algorithm. If there was idle time, tasks in the non-priority subset were scheduled. Whenever a new task was added to the task set, it was necessary to redefine the priority of the task and refresh the priority scheduling subset.

A. SCHEDULABLE CONDITIONS OF ALGORITHMS

In real-time systems, $S = \{t_1, t_2, \cdots, t_N\}$ is defined as a scheduling task set, where $t_1, t_2, \cdots, t_N$ is the periodic tasks in the task set, T_i is the period of the task t_i, and C_i is the worst execution time of task t_i. Define 1 (Hypercycle)H is the least common multiple of the real-time task period value. Because the task runs the same in different cycles, only the first hypercycle$[0, H]$needs to be discussed. The maximum number of task instances executed in a hypercycle is as follows:

$$J = \sum_{i=2}^{N} \frac{H}{T_i}. \tag{1}$$

Define 2 (Periodic Task Set Load) The periodic load is the average occupancy of the processor for one periodic task. System Periodic Task Set Load is the sum of all Periodic Task Loads [19], denoted as U, which can be expressed as follows:

$$U = \sum_{i=1}^{n} U_i = \sum_{i=1}^{n} \frac{C_i + \Delta t}{T_i}, \tag{2}$$

whereΔt refers to the system overhead under the scheduling of the EDF algorithm.

The EDF scheduling algorithm is used to schedule task set S. If only the periodic task load $U \leq 1$, the system is in a low-load state, the performance of the EDF scheduling algorithm is better. The algorithm can effectively schedule tasks in the task set. However, when task load $U > 1$, the system is overloaded and some tasks will miss deadlines. At this time, the task set cannot be scheduled.

The traditional EDF scheduling algorithm relies on a single indicator (deadline) to determine the scheduling order of tasks; however, this is unreasonable in an actual situation, it would also increase the deadline miss rate of task scheduling. In order to solve the problem of the poor performance of the EDF scheduling algorithm and the high deadline missing rate under system overload, we introduced multiple related indicators to obtain a more scientific and comprehensive priority.

III. TASK INTEGRATED PRIORITY

Determining comprehensive priority is integral to reducing the miss rate of deadlines. However, it is too one-sided to determine the priority of scheduling tasks through a single indicator. To solve this problem, Wang *et al.* [20] proposed adding task value V_i, task running timeC_i and the energy consumption of task E_i, to determine the priority of the task. Among them, k_e, k_c and k_v are the weight values of parameters such as the energy consumption of tasks, task running time, and task value. The exact values of k_e, k_c and k_v are calculated by using the FAHP. In this study, the running time of tasks and the energy consumption of tasks were inversely proportional to priority; therefore, the task priority P_i was defined as follows:

$$P_i = k_v \cdot V_i + k_c \cdot \frac{\sum_{i=1}^{N} C_i}{C_i} + k_e \cdot \frac{\sum_{i=1}^{N} E_i}{E_i}. \tag{3}$$

In order to reduce the deadline miss rate for task scheduling, more tasks had to be processed in task centralization so that the utilization rate of the CPU was close to 1. The CPU utilization was also the load U of the periodic task set. As shown in (2), the number of tasks that could be processed by the scheduling algorithm was closely related to U. The smaller the load value U_i of each cycle task, the more tasks the CPU could handle. The task scheduling process based on priority indicator P_i could only satisfy the priority scheduling of the high-priority tasks, but it could not increase the number of tasks scheduling or reduce the deadline miss rate of task scheduling. Therefore, the periodic task load value U_i had to be introduced into the calculation of task integrated priority $P_i{}'$, as follows:

$$P_i{}' = \frac{P_i}{U_i}, \tag{4}$$

$$P_i{}' = \frac{(k_v \cdot V_i + k_c \cdot \frac{\sum_{i=1}^{N} C_i}{C_i} + k_e \cdot \frac{\sum_{i=1}^{N} E_i}{E_i})}{U_i}. \tag{5}$$

A. DETERMINING k_v, k_c AND k_e BASED ON FAHP

1) ESTABLISHING THE HIERARCHICAL STRUCTURE MODEL

The established hierarchical model shown in Fig. 2, which included the solution layer, the criteria layer and the target layer, which were used to determine the values of targets k_v, k_c andk_e. The target layer aimed to get reasonable k_v, k_c and k_e values. The criteria layer influenced the factor affecting the deadline miss rate, which included task running time, task value and task energy consumption. The solution layer supplied five groups of k_v, k_c and k_e for the five experts, as shown in Table 1.

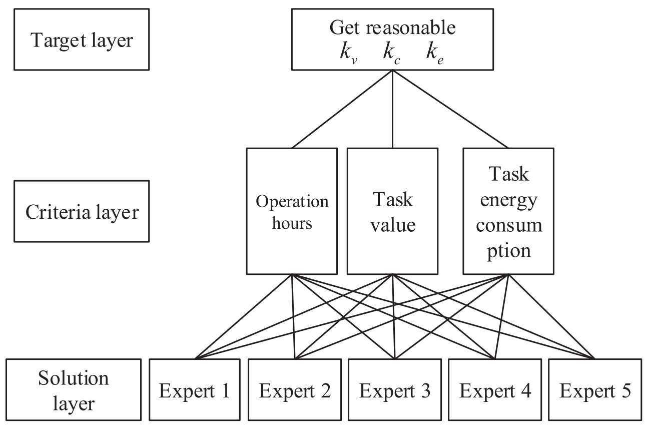

FIGURE 2. Hierarchical structure model diagram.

2) FUZZY CONSISTENCY MATRIX

Compared with the analytic hierarchy process (AHP), the FAHP improved the solving process of the judgment matrix, transformed the judgment matrix into the fuzzy consistency matrix, omitted the process of a consistency check, and overcame the inconsistency of the AHP [21]. First, the relative scalar method of the AHP was used to construct the judgment matrix A. In the relative scalar method, a_{ij} is the important degree of index i relative to index j in the criterion layer to the target and is expressed by the value of 0.1 0.9, in which$a_{ji} = 1 - a_{ij}(i, j = 1, 2, 3)$.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{array}{c} \\ v \\ c \\ e \end{array} \begin{array}{c} v \; c \; e \\ \begin{bmatrix} 0.50 & 0.01 & 0.95 \\ 0.99 & 0.05 & 0.01 \\ 0.05 & 0.01 & 0.5 \end{bmatrix} \end{array}. \quad (6)$$

After the consistency transformation of formula $r_i = \sum_{i=1}^{n} a_{ij}$ and $r_{ij} = \frac{r_i - r_j}{2n} + 0.5$, the fuzzy complementarity matrix $R = (r_{ij})_{n \times n}$ was obtained.

Based on the definition of the fuzzy complementary matrix, the necessary and sufficient condition for a fuzzy complementary matrix $R = (r_{ij})_{n \times n}$ to be a fuzzy consistent matrix is that the difference between the two corresponding elements of the arbitrarily designated two rows is constant [22]. Eq. (7) shows that the matrix R satisfied the necessary and sufficient conditions of the fuzzy consistent matrix and conformed to the standard of people's thinking and judgment; therefore, no further consistency test was needed:

$$R = \begin{array}{c} \\ v \\ c \\ e \end{array} \begin{array}{c} v \quad c \quad e \\ \begin{bmatrix} 0.50 & 0.33 & 0.65 \\ 0.67 & 0.50 & 0.82 \\ 0.35 & 0.18 & 0.50 \end{bmatrix} \end{array}. \quad (7)$$

In order to calculate the ranking vector $\omega^{(0)}$, the power method with a higher precision was used. The fuzzy complementary matrix $R = (r_{ij})_{n \times n}$ was converted into a reciprocal matrix $E = (e_{ij})_{n \times n}$ using the conversion formula $e_{ij} = \frac{r_{ij}}{r_{ji}}$. The ranking vector $\omega^{(0)}$ was used as the initial vector $V^{(0)}$, and the formulas $V^{(k+1)} = EY^{(k)}$ and $Y^{(k)} = V^{(k)}/\left\|V^{(k)}\right\|_{\infty}$, $k = 1, 2, \cdots$ were used to iterate. If $\left\|V^{(k+1)}\right\|_{\infty} - \left\|V^{(k)}\right\|_{\infty} < \varepsilon$ and ε were given errors, then $\left\|V^{(k+1)}\right\|_{\infty}$ was the maximum eigenvalue, and $P = [V_{k+1,1}/\sum_{i=1}^{n} V_{k+1,i}, \cdots, V_{k+1,n}/\sum_{i=1}^{n} V_{k+1,i}]^T$ was the sorting vector; otherwise, the iteration would continue. Finally, a high-precision sorting vector $\omega^{(k)}$ was obtained.

Similarly, by comparing the effects of k_v, k_c and k_e given by the five experts to reduce the miss rate of the task deadlines, three judgment matrices $B_i(i = 1, 2, 3)$ were constructed and transformed into a fuzzy consistency matrix $R_i(i = 1, 2, 3)$. Then, the high precision sorting vector $\omega_i^{(k)}(i = 1, 2, 3)$ was calculated.

3) COMBINATION WEIGHT OF FUZZY CONSISTENCY MATRIX

Local Weight $\omega(i)$ of Indicator Factors for Objects:

$$\omega = \begin{bmatrix} 0.2726 & 0.5895 & 0.1378 \end{bmatrix}. \quad (8)$$

Local Weight $\omega_i(k)$ of Experts for Indicators:

$$\omega_i = \begin{array}{c} \\ w_1 \\ w_2 \\ w_3 \end{array} \begin{array}{c} Exp1 \quad Exp2 \quad Exp3 \quad Exp4 \quad Exp5 \\ \begin{bmatrix} 0.2424 & 0.1394 & 0.0885 & 0.1278 & 0.4019 \\ 0.1139 & 0.2370 & 0.1164 & 0.1164 & 0.4162 \\ 0.0882 & 0.2021 & 0.2296 & 0.1216 & 0.3585 \end{bmatrix} \end{array}. \quad (9)$$

4) TARGET OPTIMAL SOLUTION

The comprehensive weights of the experts were obtained through the local weights of the judgment matrix at the criterion level and the scheme level, as follows:

$$W_k = \sum_{i=1}^{3} \omega(i)\omega_i(k) = \begin{array}{c} Exp1 \quad Exp2 \quad Exp3 \quad Exp4 \quad Exp5 \\ [0.1454 \;\; 0.2056 \;\; 0.1244 \;\; 0.1202 \;\; 0.4044] \end{array}. \quad (10)$$

Experts gave specific values for k_v, k_c and k_e, as shown in Table 1. The optimal solution is shown in Table 2. The weights were summed using the comprehensive weights and the given values of k_v, k_c and k_e.

The priority of different tasks was determined by using the weight values of energy consumption, task running time, task value and the optimal solution obtained by the FAHP method.

TABLE 1. k_v, k_c and k_e values given by experts.

Indicator	k_v	k_c	k_e
Expert1	0.1	0.3	0.6
Expert2	0.01	0.95	0.04
Expert3	0.6	0.3	0.1
Expert4	0.5	0.3	0.2
Expert5	0.05	0.85	0.1

TABLE 2. Optimal solution of FAHP method.

Optimal solution	target		
	k_v	k_c	k_e
FAHP	0.1716	0.656	0.1724

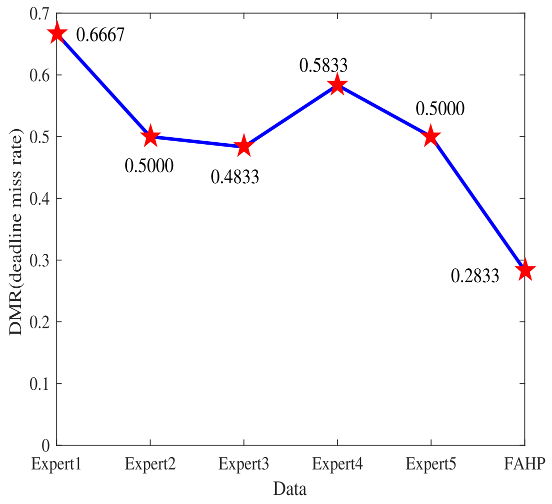

FIGURE 3. Comparison of deadline miss rates after optimization.

Scheduling experiments were then carried out to verify the deadline miss rate of the scheduling task set.

As shown in Fig. 3, the task weighting was calculated using the parameter weight values given by experts 1–5, the deadline error rate of the task set was 0.6667, 0.5000, 0.4833, 0.5833 and 0.5000 for the five experts, respectively. The lowest was only 0.4833. However, the deadline miss rate of the scheduling task set was 0.2833 after optimization through the FAHP. Therefore, FAHP reduced the deadline miss rate for task scheduling.

IV. SELECTING EXECUTION TASKS BASED ON HEAPSORT

A. HEAPSORT

After the comprehensive priority P_i' was determined, it was necessary to sort P_i' using a reasonable sorting algorithm in order to prepare for the selection of a task priority scheduling subset. In the process of sorting, there would be system overhead, which would affect the efficiency of task scheduling. The heap sort algorithm selected in this study satisfied the conditions of fewer comparisons, a lower time complexity and less system consumption. It played an important role in reducing the deadline miss rate.

1) THEORETICAL ANALYSIS

Heapsort is a sort algorithm based on heap data structure design. It was created by Robert W. Floyd and J. Williams in 1964. It first sets the top stack as the initial disordered sequence $(R_1, R_2, \cdots, R_n)$ and exchanges the top element R_1 with the last element R_n to obtain the new disordered area $(R_1, R_2, \cdots, R_{n-1})$ and the ordered area R_n, satisfying $R_{1,2\cdots n-1} \leq R_n$. Because the sorted new heap top R_1 can potentially violate the heap nature, the disordered area $(R_1, R_2, \cdots, R_{n-1})$ needs to be adjusted to the new heap. Then, earlier operation and exchange of $R_1$with $R_{n-1}$is repeated to get the disordered area $(R_1, R_2, \cdots, R_{n-2})$ and the ordered area (R_{n-1}, R_n), until the number of elements in the last ordered area is $n-1$. At this point, the task is complete. In the worst case, heapsort can achieve optimal results, it can easily achieve forward or reverse sorting. It also has similarities with direct sorting, but heapsort can preserve the results of a partial comparison in the form of a binary tree, which can reduce the number of direct sorting comparisons, the running time of programs and time complexity.

Concerning heapsort time complexity, the initial heap is built on the basis of a binary tree and new heaps are created repeatedly. The calculation of time complexity was cited in [23]. The total time complexity of a heapsort is $T(n)' = O(n\log_2 n) + O(n) + (n)$. In the worst case, the constant is ignored, and only the higher order items are considered, so the complexity of heapsort time is $O(n\log_2 n)$.

In this study, the smaller the system overhead occupied by the sorting algorithm in the process of prioritizing, the lower the time complexity of the sorting algorithm and the higher the superiority. Compared with the Shell sorting algorithm [24], the time complexity of the heapsort was lower, as in $O(n^{1.5}) > O(n\log_2 n)$, the system overhead was smaller and the system sorting time was shorter. Experimental simulations were carried out to verify the superiority of the heapsorting.

2) SIMULATION VERIFICATION

First, 100 random numbers with a numerical range of 0-100 were randomly generated, the sorting time of different algorithms was monitored by a sorting algorithm. Because time fluctuates by a small margin, we used 100 random numbers, sorted them 20 times, obtained the average of the results and then obtained a time comparison chart of the different ranking algorithms.

In Fig. 4, the bubble sorting and insertion sorting time curves are at the top, the sorting time for both curves was substantially over 1.5 s, which was greater than that of the other algorithms. The experimental data from the heapsort, the select sort, the quicksort and the shell sort algorithms were partially enlarged. As the figure shows, the sorting time for the bottom overlap sorting algorithm was less than 0.02 s, the shortest sorting time was the heapsort. Therefore, the heapsort was the preferred algorithm; it had a low time complexity and low system overhead needed to meet the needs of the study.

B. PRIORITY SCHEDULING SUBSET

After the tasks in the scheduling task set were sorted using heapsort, it was necessary to determine the subset of task priority scheduling. Tasks in the priority scheduling subset had to be processed before the deadline to reduce the deadline miss rate of the scheduling algorithm.

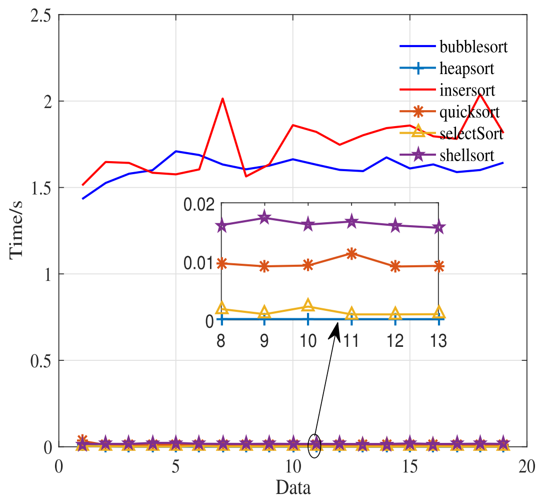

FIGURE 4. Sorting time of different algorithms.

In theory, when the priority scheduling subset is determined, all tasks in the subset satisfy the condition $U \leq 1$. However, in practice, in addition to system scheduling overhead, there is system scheduling overhead $\Delta t'$ in the case of priority scheduling tasks, which affects the deadline miss rate of a scheduling task set. Assuming that the total running time of the task was C, the EDF algorithm could be scheduled only when the priority scheduling subset satisfied the scheduling condition in the following equation; otherwise, the phenomenon of task overload would occur:

$$U = \sum_{i=1}^{n} \frac{C_i + \Delta t}{T_i} \leq \frac{T - \Delta t'}{T} = 1 - \frac{\Delta t'}{T}. \tag{11}$$

The heapsort algorithm was used to sort P_i' in descending order. Because the time complexity of the algorithm was low, the sorting time was short, the system overhead $\Delta t'$ of the algorithm was smaller than that of the Shell sorting algorithm. As (11) shows, the heapsort algorithm relaxed the range of the EDF algorithm execution. The smaller the heapsort algorithm $\Delta t'$, the larger the range $1 - \frac{\Delta t'}{T}$. Moreover, the smaller the load value U_i of each cycle task, the more tasks with priority scheduling subset Q that were added. All tasks in the priority scheduling subset had to satisfy $U \leq 1 - \frac{\Delta t'}{T}$. Then, the tasks in the priority scheduling subset were scheduled using the EDF scheduling algorithm. If there was free time, other tasks with a higher priority were processed.

V. PERFORMANCE ANALYSIS AND EXPERIMENTAL RESULTS

We used the experimental data in [20], as shown in Table 3, including task running time, task value, task energy consumption and other parameters. In this study, the weight of the parameters was obtained by using the FAHP, where $k_v = 0.1716$, $k_c = 0.6560$, and $k_e = 0.1724$. The task priority values were calculated via (3), as shown in Table 4, the task comprehensive priority values were calculated via (5), as shown in Table 5.

TABLE 3. Experimental task set.

Task	Period (ms)	Running time(ms)	Task value	Energy consumption (uJ)
1	20	6	3	10
2	20	10	8	12
3	20	12	5	15
4	40	8	6	18
5	40	10	10	15
6	40	20	4	20

TABLE 4. Task priorities P_i.

Task	1	2	3	4	5	6
P_i	9.3	7	5.5	7.3	7.1	3.6

TABLE 5. Task comprehensive priority (value density) P_i'.

Task	1	2	3	4	5	6
P_i	30.9	14	9.2	36.5	28.3	7.3

In order to verify the superiority of the improved algorithm, we conducted a comparative analysis of two aspects: task completion degree and task deadline miss rate.

A. TASK COMPLETION DEGREE

To study the scheduling performance of overload experiments, the workload values of 2.35, 2.8 and 3.25 under overload conditions were randomly selected. The task set was scheduled by the scheduling algorithm, and the performance of the task execution is shown in Table 6. We found that the order of the task completion from low to high was as follows: based on the Shell sort EDF algorithm < based on the heapsort EDF algorithm < improved Shell sort EDF algorithm < improved heapsort EDF algorithm. Compared with the first four algorithms, the improved heapsort EDF algorithm exhibited superiority in scheduling performance.

B. TASK DEADLINE MISS RATE

In order to verify that the improved algorithm could effectively reduce the task deadline miss rate, we compared the common priority with the comprehensive priority separately. The task deadline miss rate of the EDF scheduling algorithm, the Shell sort EDF scheduling algorithm and the heapsort EDF scheduling algorithm under common priority is shown in Fig. 5. The task deadline miss rate caused by the scheduling algorithm under integrated priority is shown in Fig. 6.

When the system was in an overload state, the deadline miss rate of the EDF scheduling algorithm increased significantly. It can be seen in Figs. 5 and 6 that the performance of the EDF scheduling algorithm based on the Shell sort algorithm under common priority was better than that of the EDF scheduling algorithm when the workload was less than 2.7, but the deadline miss rate was higher when the workload

TABLE 6. **Comparison of different scheduling algorithms.**

Workload	2.35		2.8		3.25	
Algorithm	Task Completion Degree	DMR	Task Completion Degree	DMR	Task Completion Degree	DMR
EDF algorithm	Task 1,Task 2,33.1% Task 3	0.6115	Task 1,Task 2	0.6667	Task 1,60% Task 2	0.7333
EDF algorithm based on Shell sort	Task 2,Task 5,48.16% Task 4	0.5864	Task 5,79.79% Task 2	0.7003	Task 5,63.13% Task 4	0.7281
EDF algorithm based on Heap sort	Task 2,Task 4,Task 5	0.5	Task 5,24.78% Task 2	0.6171	Task 5,Task 4,4.22% Task 6	0.6596
Improved Shell sort EDF algorithm	Task 1,Task 4,Task 5,13% Task 6	0.4783	Task 1,Task 2,59.59% Task 5	0.5673	Task 4,Task 5,36.03% Task 6	0.6066
An improved algorithm in this paper	Task 1,Task 2,Task 4,Task 5,42.15% Task 6	0.4297	Task 1,Task 2,Task 5,1.90% Task 6	0.4968	Task 4,Task 5,61.33% Task 6	0.5644

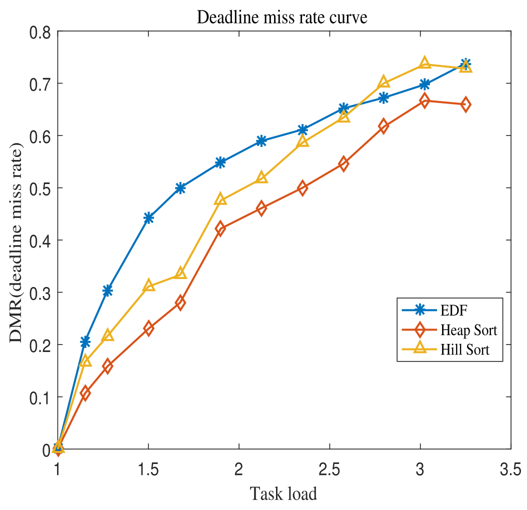

FIGURE 5. **Deadline miss rate under common priority.**

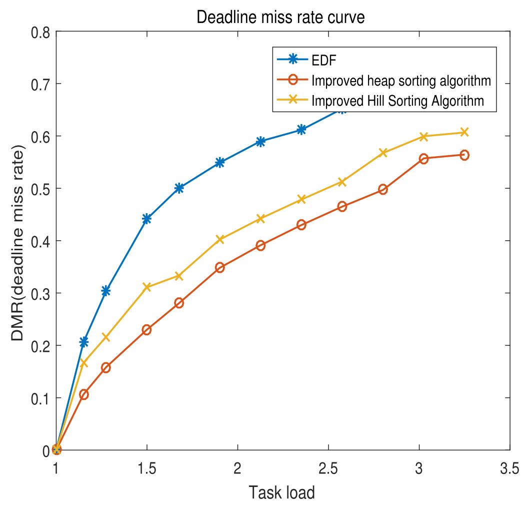

FIGURE 6. **Deadline miss rate under comprehensive priority.**

was higher than 2.7. After the Shell sort algorithm changed to the heapsort algorithm, the system overhead generated by the sorting algorithm was reduced, the deadline miss rate of

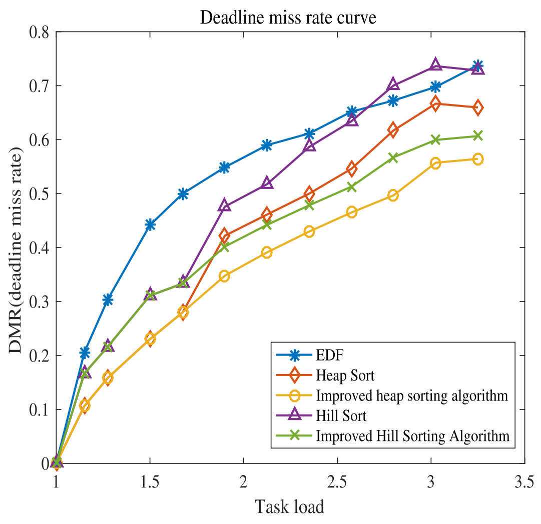

FIGURE 7. **Deadline miss rate curve.**

task scheduling was significantly reduced, which improved the scheduling performance of the system.

A comparative analysis of the composite priority and the common priority is shown in Fig. 7. When the task load was higher than 1.8, the deadline miss rate of the improved shell sort and improved heapsort algorithm was significantly lower than that of the previous ones. In the process of task scheduling, in order to obtain better system scheduling performance, it was necessary to select a subset of priority scheduling tasks more simply and efficiently, the task priority was not determined by a single indicator. The improved heapsort EDF scheduling algorithm was improved in the priority index so that the number of tasks in the task scheduling subset was increased, the tasks with higher priority were scheduled in time and the deadline miss rate was the lowest. In addition, the improved heapsort EDF scheduling algorithm added the system sorting overhead and the system scheduling overhead to the subset determining conditions, which made the scheduling more realistic and the CPU resources better utilized.

VI. CONCLUSION

In real-time systems, due to different task priorities, task scheduling is often overloaded, which leads to a decrease in the success rate of task scheduling and the waste of system resources. The improved method proposed in this study not only quickly selected the optimal scheduling subset from the task set but also prioritized the high-priority tasks in the system and improved the system resource utilization. The experimental results showed that the improved heapsort dynamic priority scheduling algorithm shortened the deadline miss rate of the task set by 0.1789, which improved the scheduling performance of the algorithm and effectively solved the problem of a low success rate in task scheduling and a waste of system resources. In future research, we will continue to study how to optimize the scheduling subset for better system scheduling performance.

REFERENCES

[1] X. Lin, Y. Wang, N. Chang, and M. Pedram, "Concurrent task scheduling and dynamic voltage and frequency scaling in a real-time embedded system with energy harvesting," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 11, pp. 1890–1902, Nov. 2016.

[2] H. E. Ghor and M. Chetto, "Overhead considerations in real-time energy harvesting systems," in *Proc. Int. Conf. Pervasive Embedded Comput. Commun. Syst. (PECCS)*, Feb. 2015, pp. 358–362.

[3] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[4] Y.-W. Zhang, "Energy-aware mixed partitioning scheduling in standby-sparing systems," *Comput. Standards Interfaces*, vol. 61, pp. 129–136, Jan. 2019.

[5] G. Li, C. Deng, J. Li, Q. Zhou, and W. Wei, "Deadline and period assignment for update transactions in co-scheduling environment," *IEEE Trans. Comput.*, vol. 66, no. 7, pp. 1119–1131, Jul. 2017.

[6] J. Lee, "Real-time uniprocessor scheduling with fewer preemptions," *Computing*, vol. 99, no. 12, pp. 1257–1270, 2017.

[7] N. Guan, "EDF in real-time calculus," in *Techniques for Building Timing-Predictable Embedded Systems*. Springer, 2016, pp. 209–225.

[8] G. Xie, G. Zeng, Z. Li, R. Li, and K. Li, "Adaptive dynamic scheduling on multifunctional mixed-criticality automotive cyber-physical systems," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 6676–6692, Aug. 2017.

[9] J.-L. Xia, W.-L. Wang, Z.-H. Cao, and Z.-B. Han, "HP-OMS: Overload management model for real-time systems," *Appl. Res. Comput.*, vol. 30, no. 6, pp. 1678–1681, 2013.

[10] J. H. Anderson, J. P. Erickson, U. C. Devi, and B. N. Casses, "Optimal semi-partitioned scheduling in soft real-time systems," *J. Signal Process. Syst.*, vol. 84, no. 1, pp. 3–23, 2016.

[11] C. Barbieru and F. Pop, "Soft real-time hadoop scheduler for big data processing in smart cities," in *Proc. IEEE 30th Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, Mar. 2016, pp. 863–870.

[12] W. L. Wang *et al.*, "Dynamic scheduling strategy PT-stds based on preemption threshold of soft real-time," *J. Chin. Comput. Syst.*, vol. 39, no. 5, pp. 986–990, 2018.

[13] C. Dong and Y. Chen, "Real-time scheduling algorithm of dynamic with fault-tolerant in heterogeneous distributed systems," *J. Syst. Simul.*, vol. 29, no. 5, pp. 1132–1140, May 2017.

[14] L. Palopoli, D. Fontanelli, L. Abeni, and B. V. Frias, "An analytical solution for probabilistic guarantees of reservation based soft real-time systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 3, pp. 640–653, Mar. 2016.

[15] H. He, G. Xu, S. Pang, and Z. Zhao, "AMTS: Adaptive multi-objective task scheduling strategy in cloud computing," *China Commun.*, vol. 13, no. 4, pp. 162–171, Apr. 2016.

[16] A. Biondi and Y. Sun, "On the ineffectiveness of 1/m-based interference bounds in the analysis of global EDF and FIFO scheduling," *Real-Time Syst.*, vol. 54, no. 3, pp. 515–536, 2018.

[17] J. Wu, "Energy-efficient concurrency control for dynamic-priority real-time tasks with abortable critical sections," *Comput. Informat.*, vol. 36, no. 4, pp. 765–792, 2017.

[18] L. Sang, Y. Lu, and L. Yu, "Optimization of EDF scheduling algorithm based on greedy policy," *Comput. Eng.*, vol. 41, no. 12, pp. 96–100, 2015.

[19] G.-L. Yu and M.-F. Zhang, "Optimization of edf scheduling algorithm based on bucket sort," *J. Lanzhou Univ. Technol.*, vol. 39, no. 4, pp. 110–113, 2013.

[20] R. Q. Wang, J. M. Zhao, and D. A. Li, "Dynamic priority scheduling algorithm based on shell's sort," *Video Eng.*, vol. 42, no. 5, pp. 57–59, 2018.

[21] A. Nikkhah, S. Firouzi, M. E. H. Assad, and S. Ghnimi, "Application of analytic hierarchy process to develop a weighting scheme for life cycle assessment of agricultural production," *Sci. Total Environ.*, vol. 665, pp. 538–545, May 2019.

[22] F. Ahmed and K. Kilic, "Fuzzy analytic hierarchy process: A performance analysis of various algorithms," *Fuzzy Sets Syst.*, vol. 362, pp. 110–128, May 2019.

[23] L. Wei, J. Bin, D. Ming, X. Teng, K. Jian, and S. Xiaohua, "Equalization optimization of modular multilevel converter based on heap sorting," *Electr. Meas. Instrum.*, vol. 55, no. 18, pp. 139–144, 2018.

[24] H. Zhipeng *et al.*, "A capacitor voltage balancing strategy adopting prime factorization method and shell sorting algorithm for modular multilevel converter," *Proc. CSEE*, vol. 35, no. 12, pp. 2980–2988, 2015.

SHANSHAN MENG received the B.Sc. degree from the Jiangsu University of Science and Technology, in 2017. She is currently a Graduate Student with the Shanghai University of Electric Power. Her main research interests include embedded systems and distributed control systems.

QIANG ZHU received the B.Sc. degree from the Shanghai University of Science and Technology, in 1993, and the M.Sc. and Ph.D. degrees from Shanghai University, in 1996 and 2011, respectively. He is currently with Shanghai Automation Instrumentation Co., Ltd., as the General Manager of DCS, the Minister of economic operations, and the Deputy Chief Engineer of the National Technology Center. His main research interests include power station automation, decentralized control systems, embedded system hardware and software, nuclear power digital systems, and functional safety systems.

FEI XIA received the B.Sc. degree from the Shenyang University of Technology, in 2000, the M.Sc. degree from the University of Poitiers, France, in 2003, and the Ph.D. degree from Tongji University, in 2017. He is currently an Associate Professor with the Shanghai University of Electric Power. His main research interests include machine vision, information security, and embedded systems in clean energy systems.