IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 29, 2021, accepted January 4, 2022, date of publication January 13, 2022, date of current version February 3, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3143037

A Schedulability Test for Sporadic Task DM Scheduling Based on Density Upper Bound

HONGBIAO LIU[1], XI CHEN[2], LEI QIAO[2], JINGKUN ZHANG[2], AND MENGFEI YANG[3]
[1]School of Computer Science and Technology, Xidian University, Xi'an 710070, China
[2]Beijing Institute of Control Engineering, Beijing 100190, China
[3]China Academy of Space Technology, Beijing 100190, China

Corresponding author: Xi Chen (954920575@qq.com)

This work was supported by the National Natural Science Foundation of China under Grant 61632005, Grant 62032004, and Grant 61802017.

ABSTRACT Due to low runtime overhead and simple implementation of DM (Deadline Monotonic) scheduling, it is widely used in real-time systems. Aiming at the schedulability test problem of the sporadic task DM scheduling under uniprocessor, a density upper bound of 0.693 is analyzed, which can determine the schedulability of a task set in linear time. The theoretical analysis process is carried out in three steps. Firstly, the case is considered where the task set contains only two tasks and the deadline ratio between the tasks is less than 2. Secondly, the case is considered where the task set contains multiple tasks and the deadline ratio between any two tasks is still less than 2. Finally, the case is considered where the task set contains multiple tasks and the deadline ratio between tasks is an arbitrary value. The experimental results show that the upper bound of density is higher than related methods and the run time overhead is much lower than that of other available exact schedulability tests. The time complexity is $O(1)$ when a task is dynamically added to a task set, while that of other methods increases rapidly along with the number of tasks. In addition, we combined this upper bound and other tests to further propose an exact schedulability test, which effectively reduces the running time overhead by 30.8% compared with the state-of-the-art schedulability test. Due to the high efficiency of our schedulability test, an online schedulability test can be implemented in open real-time systems.

INDEX TERMS Deadline monotonic, open real-time system, sporadic task, schedulability test, task density.

I. INTRODUCTION

Real-time systems [1] are widely used in safety-critical applications such as aerospace [2], automobiles [3], industrial control, robotics, communications, and medical electronics. It plays an important role in a system with strict timing requirements, fast response, and stability. The correctness of event processing results depends not only on the process but also on the time when the processing is completed. In a hard real-time system, if the system cannot meet the deadline requirements for event processing, the output results will become meaningless and even cause serious consequences.

Safety-critical unmanned systems such as spacecraft are typical real-time systems with limited computing resources. On the one hand, the system often contains a large number of data acquisition, calculation, control output and other strong real-time tasks, which need to meet the deadline requirements and ensure their real-time performance. On the other hand, to run stably in the complicated environment, the processor performance is relatively low and computing resources are limited. For example, computers in satellites may have strong cosmic radiation, which may lead to a SEU(Single Event Upset) phenomenon. We need to reduce the main frequency of the processor, and the memory chip is designed to be reinforced, so it is difficult to achieve high-performance computing.

In order to ensure the real-time performance of a system, task scheduling is the most critical technology. The industry currently generally adopts an off-line scheduling strategy, that is, using linear programming [4] or response time analysis [5] and other means to plan out the task execution sequence and form a fixed execution schedule. Tasks are started at a fixed time point, with a fixed period, and executed sequentially. Representative examples include table-based fixed-point scheduling strategy [6], 653 scheduling framework [2] and so on. With the expansion of spacecraft

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

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

functions, in face of changing the external environment and internal state without changing hardware resources, the system functions should be able to expand dynamically. Such as changes in user requirements or resistance to discovered security threats, we may reconstruct the system by software online updating [7]. At this time, the system is transformed into an open system and tasks may be dynamically added to it. However, the off-line scheduling technology is no longer applicable, and online scheduling methods are needed.

The DM(Deadline Monotonic) scheduling is proven to be the optimal fixed-priority scheduling algorithm for real-time tasks that share a critical instant [8]. The closer the task deadline, the higher the assigned priority, and vice versa. Due to its low runtime overhead and simple implementation, it is widely used in real-time systems [9], for example, aerospace and other strong real-time high safety-critical systems [10]. The dynamic addition of new tasks may cause time slice preemption and resource sharing to the original tasks in the system, which may cause tasks to miss the deadline and cause serious consequences. In order to ensure the safety and real-time performance of the spacecraft system with limited computing resources, schedulability tests with low running time overhead are required.

Therefore, our motivation can be divided into the following two aspects.

On the one hand, it comes from actual engineering needs. We need to perform online schedulability tests of task sets in resource-constrained computer systems such as spacecraft computers. However, existing schedulability tests are not applicable because of the high running time overhead.

On the other hand, for sporadic task DM scheduling, there are no research results on density upper bound analysis. If we can find a density upper bound, the schedulability of a task set can be obtained in linear time, which will greatly improve the analysis speed and make the system have better real-time performance. When the system dynamically adds a task, the time complexity is only $O(1)$, which enables the spacecraft system to perform an online schedulability analysis.

Therefore, we need to find a schedulability test with low running time overhead to implement online schedulability analysis for resource-constrained embedded real-time systems.

The main contributions and innovations of the research are summarized as follows:

1) We analyze that a density upper bound for the sporadic task is 0.693, which can determine the schedulability of a task set in linear time.
2) We combined this upper bound and other tests to further propose an exact schedulability test, which effectively reduces the running time overhead by 30.8% compared with the state-of-the-art schedulability test.

The first chapter of this article introduces the background, the second chapter introduces related work, the third chapter describes the mathematical model of the task, the fourth chapter carries on the theoretical analysis and proof of the upper bound of density, and the fifth chapter is the experiment and result analysis. The running time overhead and the schedulable ratio of this algorithm are compared with the other available schedulability tests. The last chapter is the conclusion of our work.

II. RELATED WORK

The time complexity of schedulability determination for a real-time task set is a classic theoretical problem, and it is listed as one of the five open topics in the real-time scheduling field [11]. In the real-time scheduling field, fixed priority and dynamic priority scheduling are widely studied.

As we know, the EDF(Earlier Deadline First) scheduling is the optimal dynamic priority scheduling method in a single-processor preemptive system. The closer the job deadline, the higher the assigned priority, and vice versa. For periodic tasks [12], the necessary and sufficient condition for the schedulability test of EDF scheduling is that the task set utilization rate is less than or equal to 1. However, for sporadic tasks [13], the time complexity of EDF scheduling proves to be an NP-hard problem.

In a fixed-priority scheduling system, each sporadic task is assigned a different priority, and all jobs of the task inherit this priority. Due to the advantages of low runtime overhead and simple implementation, fixed-priority scheduling is widely used in real-time operating systems, such as Ucos [14], FreeRTOS [15] and VxWorks [16]. Ekberg *et al.* reduced the fixed-priority scheduling problem to the dynamic priority EDF scheduling problem through polynomial time and proved that the schedulability determination of the fixed-priority preemptive scheduling for sporadic tasks is also an NP-hard problem [17].

Liu and Layland proposed the RMS (Rate-Monotonic Scheduling) algorithm for periodic tasks [12], which belongs to the fixed-priority preemptive scheduling. According to the period of the task (representing task request rate) to assign task priority, it is the optimal scheduling algorithm in fixed priority scheduling. A sufficient schedulability test of the hard real-time task set is given and the time complexity is only $O(n)$, which lays the foundation for the development of real-time scheduling theory. Later, Bini *et al.* [18] proposed using the hyperbolic bound to determine the schedulability of the task set scheduled by RMS. Leung Y T [8] first proposed the DM scheduling, which was proven to be the optimal fixed-priority scheduling algorithm for tasks that share a critical instant. The shorter the task's deadline, the higher the task's priority assigned. Audsley N C [19] investigated schedulability for mixtures of periodic and aperiodic tasks, proposing a sufficient schedulability test for DM scheduling. For the aperiodic task, Abdelzaber *et al.* gave a density upper bound of a task set which is 0.59 [20], which can be used for online schedulability tests in an open real-time operating system. In fact, the sporadic task model can be considered as a special case of the aperiodic task model, so this density upper bound can still be used as a schedulability test for sporadic task.

Regarding the schedulability determination of fixed-priority preemptive scheduling for task sets assigned with arbitrary priority, Bonifaci V *et al.* [21] discussed the special case of task systems with a harmonic period, and gave an exact schedulability test with polynomial time complexity. Chen *et al.* [22] proposed to determine the schedulability of the tasks by verifying the hyperbolic formula. The time complexity is only $O(n^2)$, but this is only a sufficient condition for the schedulability test. In addition, it does not support the schedulability test with release jitter, blocking time, etc.

All the above methods cannot exactly determine the schedulability of a task set. Another more general method is based on the RTA (Response Time Analysis) [5] [23], which can exactly determine the schedulability of the task set, and it is a sufficient and necessary condition for schedulability test. The RTA is given by Joseph, Pandya and Audsley, which has been widely studied in academia and applied in the industry. The RTA uses a fixed-point idea that combines the processor demands of all high-priority tasks. The iterative calculation starts from an initial value until the response time of the task no longer changes, so as to obtain the final response time of the task. By comparing the relationship between the response time and the deadline of the task, it can determine whether the task is schedulable. If the response time is greater than the deadline, the task cannot be scheduled, otherwise, it can be scheduled. This method has high time complexity. When time parameters are rational numbers, it is an NP-hard problem [24]. When time parameters are natural numbers, the time complexity is $O\left(n^2 * D_{max}\right)$, where n is the number of tasks, and D_{max} is the maximum deadline of the task set [25]. Further extensions of the RTA are researches including blocking [26] [27], release jitter [28], arbitrary deadline [29] [30], etc. In addition, the LP(linear programming) [4] can also be used to solve the problem of schedulability determination, which will set the objective optimization function according to the scheduling goal, such as the minimum task switching overhead, etc., and then set the corresponding constraint conditions according to various timing requirements. The execution schedule of each task is obtained according to the solution of each variable. Due to the large amounts of calculation and high time complexity of this method, a dedicated computing server is often used to solve the problem, and it is only suitable for offline scheduling systems.

In order to improve the efficiency of schedulability test, more researches later were to improve the test speed by finding a better initial value used in RTA. Lehoczky J *et al.* [31] proposed that RTA only needs to check the multiple integer time points, which can exactly determine the schedulability of the task set scheduled by the RMS. Hansson H *et al.* [32] [33] analyzed the lower bound of the response time and proposed an initial value

$$R_i^0 = R_i^{LB} = (C_i + B_i + \sum_{j<i} J_j U_j)/(1 - \sum_{j<i} U_j).$$

However, due to the existence of the sum term in this initial value, the test efficiency is not high. Since then, based on [34], Davis *et al.* [35] extended the initial value $D_i/2$ to include release jitter J_i and blocking time B_i, and obtained an improved initial value for exact schedulability test $R_i^0 = (D_i - J_i + C_i + B_i)/2$, and proved that it is the optimal initial value for RTA. The recent studies mainly focus on the schedulability analysis of other task models. They are self-suspended task model [36], restricted preemptive scheduling [37], DAG(Direct Acyclic Graph) task model [38],multi-core scheduling [39] and so on.

Although the schedulability test theory has been developed for many years and has made great progress, the above methods still have two problems. On the one hand, available linear time analysis algorithms lead to inefficient use of the processor. On the other hand, the exact schedulability test methods still have a high running time overhead, which results in a limited range of practical applications.

III. TASK MODEL

We consider the schedulability test of sporadic tasks DM scheduling under uniprocessor. The classic triplet model is shown in Fig. 1, $\tau_i = (T_i, C_i, D_i)$. C_i represents the worst-case execution time of the task τ_i. D_i represents the relative deadline of τ_i. One arrival of a task is called an instance (also called job) and the instance arrives at any time, but there is a minimum arrival time interval T_i between two instances. In the worst-case, it can be regarded as the period of τ_i. τ_i^j represents the jth instance of τ_i. τ_i^{j+1} represents the $(j+1)$th instance of τ_i. The difference between the arrival time of τ_i^j and the arrival time of τ_i^{j+1} is greater than or equal to T_i, where $j \in N$. According to the relationship between D_i and T_i, if $D_i = T_i$, then τ_i is called an Implicit Deadline task. If $D_i \leq T_i$, then τ_i is called a Constrained Deadline task. If D_i and T_i are not related then τ_i is called an Arbitrary Deadline task. The set of real-time tasks is denoted as $S_H = \{\tau_1, \tau_2, \ldots, \tau_n\}$, where n represents the task number of S_H. If we compare the aperiodic task model [20] with the sporadic task model, the former has a worse worst-case scenario due to the fact that each job of a task has no arrival moment, execution time and deadline constraints. In fact, the sporadic task model can be considered as a special case of the aperiodic task model.

π_i represents the priority of τ_i. The larger the π_i, the higher the priority of τ_i. The task's index is arranged in order of priority from high to low, and $\pi_j > \pi_i$ if and only if $j < i$.

The response time R_i of τ_i represents the difference between the task's finish time and arrival time. Let Δ denote the sum of the processor demand of all high-priority tasks $\{\tau_j\}$ in the interval $[0, R_i]$, $j < i$, so $R_i = C_i + \Delta$.

τ_i is called schedulable, which means that each instance of τ_i can finish before the deadline, that is $R_i \leq D_i$.

The real-time task set S_H is called schedulable, which means that all tasks in S_H are schedulable, that is, $\forall \tau_i \in S_H$, τ_i *is schedulable*.

The schedulability analysis refers to obtaining information such as the response time and schedulability of the task through mathematical derivation.

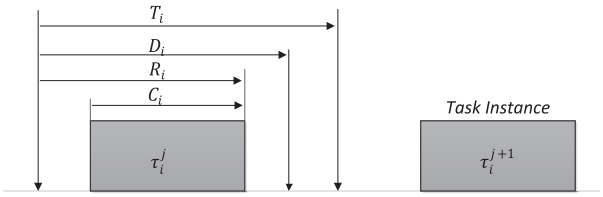

FIGURE 1. Sporadic task model.

The schedulability test refers to the process of using the schedulability analysis method to determine whether the task is schedulable. It is also called schedulability determination.

The sufficient schedulability test means that if a task is determined to be schedulable, the task must be schedulable, but if it is determined to be unschedulable, the task is not necessarily unschedulable.

The exact schedulability test means that if a task is determined to be schedulable, the task must be schedulable, otherwise, the task must not be schedulable. It is also commonly referred to as a sufficient and necessary condition for schedulability determination.

The utilization of τ_i is expressed as $u_i = C_i/T_i$, and the utilization of S_H is expressed as $U = \sum u_i, \tau_i \in S_H$.

The density of τ_i is expressed as $dt_i = C_i/D_i$, and the density of S_H is expressed as $DT = \sum dt_i, \tau_i \in S_H$.

The density upper bound analysis method means that the schedulability of a task set is determined by using the sum of the density of all tasks. They do not need to determine the schedulability of each task to get the schedulability of the task set. Therefore, such methods usually have very low run time overhead.

We consider the Constrained Deadline tasks, $\tau_i = (T_i, C_i, D_i)$, which satisfies $0 < C_i \leq D_i \leq T_i, T_i, C_i, D_i \in R$. Tasks are independent of each other, with no shared resource access.

IV. DENSITY UPPER BOUND ANALYSIS

Real-time task set $S_H = \{(T_1, C_1, D_1), (T_2, C_2, D_2), \ldots, (T_n, C_n, D_n)\}, D_1 \leq D_2 \leq \ldots \leq D_n, \pi_1 > \pi_2 > \ldots > \pi_n$. The density upper bound of the task set is analyzed in three steps. First, the case is considered where S_H contains only two tasks, and the deadline ratio is less than 2, that is $D_2/D_1 < 2$. Second, the case is considered where S_H contains multiple tasks, but the deadline ratio of any two tasks is less than 2, that is, $D_i/D_j < 2, \tau_i, \tau_j \in S_H$. Finally, the case is considered where S_H contains multiple tasks and the deadline ratio between tasks is an arbitrary value. We use a special-to-general proof approach. For each case, we analyze the minimum density of the task set under the condition that the processor is fully utilized. The minimum density may be obtained by taking the derivative and is the upper bound of the density in this case. Then we take the minimum value of density upper bound in these three cases as the density upper bound in any case of the task set. Our proof strategy is similar to that of Liu and Layland [12]. As our goal is to find the upper bound on the density, we have to find the worst-case and obtain the minimum density by taking the derivative.

A. STEP 1

Real-time task set $S_H = \{(T_1, C_1, D_1), (T_2, C_2, D_2)\}, D_1 \leq D_2, D_2/D_1 < 2, \pi_1 > \pi_2$, analyze the schedulability of S_H.

Analysis. According to the relationship among D_2, T_1 and C_1, there are 3 cases:

Case 1. $D_1 \leq D_2 < T_1$. In order to obtain the minimum density of S_H when S_H makes full use of the processor, let

$$C_2 = D_2 - C_1 \tag{1}$$

As shown in Fig. 2.

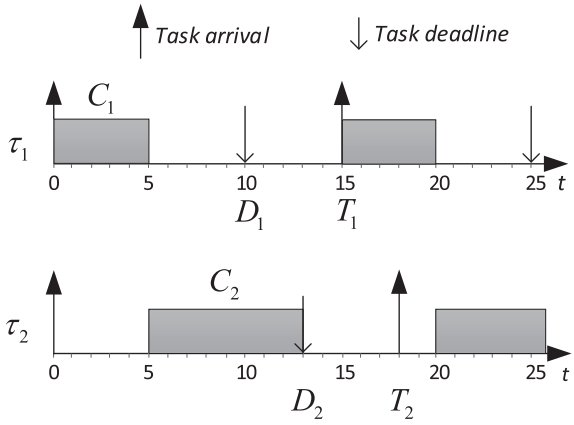

FIGURE 2. Case 1.

Combining equation (1), the density of S_H is $DT = \sum dt_i = C_1/D_1 + C_2/D_2 = C_1/D_1 + (D_2 - C_1)/D_2$. Obviously, when D_1 is larger, DT will be smaller. Combining the condition $D_1 \leq D_2 < T_1$, we can know when $D_1 = D_2$, DT is the smallest, and there is

$$DT_{min} = \frac{C_1}{D_1} + \frac{D_2 - C_1}{D_2} = \frac{C_1}{D_2} + \frac{D_2 - C_1}{D_2} = 1 \tag{2}$$

Case 2. $T_1 \leq D_2 < T_1 + C_1$. In order to obtain the minimum density of S_H when τ_1 and τ_2 makes full use of the processor, let

$$C_2 = T_1 - C_1 \tag{3}$$

As shown in Fig. 3.

Combining equation (3), the density of S_H is $DT = \sum dt_i = C_1/D_1 + C_2/D_2 = C_1/D_1 + (T_1 - C_1)/D_2$. Obviously, when D_1 and D_2 is larger, DT will be smaller. Combining the conditions $D_1 \leq T_1$ and $T_1 \leq D_2 < T_1 + C_1$, there is

$$D_1 = T_1\ ,\ D_2 = T_1 + C_1 \tag{4}$$

Therefore, DT is the smallest, that is

$$DT_{min} = \frac{C_1}{D_1} + \frac{T_1 - C_1}{D_2} = \frac{C_1}{T_1} + \frac{T_1 - C_1}{T_1 + C_1} \tag{5}$$

Case 3. $T_1 + C_1 \leq D_2 < 2D_1$. In order to obtain the minimum density of S_H when τ_1 and τ_2 make full use of the processor, let

$$C_2 = D_2 - 2C_1 \tag{6}$$

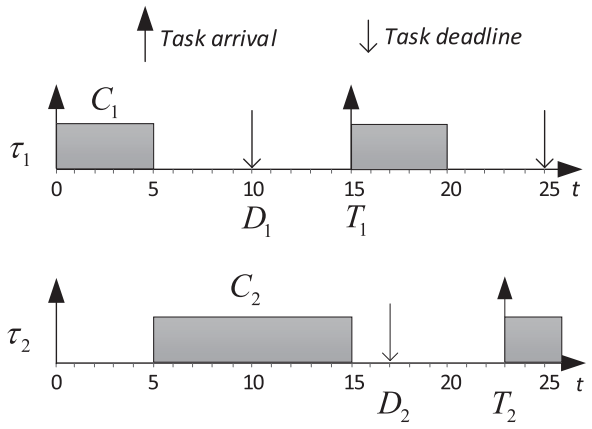

FIGURE 3. Case 2.

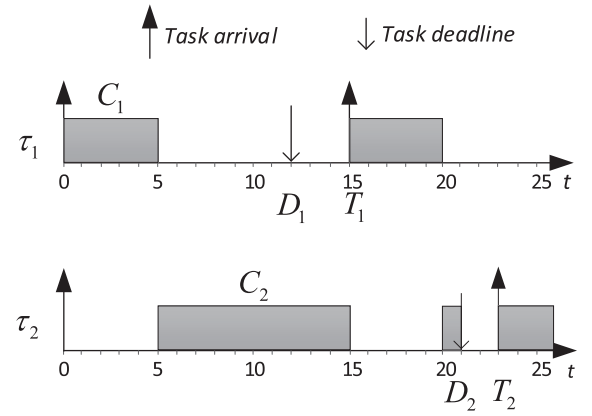

FIGURE 4. Case 3.

As shown in Fig. 4.

Combining equation (6), the density of S_H is $DT = \sum dt_i = C_1/D_1 + C_2/D_2 = C_1/D_1 + (D_2 - 2C_1)/D_2 = C_1/D_1 + 1 - 2C_1/D_2$.

Obviously, when D_1 is larger and D_2 is smaller, DT will be smaller. Combining the conditions $D_1 \leq T_1$ and $T_1 + C_1 \leq D_2 < 2D_1$, there is

$$D_1 = T_1, D_2 = T_1 + C_1 \tag{7}$$

Therefore, DT is the smallest, that is $DT_{min} = C_1/D_1 + 1 - 2C_1/D_2 = C_1/D_1 + 1 - 2C_1/(T_1 + C_1) = C_1/T_1 + (T_1 - C_1)/(T_1 + C_1)$.

Same as equation (5), we can combine Case 1,2 and 3 to derive

$$DT_{min} = min\left\{\frac{C_1}{T_1} + \frac{T_1 - C_1}{T_1 + C_1}, 1\right\} \tag{8}$$

As $C_1 \leq T_1$, there is $C_1/T_1 + (T_1 - C_1)/(T_1 + C_1) = 1 + C_1(C_1 - T_1)/[T_1(T_1 + C_1)] \leq 1$. Combining equation (8), we can get

$$DT_{min} = \frac{C_1}{T_1} + \frac{T_1 - C_1}{T_1 + C_1} \tag{9}$$

$\partial DT_{min}/\partial C_1 = 1/T_1 - 2T_1/(T_1 + C_1)^2 = 0$, the solution is

$$C_1 = T_1\left(\sqrt{2} - 1\right) \tag{10}$$

Substituting equation (10) into equation (9), there is

$$DT_{min} = \frac{T_1\left(\sqrt{2} - 1\right)}{T_1} + \frac{T_1 - T_1\left(\sqrt{2} - 1\right)}{T_1 + T_1\left(\sqrt{2} - 1\right)} = 2\left(\sqrt{2} - 1\right) \tag{11}$$

Therefore, for the real-time task set $S_H = \{(T_1, C_1, D_1), (T_2, C_2, D_2)\}, D_1 \leq D_2, D_2/D_1 < 2, \pi_1 > \pi_2$, if $DT = \sum dt_i = C_1/D_1 + C_2/D_2 \leq 2\left(\sqrt{2} - 1\right)$, then S_H can be scheduled.

Example 1: Real-time task set $S_H = \{(T_1, C_1, D_1), (T_2, C_2, D_2)\}, D_1 \leq D_2, D_2/D_1 < 2, \pi_1 > \pi_2$, from the analysis in the first step, we can know that if $C_1 = T_1\left(\sqrt{2} - 1\right), D_1 = T_1, D_2 = T_1 + C_1 = \sqrt{2}T_1$, then τ_1 and τ_2 will make full use of the processor and the density of S_H takes the minimum value $DT = C_1/D_1 + C_2/D_2 = 2\left(\sqrt{2} - 1\right)$, as shown in Fig. 5.

B. STEP 2

Real-time task set $S_H = \{(T_1, C_1, D_1), (T_2, C_2, D_2), \ldots, (T_n, C_n, D_n)\}, D_1 \leq D_2 \leq \ldots \leq D_n, D_i/D_j < 2$, where $j < i, \pi_j > \pi_i, \tau_i, \tau_j \in S_H$, analyze the schedulability of S_H.

Analysis. In the interval $[0, t]$, the preemption time caused by τ_j to low-priority tasks is $P_j(t)$,

$$P_j(t) = \frac{t}{T_j}C_j + \Delta_j\left(t, C_j, T_j\right) \tag{12}$$

From the fixed-priority preemptive scheduling, we can know that $\Delta_j\left(t, C_j, T_j\right)$ is a certain function. For t and C_j, it is a non-decreasing function, that is $\Delta_j\left(t', C_j', T_j\right) \geq \Delta_j\left(t, C_j, T_j\right), t' > t, C_j' > C_j$. For T_j, it is a non-increasing function, that is $\Delta_j\left(t, C_j, T_j'\right) \geq \Delta_j\left(t, C_j, T_j\right), T_j' > T_j$.

Combining equation (1), under the condition of making full use of the processor, it must satisfy

$$D_n = \sum_{j=1}^{n} P_j(D_n) = \sum_{j=1}^{n}\left\{\frac{D_n}{T_j}C_j + \Delta_j\left(D_n, C_j, T_j\right)\right\} \tag{13}$$

Obviously, from equation (13), D_n is a constant. We can treat C_k as a function with T_k as its argument. if T_k decreases, C_k will not increase, where $1 \leq k \leq n$, that is

$$C_k\left(T_k'\right) \leq C_k\left(T_k\right), \quad T_k{}' < T_k \tag{14}$$

The density of S_H is $DT = \sum_{k=1}^{n} d_k = C_k/D_k$. Obviously, when C_k is smaller, DT will be smaller. Combining equation (14), we can know that $\forall\tau_k \in S_H, T_k = D_k$, DT takes the minimum value. $\forall\tau_k \in S_H, \tau_k = (D_k, C_k, D_k)$ is the Implicit Deadline task model and it can be regarded as a classic periodic task model. Therefore, we only need to consider the case where T_k is equal to D_k.

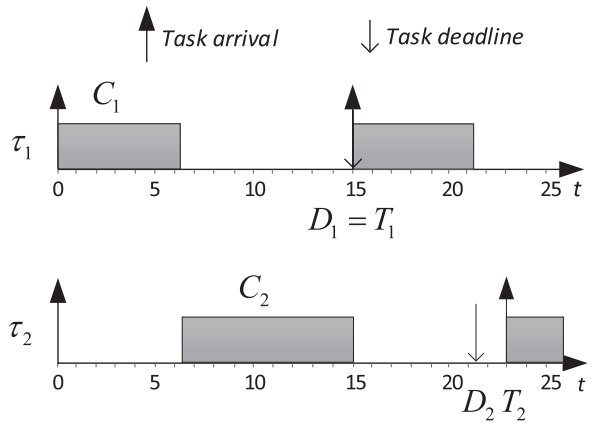

FIGURE 5. The minimum density of two tasks.

According to the enlightenment of the analysis in the first step, $C_1, C_2 \ldots C_n$ can be adjusted to make full use of the processor and make the density of S_H be the minimum value. It can be described as follows.

If $C_1 = D_2 - D_1, C_2 = D_3 - D_2, \ldots, C_{n-1} = D_n - D_{n-1}, C_n = D_n - 2(C_1 + C_1 + \ldots C_{n-1})$, then $DT = \sum_{\tau_i \in S_H} dt_i$ can be minimized.

Analysis. Assuming that $C_1 = D_2 - D_1$ is not true, one of the following two cases must be true.

Case 1: If $C_1 = D_2 - D_1 + \Delta$, $\Delta > 0$ holds, in order to fully utilize the processor, there is $C_2 = D_3 - D_2 - \Delta$, $C_3 = C_3, \ldots, C_n = C_n$.

Assuming $C_1' = D_2 - D_1, C_2' = D_3 - D_2, C_3' = C_3, \ldots, C_n' = C_n$, there is $DT - DT' = C_1/D_1 + C_2/D_2 - (C_1 - \Delta)/D_1 - (C_2 + \Delta)/D_2 = \Delta(1/D_1 - 1/D_2)$.

According to the condition $D_1 \le D_2$, $DT - DT' \ge 0$, we can get that when $C_1 = D_2 - D_1$, $DT = \sum_{\tau_i \in S_H} dt_i$ will take the minimum value.

Case 2: If $C_1 = D_2 - D_1 - \Delta$, $\Delta > 0$ holds, in order to fully utilize the processor, there is $C_2 = D_3 - D_2 + 2\Delta$, $C_3 = C_3, \ldots, C_n = C_n$.

Assuming $C_1' = D_2 - D_1, C_2' = D_3 - D_2, C_3' = C_3, \ldots, C_n' = C_n$, there is $DT - DT' = C_1/D_1 + C_2/D_2 - (C_1 + \Delta)/D_1 - (C_2 - 2\Delta)/D_2 = \Delta(2/D_2 - 1/D_1)$.

According to the condition $D_1 \le 2D_2$, so $DT - DT' \ge 0$, we can get that when $C_1 = D_2 - D_1$, $DT = \sum_{\tau_i \in S_H} dt_i$ will take the minimum value.

It can be derived in the same way that

$$\begin{aligned} &C_1 = D_2 - D_1 \quad , C_2 = D_3 - D_2, \ldots, \\ &C_{n-1} = D_n - D_{n-1}, \quad C_n = D_n - 2(C_1 + C_1 + \cdots + C_{n-1}) \end{aligned} \tag{15}$$

will make full use of the processor and $DT = \sum_{\tau_i \in S_H} dt_i$ be minimized. Therefore, there is $DT_{min} = C_1/D_1 + C_2/D_2 + \ldots + C_{n-1}/D_{n-1} + [D_n - 2(C_1 + C_2 + \ldots C_{n-1})]/D_n$. Combining equation (15), there is

$$DT_{min} = \frac{D_2}{D_1} + \frac{D_3}{D_2} + \cdots + \frac{D_n}{D_{n-1}} + \frac{2D_1}{D_n} - n \tag{16}$$

In order to find the minimum value of DT_{min}, we calculate the partial derivative of each variable.

$$\frac{\partial DT_{min}}{\partial D_1} = \frac{-D_2}{{D_1}^2} + \frac{2}{D_n} = 0 \Rightarrow 2{D_1}^2 = D_2 D_n \tag{17}$$

$$\frac{\partial DT_{min}}{\partial D_2} = \frac{1}{D_1} + \frac{-D_3}{{D_2}^2} = 0 \Rightarrow {D_2}^2 = D_1 D_3 \tag{18}$$

$$\frac{\partial DT_{min}}{\partial D_3} = \frac{1}{D_2} + \frac{-D_4}{{D_3}^2} = 0 \Rightarrow {D_{32}}^2 = D_2 D_4 \tag{19}$$

$$\frac{\partial DT_{min}}{\partial D_{n-1}} = \frac{1}{D_{n-2}} + \frac{-D_n}{{D_{n-2}}^2} = 0 \Rightarrow {D_{n-1}}^2 = D_{n-2} D_n \tag{20}$$

$$\frac{\partial DT_{min}}{\partial D_n} = \frac{1}{D_{n-1}} + \frac{-2D_1}{{D_n}^2} = 0 \Rightarrow {D_n}^2 = 2D_1 D_{n-1} \tag{21}$$

From Equation (17) to (20), we can see that $D_1, D_2, \ldots, D_n$ are geometric series. Combining equation (21). The common ratio q is solved.

$$(D_1 q^{n-1})^2 = 2D_1 D_1 q^{n-2} \Rightarrow q = \sqrt[n]{2} \tag{22}$$

Substituting equation (22) into equation (16), we can get $DT_{min} = (n-1)q + 2D_1/[D_1 q^{n-1}] - n = n\left(\sqrt[n]{2} - 1\right)$.

Therefore, for the real-time task set $S_H = \{(T_1, C_1, D_1), (T_2, C_2, D_2), \ldots, (T_n, C_n, D_n)\}$, $D_1 \le D_2 \le \ldots \le D_n$, $D_i/D_j < 2, j < i, \pi_j > \pi_i, \tau_i, \tau_j \in S_H$, if $DT = \sum dt_i = C_1/D_1 + C_2/D_2 + \ldots + C_n/D_n \le n\left(\sqrt[n]{2} - 1\right)$, then S_H can be scheduled.

C. STEP 3

Real-time task set $S_H = \{(T_1, C_1, D_1), (T_2, C_2, D_2), \ldots, (T_n, C_n, D_n)\}$, $D_1 \le D_2 \le \ldots \le D_n$, where $j < i, \pi_j > \pi_i, \tau_i, \tau_j \in S_H$, analyze the schedulability of S_H.

Analysis. If $\exists \tau_i, D_n/D_i \ge 2$, there is

$$D_n = \frac{D_n}{D_i} D_i + p, \quad p \ge 0, \ m = \frac{D_n}{D_i} \ge 2 \tag{23}$$

Replacing τ_i with τ_i', where

$$\tau_i' = \left(T_i', C_i', D_i'\right), C_i' = C_i, D_i' = mD_i, T_i' = D_i' \tag{24}$$

In order to make full use of the processor, C_n needs to increase $(m-1)C_i$ at most. Therefore, there is $DT' = DT - C_i/D_i - C_n/D_n + C_i'/D_i' + [C_n + (m-1)C_i]/D_n$.

Substituting equation (24), we can get $DT' = DT + (m-1)C_i[(1/(mD_i + p) - 1/(mD_i)]$.

Combining equation (23), we can see that $DT' - DT \le 0$, so $DT' \le DT$.

Therefore, the density of task set will be minimized if and only if $\forall \tau_i, \tau_j, D_i/D_j < 2$. The schedulability analysis result is the same as Step 2.

Summarizing the above three steps, we can derive the following theorem.

Theorem 1: The real-time task set $S_H = \{(T_1, C_1, D_1), (T_2, C_2, D_2), \ldots, (T_n, C_n, D_n)\}$, $D_1 \le D_2 \le \ldots \le D_n$,

$\pi_1 > \pi_2 > \ldots > \pi_n$, if

$$DT = \sum dt_i = \frac{C_1}{D_1} + \frac{C_2}{D_2} + \ldots + \frac{C_n}{D_n} \leq n\left(\sqrt[n]{2} - 1\right)$$

Then S_H can be scheduled.

When $n \to \infty, DT \approx 0.693$. Hence, the density upper bound of the sporadic task DM scheduling is 0.693. This result shows that the upper bound on the density of the sporadic task and the upper bound on the utilization of the periodicic task [12] are consistent.

Theorem 1 is a schedulability test for DM scheduling and it is called DM improve(S_H).

Algorithm 1 Bool DM Improve(S_H)

Input: Real-time task set S_H
Output: The schedulability of S_H

```
DT = 0;
for τ_i : S_H do
    DT = DT + C_i/D_i;
end for
if DT > n(∛2 − 1) then
    return false;
else
    return true;
end if
```

Example 2: Real-time task set $S_H = \{(T_1, C_1, D_1), (T_2, C_2, D_2), \ldots, (T_n, C_n, D_n)\}$, $D_1 \leq D_2 \leq \cdots \leq D_n, \pi_1 > \pi_2 > \ldots \pi_n$, from equation (15) and (22), there is

$$\frac{D_{i+1}}{D_i} = q = \sqrt[n]{2},\ T_i = D_i, C_i = D_i\left(\sqrt[n]{2} - 1\right),$$
$$C_n = D_n - 2\sum_{i=1}^{n-1} C_i$$

The density upper bound of S_H can take the minimum value $DT_{min} = n\left(\sqrt[n]{2} - 1\right)$, as shown in Fig. 6.

V. EXPERIMENT AND RESULT ANALYSIS

In the schedulability test of the real-time task set, the performance of the improved DM(called DM improve in the experiment) proposed in this paper is compared with the available schedulability tests.

As the sporadic task model can be considered as a special case of the aperiodic task model, the density upper bound in [20] can still be used as a schedulability test for sporadic tasks and is referred to as the original DM in this paper. Schedulability tests compared in this paper include the original DM (called DM original in the experiment) [20], the original RTA (called RTA original in the experiment) [5], the improved RTA (called RTA improve in the experiment, which is the state-of-the-art schedulability test) [35] and Hyperbolic algorithm [22]. 9 sets of comparative experiments will be conducted. Experiment 1-3 compare the schedulable ratio r, and Experiment 4-8 compare the run time overhead t. In addition, another exact schedulability test is proposed and is compared in Experiment 9. Computers are discrete systems and the parameters T_i, C_i, D_i of a task are randomly generated as integers, which are in accord with actual engineering applications.

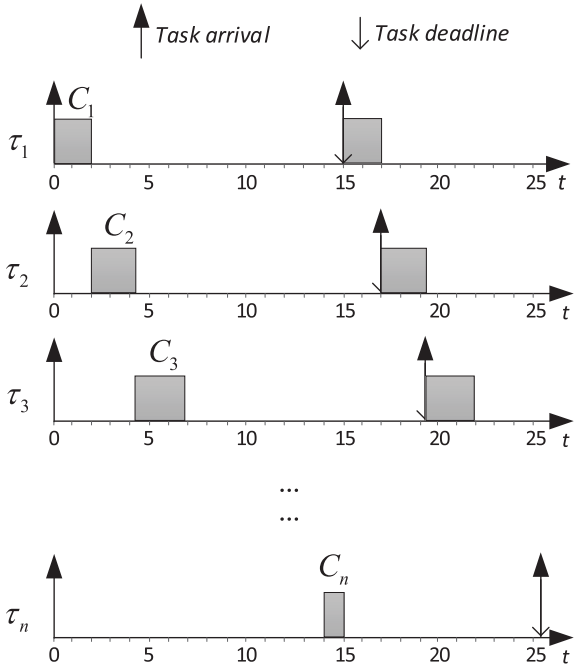

FIGURE 6. The minimum density of multiple tasks.

TABLE 1. Experimental system parameters.

Programming platform: PC	Operating platform: space station computer
CPU: Intel i5-3470	CPU: BM3803 single core
Main frequency: 3.2GHz	Main frequency: 80MHz
RAM: 8GB	RAM: SDRAM 4MB
Operating system: Win7	Real-time operating system: SpaceOS 2.0

In order to meet the actual applications, we choose the aerospace computer as the experimental platform. The China space station computer is a typical embedded real-time system. It uses the Chinese-produced 32-bit radiation-resistant single-core processor BM3803 based on the SPARC architecture [40]. It runs the SpaceOS real-time operating system. The SpaceOS is developed by the Beijing Institute of Control Engineering, which has high reliability and has been widely used in many major aerospace projects in China. The experimental platform parameters are shown in TABLE 1.

A. REGARDING THE SCHEDULABLE RATIO, THE DEFAULT EXPERIMENTAL PARAMETERS ARE AS FOLLOWS

- Number of task sets: Generate random task sets. The task set density DT is in the range $(0.1, 1]$, and the

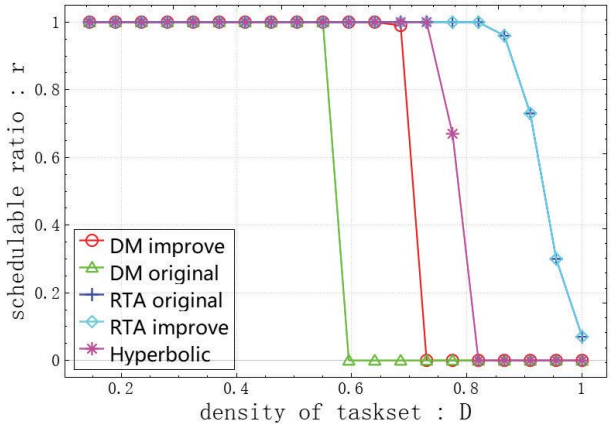

FIGURE 7. The schedulable ratio comparison when DT/U = 1.1.

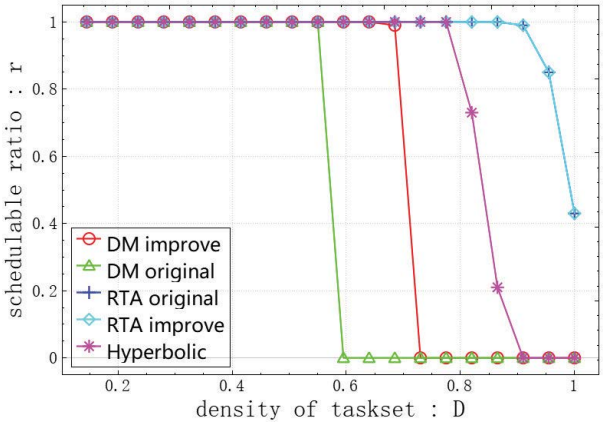

FIGURE 8. The schedulable ratio comparison when $DT/U = 1.2$.

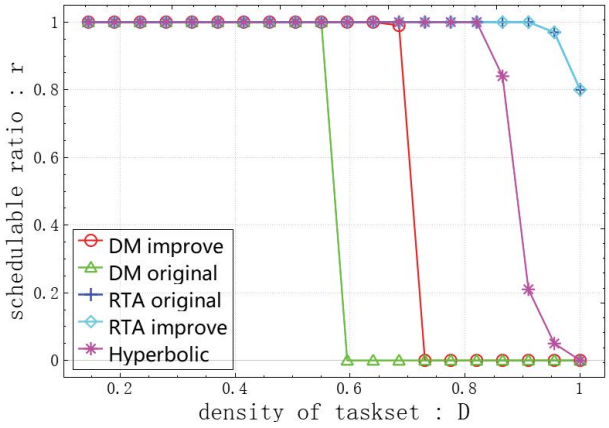

FIGURE 9. The schedulable ratio comparison when $DT/U = 1.5$.

ratio of task set density to utilization is DT/U. The step size is 0.045, and 100 task sets are generated for each step. Counting the proportion of the number of task sets determined to be schedulable as the corresponding schedulable ratio under the current density, so a total of 2000 task sets will be generated;

- The number of tasks: $n = 20$, the density dt_i of task is generated by UUnifast algorithm [41];
- Task deadline D_i: The deadline of the task is uniformly distributed, and the range is [1, 20000];
- Worst-case execution time C_i: The density of each task is multiplied by the deadline D_i to obtain the worst-case execution time C_i of the task. If C_i is less than 0, the task deadline is regenerated until C_i is greater than 0;

Experiment 1: The ratio of task set density to utilization is $DT/U = 1.1$, and other parameters are kept as default. The task set density DT is different, and the comparison of the schedulable ratio r is tested.

Experiment 2: The ratio of task set density to utilization is $DT/U = 1.2$, and other parameters are kept as default. The task set density DT is different, and the comparison of the schedulable ratio r is tested.

Experiment 3: The ratio of task set density to utilization is $DT/U = 1.5$, and other parameters are kept as default. The task set density DT is different, and the comparison of the schedulable ratio r is tested.

Experimental results: When the task set density is low, the schedulable ratio of all schedulability tests is 1. When the density is greater than 0.59, the schedulable ratio of the original DM quickly drops to 0. When the density is greater than 0.69, the schedulable ratio of the improved DM quickly drops to 0. Therefore, under the same schedulable ratio, the density upper bound of improved DM is 0.1 higher than the original DM. As the ratio DT/U increases, the starting point for changes in the schedulable ratio of original RTA, improved RTA, and Hyperbolic increases, as shown in Fig. 7, 8, and 9. We can see that the original RTA and improved RTA have the highest schedulable ratio.

Result analysis: Original DM and improved DM are density upper bound analysis methods. According to the calculation formula, the density upper bound of original DM is about 0.59, and that of improved DM is about 0.69. The drop in schedulable ratio has nothing to do with the ratio DT/U. The density upper bound proposed in [20] is for the aperiodic task model, but it also applies to the sporadic task model we considered. The density upper bound we derived is a bit higher than paper [20]. Thus, for the sporadic task, the schedulable ratio of improved DM is higher. Hyperbolic is only a sufficient condition for the schedulability determination. Original RTA and improved RTA are based on response time analysis and they are exact schedulability tests, so the schedulable ratio of original RTA and improved RTA are higher than that of Hyperbolic. As the ratio DT/U increases, the frequency of task requesting is relatively reduced, and the amount of preemption to low-priority tasks by high-priority tasks decreases, thereby increasing the schedulable ratio.

B. REGARDING THE RUNNING TIME OVERHEAD, THE DEFAULT EXPERIMENTAL PARAMETERS ARE AS FOLLOWS

- Number of task: A task is generated randomly and added dynamically to the task set each time. Therefore, the task number n of the task set gradually increases, and the range is [1, 30]. In order to prevent the interference of

extreme values when a task is randomly generated, the experiment is repeated 20 times for each step, and the average run time overhead of these 20 experiments is taken as the result of the current step;

- The task set density DT: The task set density DT gradually increases with the dynamic addition of tasks, and the maximum is 0.5. Each task density dt_i is generated by the UUnifast algorithm [41], and the ratio of the task set density to the utilization is DT/U;
- Task deadline D_i: The deadline of the task is uniformly distributed, and the range is $[1, 20000]$;
- Worst-case execution time C_i: The density of each task is multiplied by the deadline D_i to obtain the worst-case execution time C_i. If C_i is less than 0, the task deadline is regenerated until C_i is greater than 0;

Experiment 4: The ratio of task set density to utilization $DT/U = 1.1$, other parameters remain the default. The task set dynamically adds a task. The task number n of the task set is different, and the comparison of the average run time overhead t is tested.

Experiment 5: The ratio of task set density to utilization $DT/U = 1.2$, other parameters remain the default. The task set dynamically adds a task. The task number n of the task set is different, and the comparison of the average run time overhead t is tested.

Experiment 6: The ratio of task set density to utilization $DT/U = 1.5$, other parameters remain the default. The task set dynamically adds a task. The task number n of the task set is different, and the comparison of the average run time overhead t is tested.

Experimental results: As the number of tasks increases, the running time overhead of original DM and improved DM remain unchanged, while the running time overhead of original RTA, improved RTA and Hyperbolic increases rapidly. As shown in Fig. 10, 11, and 12, as DT/U increases, the running time overhead of original RTA, improved RTA and Hyperbolic is relatively reduced. The running time of improved DM and original DM is the same, only 0.005ms, which is much lower than that of Hyperbolic, original RTA and improved RTA.

Result analysis: Original DM and improved DM are density upper bound analysis methods, which can incrementally accumulate current task density. Therefore, the running time overhead has nothing to do with the number of tasks and the ratio DT/U. The density upper bound of original DM is a constant $1/(1+\sqrt{1/2})$, and the density upper bound of improved DM is $n(\sqrt[n]{2}-1)$, which can be calculated in advance through an array, so it has the same constant running time overhead as original DM and further improve the speed of schedulability test. The original RTA, improved RTA and Hyperbolic will determine the schedulability of the newly added task and all lower priority tasks. When the task parameters are integers, the running time overhead increases quadratically with the increase of the number of tasks [25]. As the ratio of task set density to utilization DT/U increases, the task set utilization decreases

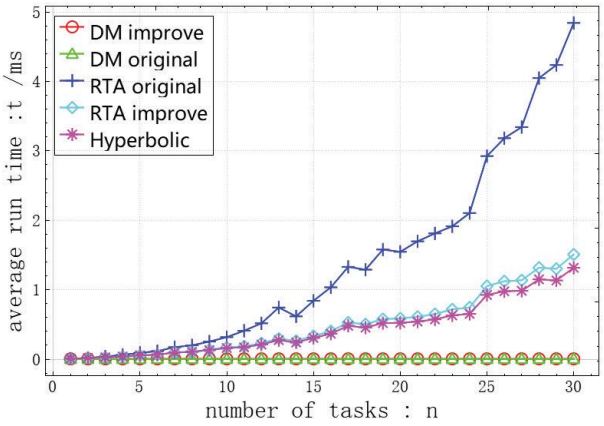

FIGURE 10. The run time overhead comparison when $DT/U = 1.1$.

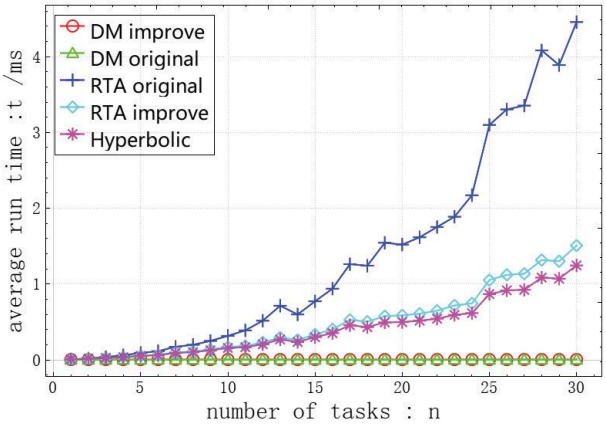

FIGURE 11. The run time overhead comparison when $DT/U = 1.2$.

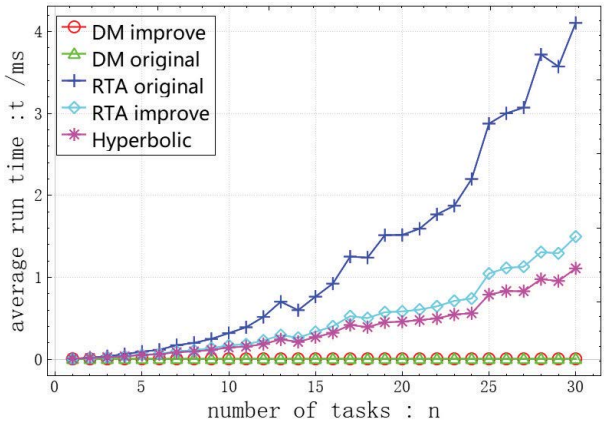

FIGURE 12. The run time overhead comparison when $DT/U = 1.5$.

relatively. According to the upper bound of response time $[C_i + \sum_{j<i} C_j (1 - U_j)]/(1 - \sum_{j<i} U_j)$, we can know that the task response time is reduced [42]. Therefore, it converges faster and the running time overhead of the original RTA and improved RTA is relatively reduced.

Experiment 7: All parameters are generated following Experiment 2. The task set utilization U is different, and the comparison of the average running time overhead t is

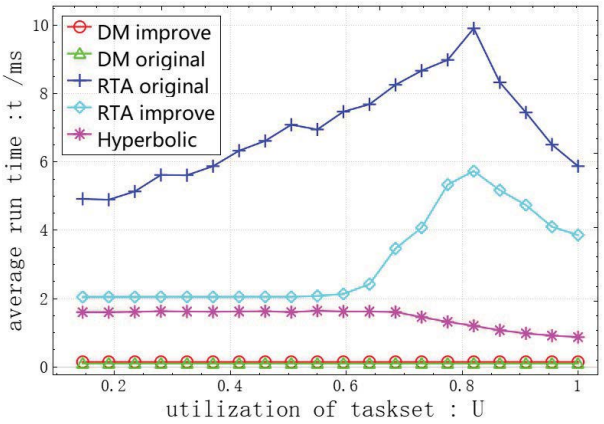

FIGURE 13. The impact of utilization on the run time.

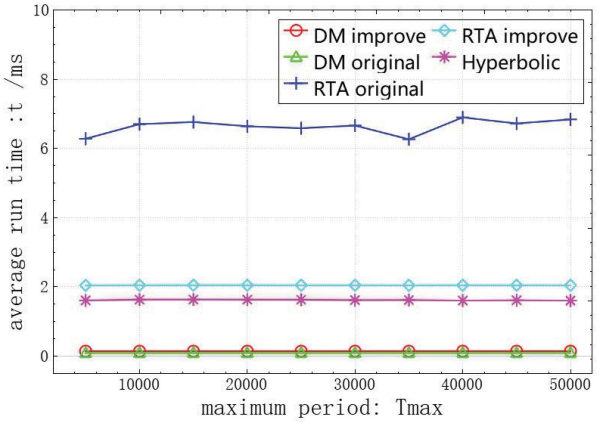

FIGURE 14. The impact of maximum period on the run time.

tested. As shown in Fig. 13, with the utilization of the task set increases, the running time overhead of the original RTA and improved RTA increases significantly. However, the running time cost of other schedulability tests is almost constant.

Experimental analysis: The original DM and improved DM determine the schedulability through the upper bound of density and do not need iteration. Therefore, the change in the utilization of the task set will not have an impact on the running time overhead. According to the upper bound of response time $[C_i+\sum_{j<i} C_j\left(1-U_j\right)]/(1-\sum_{j<i} U_j)$, it can be seen that the worst-case response time of a task is related to the utilization of high-priority tasks. When the utilization increases, the worst-case response time will increase, thus making the number of iterations increase. Therefore, the running time overhead of the original RTA increases. The initial value adopted by improved RTA is $R_i^0 = (D_i - J_i + C_i + B_i)/2$. When the utilization of the task set is low, this initial value is much larger than the response time of the task and the schedulability can be obtained without iteration. Therefore, the increase in the utilization of the task set will not lead to an increase in the runtime overhead of the improved RTA. However, when the utilization is higher than 0.65, the response time of the task is close to the deadline, so that the improved RTA requires multiple iterations to obtain the schedulability result and the running time overhead increases rapidly. When the utilization is higher than 0.7, it can be seen from Experiment 2 that the schedulable ratio drops, so the sufficient test Hyperbolic will fail easily. Hyperbolic will exit early and the running time overhead will gradually decrease. When the utilization is higher than 0.85, tasks are almost unschedulable and the higher priority task cannot be scheduled first, so the exact test ends soon, resulting in the running time overhead of the original RTA and improved RTA being significantly reduced. Therefore, when the task set is at an unschedulable critical utilization, the running time overhead of the exact tests will fluctuate greatly.

Experiment 8: The ratio of task set density to utilization $DT/U = 1.5$, the number of tasks is 30, and the utilization is 0.5. As shown in Fig. 14, the maximum period $T_{\max}$ of tasks is different, and the comparison of the run time overhead t is tested. As the $T_{\max}$ increases, the runtime overhead of the original RTA fluctuates slightly, and other schedulability tests are a constant. The run time overhead of the original DM and improved DM will not change because they only depend on the number of tasks. The period increases will cause the execution time and deadline to increase simultaneously. Therefore, the schedulability of tasks remains unchanged. The run time overhead of the original RTA, improved RTA and Hyperbolic is related to the schedulable ratio. Therefore, the running time overhead of these tests is unchanged with the $T_{\max}$ increases.

In actual engineering applications, we may not know the information of the task set in advance, including utilization, number of tasks and density. In this case, we can combine improved DM and improved RTA to become a new schedulability test called DM with RTA. The pseudo code is shown in Algorithm 2. Improved DM is used first to test the schedulability of the task set. If it is schedulable, it ends. Otherwise, improved RTA will be used to further test the schedulability of the task set.

Algorithm 2 Bool DM With RTA(S_H)

Input: Real-time task set S_H
Output: The schedulability of S_H

```
if DM improve(S_H) then
    return true;
else
    for τ_i : S_H do
        R_i^#0 = (D_i + C_i)/2;
        R_i =RTA improve(τ_i, R_i^#0);
        if R_i > D_i then
            return false;
        end if
    end for
    return true;
end if
```

The next experiment compares the schedulable ratio and running time overhead of this schedulability test and other tests.

TABLE 2. Comparison of DM with RTA and other tests.

Algorithm	schedulable ratio	average run time
DM with RTA	0.8100	0.6979 ms
DM improve	0.5700	0.1069 ms
RTA improve	0.8100	1.0087 ms

Experiment 9: The task set is randomly generated by the following parameters. The number of tasks is generated randomly in the range [1, 30] and the utilization is generated randomly in the range [0.1, 1]. DM with RTA, improved DM and improved RTA perform schedulability determination to task set, and their schedulable ratio and running time overhead are measured. As shown in TABLE 2, the schedulable ratio of DM with RTA is 0.8100 which is the same as improved RTA. However, the run time overhead of DM with RTA is only 0.6979ms and it is much lower than that of improved RTA, making a reduction of 30.8%. Therefore, DM with RTA dominates improved RTA. Compared with improved DM, the schedulable ratio of DM with RTA has increased by 42%, but the running time overhead has increased by about 5.5 times. In general, this running time overhead is an acceptable range.

Since improved DM has linear time complexity characteristics and improved RTA can make an exact schedulability determination, we combined them to get DM with RTA, making full use of the advantages of both. Therefore, the schedulability can be exactly determined, and the running time overhead of the schedulability determination can be reduced. DM with RTA has strong versatility and is a very worth schedulability test.

C. EXPERIMENTAL CONCLUSION

We compare the existing related research results in experiments. The experimental results show that the density upper bound of sporadic task DM scheduling is about 0.69. The running time overhead of improved DM when a task is added dynamically to the task set has nothing to do with the number of tasks and DT/U. It is a constant time, which is equal to the existing density upper analysis bound methods, only 0.005ms. However, the running time overhead of other schedulability tests increases rapidly along with the number of tasks and it is much larger than the density upper bound analysis methods. The lower the ratio of task set density to utilization, the more the number of tasks, the greater the advantage of our improved DM test. In addition, we combined the characteristics of the existing tests to further propose an exact schedulability test DM with RTA, which effectively reduces the running time overhead by 30.8% compared with the state-of-the-art schedulability test. Which one of our methods is used can be determined according to the actual application scenario. If we are very concerned about the run time overhead, we can use improved DM. If we are more concerned about the schedulable ratio, then we can use DM with RTA.

VI. CONCLUSION

We analyze that the density upper bound of sporadic task DM scheduling is about 0.69. The theoretical analysis process is carried out in three steps. Firstly, the case is considered where the task set contains only two tasks and the deadline ratio between the tasks is less than 2. Secondly, the case is considered where the task set contains multiple tasks and the deadline ratio between any two tasks is still less than 2. Finally, the case is considered where the task set contains multiple tasks and the deadline ratio between tasks is an arbitrary value. We compare the existing related research results in experiments. The experimental results show that the density upper bound of sporadic task DM scheduling is about 0.69. The running time overhead of the schedulability test when a task is added dynamically to the task set has nothing to do with the number of tasks. The time complexity is $O(1)$. The running time is equal to the existing density upper analysis bound methods, only 0.005ms. However, the running time of other schedulability tests increases rapidly along with the number of tasks and it is much larger than the density upper bound analysis methods. In addition, we combined the characteristics of the existing tests to further propose an exact schedulability test DM with RTA, which effectively reduces the running time overhead by 30.8% compared with the state-of-the-art schedulability test. We analyze a theoretical density upper bound and it is of great value for theoretical and engineering applications. Due to the high efficiency of our schedulability test, an online schedulability test can be implemented in open real-time systems. Next, we will consider multi-core systems.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their helpful comments.

REFERENCES

[1] G. Lipari and G. Buttazzo, "Resource reservation for mixed criticality systems," in *Workshop on Real-Time Systems: The Past, The Present, and the Future*. 2013.

[2] H.-C. Jo, J.-K. Park, H.-W. Jin, S. H. Lee, and H.-S. Yoon, "Quantitative analysis of ARINC-653 scheduling overheads on multi-core systems," in *Proc. IEEE/AIAA 37th Digit. Avionics Syst. Conf. (DASC)*, Sep. 2018, pp. 1–5.

[3] M. Copic, R. Leupers, and G. Ascheid, "Efficient sporadic task handling in parallel AUTOSAR applications using runnable migration," in *Proc. 24th Asia South Pacific Design Autom. Conf.*, Jan. 2019, pp. 603–608.

[4] Y. Hao, M. Mu, X. Dai, and X. Li, "Schedulability test for IMA systems based on mixed integer linear programming formulation," *TURKISH J. Electr. Eng. Comput. Sci.*, vol. 26, no. 2, pp. 844–855, Mar. 2018.

[5] M. Joseph, "Finding response times in a real-time system," *Comput. J.*, vol. 29, no. 5, pp. 390–395, May 1986.

[6] J. Gong, M. Yang, L. Qiao, H. Yang, B. Gu, F. Peng, Q. Wang, J. Xu, and B. Liu, "A prioritized round-robin preemptive operating system Cron scheduling method," (in Chinese), China Patent 10 371 394 8A, 2015.

[7] W. Yi, "Design and dynamic update of real-time systems," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, Dec. 2019, pp. 1–3.

[8] J. Y.-T. Leung and J. Whitehead, "On the complexity of fixed-priority scheduling of periodic, real-time tasks," *Perform. Eval.*, vol. 2, no. 4, pp. 237–250, Dec. 1982.

[9] A. Minaeva and Z. Hanzálek, "Survey on periodic scheduling for time-triggered hard real-time systems," *ACM Comput. Surv.*, vol. 54, no. 1, pp. 1–32, Apr. 2021.

[10] B. J. Krämer and N. Völker, *Safety-Critical Real-Time Systems*. Norwell, MA, USA: Kluwer, 1997.
[11] S. Baruah and K. Pruhs, "Open problems in real-time scheduling," *J. Scheduling*, vol. 13, no. 6, pp. 577–582, Dec. 2010.
[12] 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.
[13] F. Eisenbrand and T. Rothvoß, "EDF-schedulability of synchronous periodic task systems is coNP-hard," in *Proc. Annu. ACM-SIAM Symp. Discrete Algorithms*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2010, pp. 1029–1034.
[14] Accessed: May 20, 2021. [Online]. Available: https://weston-embedded.com/micrium-kernels
[15] Accessed: May 20, 2021. [Online]. Available: https://www.freertos.org
[16] Accessed: May 20, 2021. [Online]. Available: http://windriver.com/products/vxworks/
[17] P. Ekberg and W. Yi, "Fixed-priority schedulability of sporadic tasks on uniprocessors is NP-hard," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, Dec. 2017, pp. 139–146.
[18] E. Bini, G. C. Buttazzo, and G. M. Buttazzo, "Rate monotonic analysis: The hyperbolic bound," *IEEE Trans. Comput.*, vol. 52, no. 7, pp. 933–942, Jul. 2003.
[19] N. C. Audsley, A. Burns, M. F. Richardson, and A. J. Wellings, "Deadline monotonic scheduling theory," in *Real-Time Programming 1992*. Amsterdam, The Netherlands: Elsevier, 1992, pp. 55–60.
[20] T. F. Abdelzaher, V. Sharma, and C. Lu, "A utilization bound for aperiodic tasks and priority driven scheduling," *IEEE Trans. Comput.*, vol. 53, no. 3, pp. 334–350, Mar. 2004.
[21] V. Bonifaci, A. Marchetti-Spaccamela, N. Megow, and A. Wiese, "Polynomial-time exact schedulability tests for harmonic real-time tasks," in *Proc. IEEE 34th Real-Time Syst. Symp.*, Dec. 2013, pp. 236–245.
[22] J. J. Chen, W. H. Huang, and C. Liu, "k2U: A general framework from k-point effective schedulability analysis to utilization-based tests," in *Proc. IEEE Real-Time Syst. Symp.*, Dec. 2015, pp. 107–118.
[23] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. J. Wellings, "Applying new scheduling theory to static priority pre-emptive scheduling," *Softw. Eng. J.*, vol. 8, no. 5, p. 284, 1993.
[24] F. Eisenbrand and T. Rothvoß, "Static-priority real-time scheduling: Response time computation is NP-hard," in *Proc. Real-Time Syst. Symp.*, Nov. 2008, pp. 397–406.
[25] S. Baruah, M. Bertogna, and G. Buttazzo, *Multiprocessor Scheduling for Real-Time Systems*. Springer, 2015, p. 162.
[26] L. Sha, R. Rajkumar, and J. P. Lehoczky, "Priority inheritance protocols: An approach to real-time synchronization," *IEEE Trans. Comput.*, vol. 39, no. 9, pp. 1175–1185, Sep. 1990.
[27] T. P. Baker, "Stack-based scheduling of realtime processes," *Real-Time Syst.*, vol. 3, no. 1, pp. 67–99, 1991.
[28] K. Tindell, "Using offset information to analyse static priority preemptively scheduled task sets," Dept. Comput. Sci., Univ. York, York, U.K., 1992, pp. 4–5.
[29] J. P. Lehoczky, "Fixed priority scheduling of periodic task sets with arbitrary deadlines," in *Proc. 11th Real-Time Syst. Symp.*, 1991, pp. 201–209.
[30] K. W. Tindell, A. Burns, and A. J. Wellings, "An extendible approach for analyzing fixed priority hard real-time tasks," *Real-Time Syst.*, vol. 6, no. 2, pp. 133–151, Mar. 1994.
[31] J. Lehoczky, L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: Exact characterization and average case behavior," in *Proc. Real-Time Syst. Symp.*, 1990, pp. 166–171.
[32] M. Sjödin and H. Hansson, "Improved response-time analysis calculations," in *Proc. IEEE Real-Time Syst. Symp.*, Dec. 1998, pp. 399–408.
[33] R. J. Bril, W. F. J. Verhaegh, and E.-J.-D. Pol, "Initial values for online response time calculations," in *Proc. Euromicro Conf. Real-Time Syst.*, 2003, pp. 13–22.
[34] W.-C. Lu, K.-J. Lin, H.-W. Wei, and W.-K. Shih, "Period-dependent initial values for exact schedulability test of rate monotonic systems," in *Proc. 21th Int. Parallel Distrib. Process. Symp.*, Long Beach, CA, USA, Mar. 2007, pp. 26–30.
[35] R. I. Davis, A. Zabos, and A. Burns, "Efficient exact schedulability tests for fixed priority real-time systems," *IEEE Trans. Comput.*, vol. 57, no. 9, pp. 1261–1276, Sep. 2008.
[36] L. Schonberger, W.-H. Huang, G. Von Der Bruggen, K.-H. Chen, and J.-J. Chen, "Schedulability analysis and priority assignment for segmented self-suspending tasks," in *Proc. IEEE 24th Int. Conf. Embedded Real-Time Comput. Syst. Appl. (RTCSA)*, Aug. 2018, pp. 157–167.
[37] B. Yalcinkaya, M. Nasri, and B. B. Brandenburg, "An exact schedulability test for non-preemptive self-suspending real-time tasks," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 1228–1233.
[38] S. Zhao, X. Dai, I. Bate, A. Burns, and W. Chang, "DAG scheduling and analysis on multiprocessor systems: Exploitation of parallelism and dependency," in *Proc. RTSS*, Dec. 2020, pp. 128–140.
[39] S. Nogd, G. Nelissen, M. Nasri, and B. B. Brandenburg, "Response-time analysis for non-preemptive global scheduling with FIFO spin locks," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, Dec. 2020, pp. 115–127.
[40] D. L. Weaver, *The SPARC Architecture Manual*. Upper Saddle River, NJ, USA: Prentice-Hall, 1994.
[41] E. Bini and G. C. Buttazzo, "Measuring the performance of schedulability tests," *Real-Time Syst.*, vol. 30, nos. 1–2, pp. 129–154, May 2005.
[42] R. I. Davis and A. Burns, "Response time upper bounds for fixed priority real-time systems," in *Proc. Real-Time Syst. Symp.*, Nov. 2008, pp. 407–418.

HONGBIAO LIU received the B.S. degree in applied physics from the South China University of Technology, in 2017, and the M.E. degree from the School of Computer Science and Technology, Xidian University, Xi'an, in 2019, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Technology. His current research interests include operating system real-time scheduling and formal verification.

XI CHEN received the B.E. degree in telecommunication engineering from the Beijing University of Posts and Telecommunications, in 2019. She is currently pursuing the M.E. degree with the Beijing Institute of Control Engineering. Her current research interests include operating system real-time scheduling and formal verification.

LEI QIAO received the Ph.D. degree in computer science from USTC, Hefei, in 2007. He is currently a Professor with the Beijing Institute of Control Engineering. His research interests include operating system design and formal verification.

JINGKUN ZHANG received the M.E. degree from the Beijing Institute of Control Engineering, in 2018. He is currently working with the Beijing Institute of Control Engineering. His current research interests include operating system real-time scheduling and formal verification.

MENGFEI YANG received the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University. He is currently a Professor with the China Academy of Space Technology. He is also an Academician with the Chinese Academy of Sciences. His research interests include operating systems, formal verification, and artificial intelligence.