3-2-2018

Scheduling Based on Interruption Analysis and PSO for Strictly Periodic and Preemptive Partitions in Integrated Modular Avionics

Hui Lu
Beihang University, China

Qianlin Zhou
Beihang University, China

Zongming Fei
University of Kentucky, zongming.fei@uky.edu

Rongrong Zhou
Beihang University, China

Scheduling Based on Interruption Analysis and PSO for Strictly Periodic and Preemptive Partitions in Integrated Modular Avionics

Notes/Citation Information
Published in *IEEE Access*, v. 6, p. 13523-13539.

Digital Object Identifier (DOI)
https://doi.org/10.1109/ACCESS.2018.2811539

IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received January 15, 2018, accepted February 22, 2018, date of publication March 2, 2018, date of current version March 28, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2811539

Scheduling Based on Interruption Analysis and PSO for Strictly Periodic and Preemptive Partitions in Integrated Modular Avionics

HUI LU[1], QIANLIN ZHOU[1], ZONGMING FEI[2], AND RONGRONG ZHOU[1]
[1]School of Electronic and Information Engineering, Beihang University, Beijing 100191, China
[2]Department of Computer Science, University of Kentucky, Lexington, KY 40506-0495, USA

Corresponding author: Hui Lu (mluhui@vip.163.com)

This work was supported by the National Natural Science Foundation of China under Grant 61671041 and Grant 61101153.

ABSTRACT Integrated modular avionics introduces the concept of partition and has been widely used in avionics industry. Partitions share the computing resources together. Partition scheduling plays a key role in guaranteeing correct execution of partitions. In this paper, a strictly periodic and preemptive partition scheduling strategy is investigated. First, we propose a partition scheduling model that allows a partition to be interrupted by other partitions, but minimizes the number of interruptions. The model not only retains the execution reliability of the simple partition sets that can be scheduled without interruptions, but also enhances the schedulability of the complex partition sets that can only be scheduled with some interruptions. Based on the model, we propose an optimization framework. First, an interruption analysis method to decide whether a partition set can be scheduled without interruptions is developed. Then, based on the analysis of the scheduling problem, we use the number of interruptions and the sum of execution time for all partitions in a major time frame as the optimization objective functions and use particle swarm optimization (PSO) to solve the optimization problem when the partition sets cannot be scheduled without interruptions. We improve the update strategy for the particles beyond the search space and round all particles before calculating the fitness value in PSO. Finally, the experiments with different partitions are conducted and the results validate the partition scheduling model and illustrate the effectiveness of the optimization framework. In addition, other optimization algorithms, such as genetic algorithm and neural networks, can also be used to solve the partition problem based on our model and solution framework.

INDEX TERMS Integrated modular avionics, partition scheduling model, optimization framework, interruption analysis, particle swarm optimization.

I. INTRODUCTION

With the development of the microelectronic technology and software technology, the system architecture of avionics has been evolving from traditional discrete and federated stages to integrated and highly-integrated stages [1]. In the new generation of the avionics system, integrated modular avionics architecture was proposed and has been validated on many large passenger planes, like A380 and B7E7. It is one kind of highly-integrated avionics under software control mode and aims at standardization, reusability and interchangeability of avionics modules. Generally, the core idea of IMA is hardware resource-sharing mode. Many applications utilize the same computing, communication and I/O resources to reduce the hardware redundancy and improve the resource utilization [2], [3]. Therefore, IMA can easily achieve the goal of reduction in size, cost and weight [4] and the greater flexibility in resources allocation.

A. THE ANALYSIS OF THE RELATED WORK

In order to guarantee that one or more avionics applications can execute independently in a core module, IMA introduces the concept of partition [5]–[7], which is similar to a program in a single application environment. The partitions are divided based on the functions of the applications and each partition is activated in one or more time-windows allocated by the system. Each partition has no effect on other partitions in time and space. All partitions in a core module share the common resources. Besides, each partition contains processes to complete the corresponding application. The system resources occupied by a partition are shared by all processes in it. In order to guarantee the stability of the system, a high performance two-level scheduling strategy for partitions and processes is critical for the operation system to allocate the occupation time of the processor, memory and other resources for each partition. There are two trends for the two-level scheduling problem, the hybrid solution and the hierarchical solution. Many researches considered the two-level scheduling of partitions and processes simultaneously. However, the scheduling strategy of the process can adopt the classic scheduling algorithms in embedded system, like earliest deadline first (EDF) [8], least laxity first (LLF) [9] scheduling algorithms, which have excellent performance. At the same time, considering the two-level scheduling algorithms simultaneously makes it difficult to optimize the scheduling problem. Therefore, we study the scheduling problem of partition and process separately. Then we combine them together in meeting the constraints of each other. In this paper, we consider the partition scheduling problem on a single processor.

Avionics application standard software interface named ARINC653 gives the basic rules for partition scheduling [6], [10], [11]. First, partition scheduling is strictly deterministic over time. Second, all partitions have no priority and they can only execute in their own time-windows. Third, the scheduling algorithm is predefined and all partitions execute in a certain period. Based on these rules, many researchers have proposed a variety of scheduling models and analyzed the schedulability for arbitrary partition sets. Several proposed models even break the restriction that partitions do not have priority.

Round robin (RR) scheduling is the most frequently adopted strategy for partition scheduling problem. On the basis of RR scheduling, Sheikh *et al.* [12] used the model that arbitrary two partitions cannot be released with overlap. They proposed an optimization goal and a best-response algorithm based on the game theory to achieve the maximum stability for the schedulable partition set under their partition model. However, the model cannot schedule the partitions with complex periods. It means that if the partition periods are coprime, the schedulability of the system will be sharply reduced. Lee *et al.* [13] presented a partition and channel-scheduling algorithm for the strong partitioned real-time system. They used a two-level hierarchical schedule that activates partitions following a distance-constraints guaranteed cyclic schedule and then dispatches tasks according to a fixed priority schedule. However, they did not give a specific scheduling algorithm. Tao *et al.* [14] proposed a scheduling scheme with partition readjustment based on the fixed priority strategy. Through adjusting the length of each partition and reconstructing them, their scheme can reduce the resource costs and improve schedulability. In addition, they also gave the partition adjustment algorithm based on their scheduling model. However, the algorithm will change the number of partitions. Gui *et al.* [15] proposed a partition scheduling model that always allocates the time slots for the newly released partitions, and gave rules to ensure the correct execution of all partitions. They also proved that their model has the maximum schedulability for complex partition sets. However, partitions may be interrupted frequently.

For the case of multiprocessor, Eisenbrand *et al.* [16] scheduled the periodic tasks on a minimum number of processors. In addition, they proved that there exists a 2-approximation for the minimization problem when the periods are harmonic. Kermia and Sorel [17] dealt with the non-preemptive scheduling of tasks onto multi-processor by considering both precedence relation and periodicity constraints. Their objective was to minimize the global execution time of the system.

Some researchers studied the schedulability of partitions for IMA. Wan and Tian [18] considered the partition scheduling as a fixed priority preemptive scheduling problem on a single processor and analyzed the condition of schedulability for several periodic tasks based on the rate monotonic (RM) algorithm [8], [19]. Marouf and Sorel [20] considered the cases of the tasks with harmonic periods and the tasks with non-harmonic periods separately. They gave the schedulability conditions for the harmonic case and proposed local schedulability conditions for the non-harmonic case. However, the above analyses depended on the specific scheduling model.

From the above researches, on a single processor, there exist some defects on the scheduling models and the scheduling algorithms based on the analysis of the current situation. First, the existing model can be divided into two categories. One forbids interruptions for all partitions. It means that a partition must be finished once it is released and other partitions cannot be released when one partition is running in the processor, as in the model proposed by Sheikh *et al.* [12]. The other allows a running partition to be interrupted by the new coming partition, as in the model proposed by Gui *et al.* [15]. For periodic partitions, the former scheme will greatly reduce the schedulability of the partition set, especially for the partition sets with non-harmonic periods. For the latter scheme, the influence caused by the interruption is ignored and the partitions are allowed to be released at any time. Therefore, the partitions are likely to be interrupted frequently, although the processor utilization and the schedulability of the partition set will be enhanced. As for the existing algorithms, they are designed based on the specific models and most of them cannot handle other models effectively.

B. OUR WORK

In this paper, we propose a comprehensive partition scheduling model which combines the advantages of the existing two models without violating the definition and rules in ARINC 653. Based on the model, we develop an optimization framework for the partition scheduling model. It can be divided into two steps. The first one is the interruption analysis to determine whether the partition set is schedulable without interruptions. The second step is to use an appropriate algorithm to optimize the scheduling model. The appropriate algorithm framework consists of two parts based on the result of the first step. For the partition sets that are schedulable without interruptions the framework uses the optimization goal and the algorithm proposed by Sheikh *et al.* [12]. For the rest of the more general partition sets, we analyze the properties of the partition scheduling problem of our model, and improve particle swarm optimization (PSO) [21]–[24] to search a good scheduling scheme. Other meta-heuristic algorithms can replace the improved PSO as the scheduling algorithm. Our contribution can be described in three aspects.

First, based on the analysis of the model proposed by Sheikh *et al.* and the model proposed by Gui *et al.*, we propose a comprehensive partition scheduling model that uses different scheduling strategies to schedule different partitions based on whether the partition set is schedulable without interruptions. The main idea is that the interruption is allowed but it should be avoided as much as possible. For the schedulable partition sets without interruption, the model uses the scheduling scheme proposed by Sheikh *et al.* [12]. However, the complex partition sets that cannot be scheduled without interruption are more common, and we use the minimum interruption strategy to schedule this type of partition sets. Therefore, compared with the existing two models, our model can improve partition sets' schedulability and reduce the number of interruptions.

Second, we propose an optimization framework to optimize our scheduling model. The optimization framework can be divided into two steps, which are interruption analysis and algorithm optimization. For the first step, we design an interruption analysis method to determine the schedulability of the partition sets without interruptions. There exist four cases for the relationship of the arbitrary two partition's periods. We analyze the schedulability of the arbitrary two partitions for each case. Therefore, we can obtain the schedulability of the arbitrary partition sets without interruptions by using two partitions as the basis and expanding the number of considered partitions gradually until all partitions are considered. This is a critical step to decide which optimization strategy will be used to search the scheduling scheme for the partition set.

Finally, for the second step of the optimization framework, we propose two objective functions with one playing a supplementary role for the other to evaluate the performance of each candidate solution. Besides, based on the properties of the scheduling model, we use PSO to optimize all partitions' first release time points for the partition set that cannot be scheduled without interruptions. We improve the randomization strategy to update the positions of the particles that fly beyond the search space so that all candidate solutions can have the same possibility to be searched. In addition, the updated positions of all particles are rounded before calculating the fitness values. The rounding operation can make the algorithm search better solutions more easily based on the model we proposed. When the partition set is schedulable without interruptions, we will use the optimization goal and

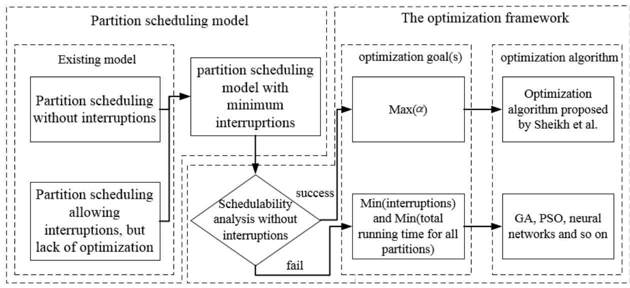

FIGURE 1. The process and the solution framework in the paper.

algorithm proposed by sheikh *et al.* to optimize the partition scheduling problem.

The basic process and the solution framework proposed in the paper is shown in Fig. 1. The meaning of the optimization goal proposed by Sheikh *et al.* when the partition set is schedulable without interruptions will be briefly introduced in section III.B.

The rest of the paper is organized as following. Section II investigates the strictly periodic and preemptive partition scheduling model and the objective functions. In section III, the interruption analysis to determine the partition set's schedulability without interruptions and the corresponding optimization scheme and the optimization algorithm are presented. Section IV gives experiment results and analyses to show the effectiveness of our proposed model and the solution frameworks. In addition, the properties of the partition scheduling problem are also addressed in this section. Finally, a brief conclusion follows in Section V.

II. STRICTLY PERIODIC AND PREEMPTIVE PARTITION SCHEDULING MODEL

A. PARTITION SCHEDULING MODEL

In this paper, we focus on the partitions on one processor. Assume a set of partitions $\Pi = \{P_1, P_2, P_3, \cdots, P_n\}$. Every partition P_i has a period $m_i T$ and execution time C_i. There are some hypotheses and rules.

(1) All partitions have the same operation in each major time frame (MTF),which means the minimum and fixed cycle of the processor's runtime operation, can be defined as the least common multiple (LCM) of all partitions' periods. The formula to calculate MTF is as following.

$$MTF = LCM(m_1 T, m_2 T, m_3 T \cdots m_n T) \tag{1}$$

(2) The running times for P_i in each MTF can be calculated as follows.

$$N_i = MTF / m_i T \tag{2}$$

(3) Each partition cannot be executed in other partition's time-windows, including the idle time-windows.

(4) Each partition may have one or more time-windows in MTF, and the time-windows can have different length.

(5) The necessary and sufficient condition for each partition's schedulability is that the partition has been finished before a new release.

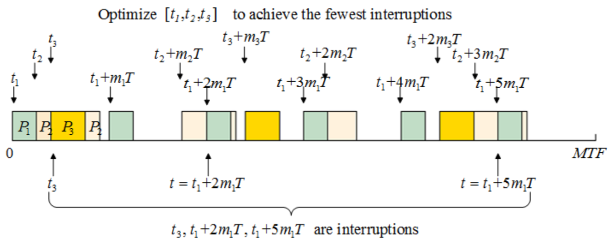

FIGURE 2. The illustration for partition scheduling problem.

(6) The new coming partition can be executed first, and the execution status of the partition which is being executed will be saved in cache. Each time when it happens, we regard it as an interruption.

(7) The partition with closer next release will resume execution first when two or more partitions are in the cache [15].

(8) The partition with a smaller period will be executed first when two or more partitions are released simultaneously.

(9) The time of the first release of the partition with the minimum period is used as the start of MTF.

Based on the above hypotheses and rules, if the first release time (FRT) for each partition $[t_1, t_2, t_3 \cdots t_n]$ is given, the processor's runtime operation is certain and we can determine whether the partition set is schedulable. Therefore, the aim of scheduling is to search the optimal FRT with regard to the objective functions, like the number of interruptions and the sum of execution time. The illustration of the scheduling problem is shown in Fig. 2.

B. SCHEDULING OBJECTIVE

There are two objectives. They are the number of interruptions for all partitions in each major time frame and the sum of execution time for all partitions in each major time frame. It is worth emphasizing that the scheduling optimization is not a multi-objective optimization problem. We use the number of interruptions for all partitions as the primary objective. If more than one optimal solutions are obtained, we use the sum of execution time for all partitions in each major time frame as the auxiliary objective.

1) THE NUMBER OF INTERRUPTIONS FOR ALL PARTITIONS IN *MTF*

Interruption means that a running partition is interrupted by a new coming partition. The jitter of release with the interruption will increase the uncertainty of the processor's runtime operation. Therefore, the smaller the number of interruptions is, the less likely the error caused by interruptions will occur. To guarantee the certainty of the partition's execution, we select the number of interruptions for all partitions in MTF as the primary optimization goal. It is shown as following.

$$f_1(FRT) = min\ NI, \quad satisfy\ F_{P_i}(s_i) \le R_{P_i}(s_i+1) \quad \forall P_i \in \{P_1, P_2, P_3 \cdots P_n\}$$

Here, NI is the number of interruptions for all partitions. $F_{P_i}(s_i)$ is the finish time of the s_i-th release of partition P_i. $R_{P_i}(s_i+1)$ is the (s_i+1)-th release time of partition P_i.

TABLE 1. Partition parameters.

Partition	m_iT/ms	C_i/ms	FRT_1/ms	FRT_2/ms
P_1	20	5	0	0
P_2	30	6	5	17
P_3	40	7	12	9

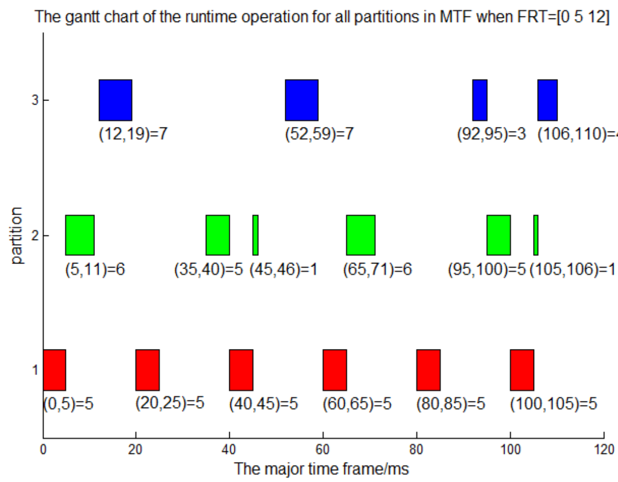

FIGURE 3. The gantt chart of the runtime operation for all partitions with FRT_1.

The number of interruptions can be calculated based on Eq. (3) when the runtime operation of all partitions is known.

$$NI = STW - SN \tag{3}$$

Here, STW is the number of all partition' time-windows in MTF. SN is the number of all partitions' execution times in MTF. SN is a constant for a certain partition set. Based on Eq. (3), the smaller time-windows for all partitions means the fewer interruptions. Therefore, the optimal scheduling scheme with the minimum interruptions can make each release of all partitions execute completely in the smallest number of possible time-windows.

It should be emphasized that we do not consider the situation that two or more partitions are released simultaneously as an interruption. Based on the rules of the execution, if there are no other releases, the partitions released simultaneously will be executed in ascending order of their periods.

2) THE SUM OF EXECUTION TIME FOR ALL PARTITIONS IN *MTF*

The minimum number of interruptions for all partitions in MTF can guarantee the partition's certainty. However, there may exist many different solutions of FRT with the same number of interruptions but different schedule for all partitions. An example is shown in Table 1. The two scheduling results of the example are shown in Fig. 3 and Fig. 4.

The number of interruptions in Fig. 3 and Fig. 4 can be calculated based on Eq. (3).

$$NI_{FRT_1} = \underbrace{(6+6+4)}_{STW} - \underbrace{(6+4+3)}_{SN} = 3 \tag{4}$$

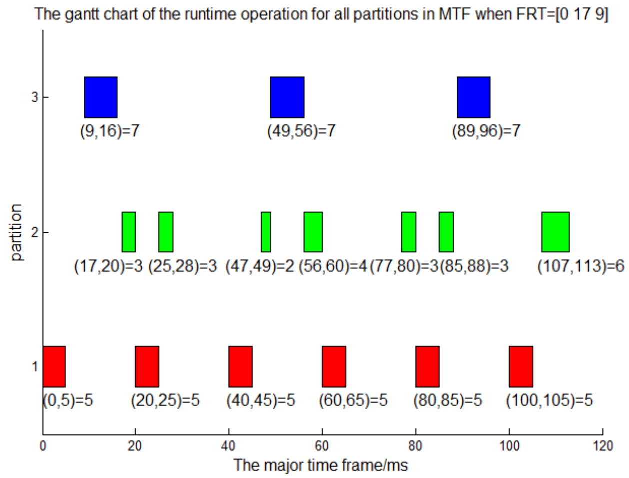

FIGURE 4. The gantt chart of the runtime operation for all partitions with FRT_2.

NI_{FRT_2} is equal to NI_{FRT_1}. However, the runtime operations for partition 2 and partition 3 are totally different. We define the sum of execution time (SET) for all partitions in MTF as another optimization goal to select the best FRT when the number of interruptions is the same. SET includes the interruption time of all partitions and equals to the sum of the difference of the finish time and the beginning time for all partitions' every release. The smaller SET is, the shorter the interruption time for all partitions is. Therefore, the objective function can be described as following.

$$f_2(FRT') = min\ SET$$

FRT' is the set of FRT which can make NI obtain the same minimum. SET can be calculated as following if the runtime operation of all partitions is known.

$$SET = SAC + \sum_{i=1}^{n} k_i C_i \tag{5}$$

Here, SAC is the sum of all partitions' execution time without any interruptions. $k_i C_i$ is the sum of the partition P_i's execution time in the interruption. Based on the execution process of all partitions in Fig. 3 and Fig. 4, we can obtain SET as following.

$$\begin{aligned} SET_{FRT_1} &= \underbrace{5+5+5+5+5+5}_{P_1} \\ &\quad + \underbrace{6+5+5+1+6+5+5+1}_{P_2} \\ &\quad + \underbrace{7+7+3+4+5+5+1+4}_{P_3} \\ &= \underbrace{6\times5+4\times6+3\times7}_{SAC} + \underbrace{3\times5+1\times6+0\times7}_{\sum_{i=1}^{n} k_i C_i} \\ &= 96ms \end{aligned} \tag{6}$$

$$\begin{aligned} SET_{FRT_2} &= \underbrace{5+5+5+5+5+5}_{P_1} \\ &\quad + \underbrace{3+5+3+2+7+4+3+5+3+6}_{P_2} \\ &\quad + \underbrace{6+7+7}_{P_3} \\ &= \underbrace{6\times5+4\times6+3\times7}_{SAC} + \underbrace{2\times5+0\times6+1\times7}_{\sum_{i=1}^{n} k_i C_i} \\ &= 92ms \end{aligned} \tag{7}$$

Therefore, FRT_2 is better than FRT_1.

Though the fewer interruptions means the smaller SET in general, there exists the situation that a FRT is corresponding to more interruptions but smaller SET compared with other FRT. SET reflects the impact of the interruptions, but cannot replace the interruptions. Therefore, we select NI and SET as two optimization goals, and the latter is supplementary for the former.

III. STRICTLY PERIODIC AND PREEMPTIVE PARTITION SCHEDULING OPTIMIZATION

In this section, we introduce the optimization framework that we proposed for our model. In the optimization framework, we first propose an interruption analysis method to determine whether a partition set is schedulable without interruptions. For the schedulable partition sets without interruptions, we give a simple summarization of the optimization goal and the algorithm proposed by Sheikh *et al.* [12] which is effective to optimize the partition scheduling. For the non-schedulable partition sets without interruptions, we give the detailed algorithm to calculate NI and SET. Here, many optimization algorithms can be used to optimize FRT for obtaining the minimum NI and SET. Due to the features of simple structure and simple process of PSO, we improve it based on the properties of the scheduling problem to optimize FRT to show the general process of the optimization and the details that should be considered.

A. INTERRUPTION ANALYSIS

We discuss the process of the interruption analysis method in two steps. The first step is the available time analysis of FRT without interruptions for two partitions. The second step is the comprehensive analysis for the arbitrary partition sets.

1) AN EXAMPLE TO EXPLAIN THE INTERRUPTION ANALYSIS FOR TWO PARTITIONS

Assume there are two partitions P_1, P_2 with periods 2T and 3T, respectively. The execution time of P_1 and P_2 is C_1 and C_2, respectively. Based on the rules, MTF is equal to 6T, and the first release time of partition P_1 is zero. As shown in Fig. 5(a), the runtime intervals of partition P_1 allocated by MTF is $[0, C_1]$, $[2T, 2T + C_1]$ and $[4T, 4T + C_1]$ if partition P_2 is not considered. Here, we use t_2 to denote the release time of partition P_2. If there are no interruptions, every runtime of

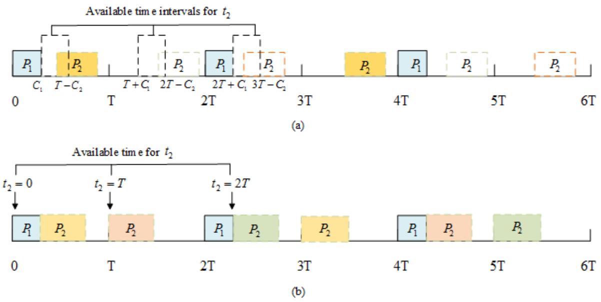

FIGURE 5. The available time for t_2.

partition P_2 cannot overlap these three intervals. To guarantee the complete execution of the last release in MTF for partition P_2, t_2 should be in $[0, 3T - C_2]$. If we move t_2 from 0 to $3T - C_2$, three available time-intervals for t_2 can be obtained.

$$\begin{aligned} &Available\ t_2\ s.t.\ (n_2 \times 3T + t_2, n_2 \times 3T + t_2 + C_2) \\ &\cap (n_1 \times 2T, n_1 \times 2T + C_1) = 0 \quad (n_1 = 0, 1, 2;\ n_2 = 0, 1) \quad (8) \end{aligned}$$

Here, n_1 is the n_1-th execution of P_1 and n_2 is the n_2-th execution of P_2.

Based on Eq. (8), the three available time-intervals for t_2 are as following.

$$\begin{cases} t_2 \geq C_1 \\ t_2 + C_2 \leq 2T \\ 3T + t_2 \geq 2T + C_1 \\ 3T + t_2 + C_2 \leq 4T \end{cases} \Rightarrow t_2 \in [C_1, T - C_2] \quad (9)$$

$$\begin{cases} t_2 \geq C_1 \\ t_2 + C_2 \leq 2T \\ 3T + t_2 \geq 4T + C_1 \end{cases} \Rightarrow t_2 \in [T + C_1, 2T - C_2] \quad (10)$$

$$\begin{cases} t_2 \geq 2T + C_1 \\ t_2 + C_2 \leq 4T \\ t_2 + C_2 \leq 3T \\ 3T + t_2 \geq 4T + C_1 \end{cases} \Rightarrow t_2 \in [2T + C_1, 3T - C_2] \quad (11)$$

The necessary and sufficient condition for the existence of the three time-intervals is $aT + C_1 \leq (a + 1)T - C_2$ $(a = 0, 1, 2)$, namely $C_1 + C_2 \leq T$.

In addition, there also exist some available time points which make different partitions release simultaneously. They can be calculated as following.

$$\begin{aligned} t_2 = \{t_2 | &t_1 + n_1 \times 2T = t_2 + n_2 \times 3T, \\ &n_1 = 0, 1, 2; n_2 = 0, 1; t_1 = 0;\ 0 \leq t_2 \leq 3T - C_2\} \quad (12) \end{aligned}$$

Therefore, 0,T and 2T, as the available time points for t_2, can be obtained. As shown in Fig. 5(b), when t_2 is equal to 0, the first release of P_1 and P_2 is simultaneous, and MTF allocates the time window $[0, C_1]$ to P_1 and $[C_1, C_1 + C_2]$ to P_2. When t_2 is equal to T, the second release of P_2 and the third release of P_1 are simultaneous, and MTF allocates the time window $[4T, 4T + C_1]$ to P_1 and $[4T + C_1, 4T + C_1 + C_2]$ to P_2. When t_2 is equal to 2T, the first release of P_2 and the second release of P_1 are simultaneous, and MTF allocates the time window $[2T, 2T + C_1]$ to P_1 and $[2T + C_1, 2T + C_1 + C_2]$ to P_2. The necessary and sufficient condition for the existence of time points 0,T and 2T is $C_i \leq T\ (i = 1, 2)$.

Therefore, the available time for t_2 is shown as Eq. (13).

$$\begin{aligned} &t_2 \in [aT + C_1, (a + 1)T - C_2] \\ &(a = 0, 1, 2;\ C_1 + C_2 \leq T) \cup \{0, T, 2T\} \quad (C_i \leq T\ (i = 1, 2)) \quad (13) \end{aligned}$$

2) INTERRUPTION ANALYSIS FOR ARBITRARY TWO PARTITIONS

The relationship of the periods for the arbitrary two partitions in a partition set $\Pi = \{P_1, P_2, P_3, \cdots, P_n\}$ can be divided into four categories. The first one is that m_i and m_j are equal. The second one is that m_i is a divisor of m_j but not equal to m_j. The third one is that m_i and m_j are coprime. The last one is that m_i and m_j have common factor greater than 1 and m_i is not a divisor of m_j $(m_i < m_j)$, such as 6 and 9. However, the arbitrary two partitions do not always include the partition with the minimum period in a partition set. As a result, the first release time of partitions P_i and P_j are both variable. Assume $m_i < m_j$ if m_i is not equal to m_j. Therefore, we discuss the available t_j based on the value of t_i.

a: m_i EQUALS TO m_j

When m_i is equal to m_j, both partitions P_i and P_j are executed only once in $MTF' = LCM(m_iT, m_jT)$, as shown in Fig. 6.

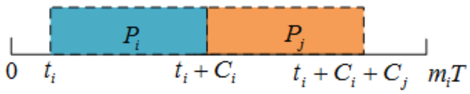

FIGURE 6. The gantt chart of partition P_i and P_j in m_iT.

Based on Fig. 6, if $C_i + C_j$ is greater than m_iT, partition P_j is not schedulable no matter what t_j is. When $C_i + C_j$ is not greater than m_iT, there are two cases based on the size of $C_i + 2C_j$ and m_iT. If $C_i + 2C_j$ is greater than m_iT, there must exist an interval for t_i to make P_j non-schedulable when t_i slides from 0 to $m_iT - C_i$, as shown in Fig. 7(a). If $C_i + 2C_j$ is not greater than m_iT, P_j is always schedulable no matter what t_j is equal to, as shown in Fig. 7(b).

Based on Fig. 7(a), when $0 \leq t_i \leq m_iT - C_i - C_j$, the available interval for t_j is $[t_i + C_i, m_iT - C_j]$, which ensures no interruption for partitions P_i and P_j. In addition, if partitions P_i and P_j release simultaneously, meaning $t_i = t_j$, there is also no interruption based on the rules in section II. Therefore, the available t_j is $[t_i + C_i, m_iT - C_j] \cup \{t_i\}$ when $0 \leq t_i \leq m_iT - C_i - C_j$. In addition, available t_j does not exist when $m_iT - C_i - C_j < t_i < C_j$ and $t_j \in [0, t_i - C_j]$ when $C_j \leq t_i \leq m_iT - C_i$. For the case in Fig. 7(b), the analysis process is similar to the above. Therefore, all the available intervals for t_j are as following when m_i equals to m_j.

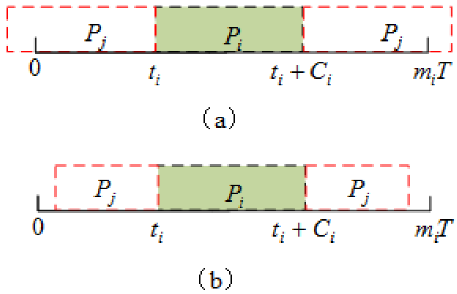

FIGURE 7. Two cases based on the size of $2C_j + C_i$ and m_iT. (a) $m_iT < 2C_j + C_i$ (b) $2C_j + C_i \le m_iT$.

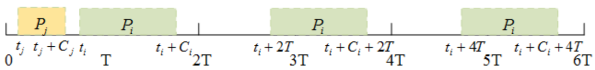

FIGURE 8. The gantt chart of partition P_i and P_j in $6T$.

a) $2C_j + C_i \le m_iT$

$$t_j \in \begin{cases} [t_i + C_i, m_iT - C_j] \cup \{t_i\} \quad 0 \le t_i < C_j \\ [0, t_i - C_j] \cup [t_i + C_i, m_iT - C_j] \cup \{t_i\} \\ \quad C_j \le t_i \le m_iT - C_i - C_j \\ [0, t_i - C_j] \\ \quad m_iT - C_i - C_j < t_i \le m_iT - C_i \end{cases} \tag{14}$$

b) $C_j + C_i \le m_iT < 2C_j + C_i$

$$t_j \in \begin{cases} [t_i + C_i, m_iT - C_j] \cup \{t_i\} \\ \quad 0 \le t_i \le m_iT - C_i - C_j \\ \emptyset \quad m_iT - C_i - C_j < t_i < C_j \\ [0, t_i - C_j] \quad C_j \le t_i \le m_iT - C_i \end{cases} \tag{15}$$

c) $C_j + C_i > m_iT$

$$t_j \in \emptyset \tag{16}$$

b: m_i IS A DIVISOR OF m_j BUT NOT EQUAL TO m_j

Take $m_i = 2$ and $m_j = 6$ as an example, partition P_i will execute for three times while P_j executes only once in $MTF' = LCM(2T, 6T) = 6T$, as shown in Fig. 8.

Based on Fig. 8, the maximum idle interval is $m_iT - C_i$ when only partition P_i is executing. If C_j is grater than $m_iT - C_i$, meaning $C_j + C_i > m_iT$, partition P_j is not schedulable no matter what t_j is.

The analysis is similar to the case that m_i is equal to m_j. For example, under the premise that $C_j + C_i \le m_iT < 2C_j + C_i$, if $0 \le t_i \le m_iT - C_i - C_j$, the available interval for t_j is $[t_i + C_i, m_iT + t_i - C_j]$, $[m_iT + t_i + C_i, 2m_iT + t_i - C_j] \ldots [am_iT + t_i + C_i, (a+1)m_iT + t_i - C_j]$ $(a = m_j/m_i - 2)$ and $[(m_j/m_i - 1)m_iT + t_i + C_i, m_jT - C_j]$. Besides, if partitions P_i and P_j release simultaneously, the available time for t_j is $\{t_i + bm_iT\}(b = 0, 1 \cdots m_j/m_i - 1)$. Therefore, the available interval for t_j is $[am_iT + t_i + C_i, (a+1)m_iT + t_i - C_j] \cup [(m_j/m_i - 1)m_iT + t_i + C_i, m_jT - C_j] \cup \{t_i + bm_iT\}(a = 0, 1 \cdots m_j/m_i - 2; b = 0, 1 \cdots m_j/m_i - 1)$. Similar to the above analysis, all the available intervals for t_j are as following when m_i is a divisor of m_j but not equal to m_j.

a) $2C_j + C_i \le m_iT$

$$t_j \in \begin{cases} [am_iT + t_i + C_i, (a+1)m_iT + t_i - C_j] \\ \quad \cup [(m_j/m_i - 1)m_iT + t_i + C_i, m_jT - C_j] \\ \quad \cup \{t_i + bm_iT\} \quad (a = 0, 1 \cdots m_j/m_i - 2; \\ \quad b = 0, 1 \cdots m_j/m_i - 1) \quad 0 \le t_i < C_j \\ [0, t_i - C_j] \cup [am_iT + t_i + C_i, (a+1)m_iT + t_i - C_j] \\ \quad \cup [(m_j/m_i - 1)m_iT + t_i + C_i, m_jT - C_j] \\ \quad \cup \{t_i + bm_iT\} \quad (a = 0, 1 \cdots m_j/m_i - 2; \\ \quad b = 0, 1 \cdots m_j/m_i - 1) \quad C_j \le t_i \le m_iT - C_i - C_j \\ [0, t_i - C_j] \cup [am_iT + t_i + C_i, (a+1)m_iT + t_i - C_j] \\ \quad \cup \{t_i + bm_iT\} \quad (a = 0, 1 \cdots m_j/m_i - 2; \\ \quad b = 0, 1 \cdots m_j/m_i - 2) \quad m_iT - C_i - C_j < t_i \le m_iT - C_i \end{cases} \tag{17}$$

b) $C_j + C_i \le m_iT < 2C_j + C_i$

$$t_j \in \begin{cases} [am_iT + t_i + C_i, (a+1)m_iT + t_i - C_j] \\ \quad \cup [(m_j/m_i - 1)m_iT + t_i + C_i, m_jT - C_j] \\ \quad \cup \{t_i + bm_iT\} \quad (a = 0, 1 \cdots m_j/m_i - 2; \\ \quad b = 0, 1 \cdots m_j/m_i - 1) \quad 0 \le t_i \le m_iT - C_i - C_j \\ [am_iT + t_i + C_i, (a+1)m_iT + t_i - C_j] \\ \quad \cup \{t_i + bm_iT\} \quad (a = 0, 1 \cdots m_j/m_i - 2; \\ \quad b = 0, 1 \cdots m_j/m_i - 2) \quad m_iT - C_i - C_j < t_i < C_j \\ [0, t_i - C_j] \cup [am_iT + t_i + C_i, (a+1)m_iT \\ \quad + t_i - C_j] \cup \{t_i + bm_iT\} \quad (a = 0, 1 \cdots m_j/m_i - 2; \\ \quad b = 0, 1 \cdots m_j/m_i - 2) \quad C_j \le t_i \le m_iT - C_i \end{cases} \tag{18}$$

c) $C_j + C_i > m_iT$

$$t_j \in \emptyset \tag{19}$$

c: m_i AND m_j are coprime

Based on the rules in section II, we distinguish two cases based on whether there exist some different partitions that release simultaneously. t_{j1} represents the available time intervals for partition P_j obtained by the case that all release time points are unequal, and t_{j2} represents the available release time for partition P_j meeting the case that some releases of partitions P_i and P_j are simultaneous.Therefore, all available time-intervals for t_j can be calculated by the following formula.

$$t_j = t_{j1} \cup t_{j2} \tag{20}$$

- The analysis of t_{j1}

Take $m_i = 2$ and $m_j = 3$ as an example, partition P_i will execute three times while P_j executes twice in $MTF' = LCM(2T, 3T) = 6T$, as shown in Fig. 9.

Based on Fig. 9, the maximum idle interval is $T - C_i$ as long as the periods of partitions P_i and P_j are coprime. Therefore,

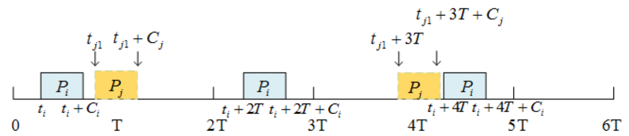

FIGURE 9. The gantt chart of partition P_i and P_j in $6T$.

partition P_j is not schedulable no matter what t_{j1} is if C_j is greater than $T - C_i$, i.e., $C_j + C_i > T$.

Under the premise that $C_j + C_i \le T < 2C_j + C_i$, there are different available time-intervals for t_{j1} when t_i slides from 0 to $m_iT - C_i$. Assume that $m_i = 2$ and $m_j = 3$, if $0 \le t_i \le T - C_i - C_j$, there are three time-intervals for t_{j1}. They are $[t_i + C_i, T + t_i - C_j]$, $[T + t_i + C_i, 2T + t_i - C_j]$ and $[2T + t_i + C_i, 3T - C_j]$. If $T - C_i - C_j < t_i < C_j$, there are two time- intervals for t_{j1}. They are $[t_i + C_i, T + t_i - C_j]$ and $[T + t_i + C_i, 2T + t_i - C_j]$. If $C_j \le t_i \le T - C_i$, there are three time-intervals for t_{j1}. They are $[0, t_i - C_j]$, $[t_i + C_i, T + t_i - C_j]$ and $[T + t_i + C_i, 2T + t_i - C_j]$. If $T - C_i < t_i < T$, there are three time-intervals for t_{j1}. They are $[-T + t_i + C_i, t_i - C_j]$, $[t_i + C_i, T + t_i - C_j]$ and $[T + t_i + C_i, 2T + t_i - C_j]$. If $T \le t_i \le 2T - C_i - C_j$, there are three time-intervals for t_{j1}. They are $[-T + t_i + C_i, t_i - C_j]$, $[t_i + C_i, T + t_i - C_j]$ and $[T + t_i + C_i, 3T - C_j]$. If $2T - C_i - C_j < t_i < T + C_j$, there are two time-intervals for t_{j1}. They are $[-T + t_i + C_i, t_i - C_j]$ and $[t_i + C_i, T + t_i - C_j]$. If $T + C_j \le t_i \le 2T - C_i$, there are three time-intervals for t_{j1}. They are $[0, t_i - T - C_j]$, $[t_i + C_i - T, t_i - C_j]$ and $[t_i + C_i, T + t_i - C_j]$.

Under the premise that $2C_j + C_i \le T$, the analysis is similar to the above process. Therefore, all the available time-intervals for t_{j1} can be summarized as follows.

a) $2C_j + C_i \le T$

$$t_{j1} \in \begin{cases} [(a-n)T + t_i + C_i, (a+1-n)T + t_i - C_j] \\ \quad \cup [(m_j - 1 - n)T + t_i + C_i, m_jT - C_j] \\ \quad (a = 0, 1 \cdots m_j - 2) \quad nT \le t_i < nT + C_j \\ \quad (n = 0, 1 \cdots m_i - 1) \\ [0, t_i - nT - C_j] \cup [(a-n)T + t_i + C_i, \\ \quad (a+1-n)T + t_i - C_j] \cup [(m_j - 1 - n)T + t_i \\ \quad + C_i, m_jT - C_j] \quad (a = 0, 1 \cdots m_j - 2) \\ \quad nT + C_j \le t_i \le (n+1)T - C_i - C_j \\ \quad (n = 0, 1 \cdots m_i - 1) \\ [0, t_i - nT - C_j] \cup [(a-n)T + t_i + C_i, \\ \quad (a+1-n)T + t_i - C_j] \quad (a = 0, 1 \cdots m_j - 2) \\ \quad (n+1)T - C_i - C_j < t_i \le (n+1)T - C_i \\ \quad (n = 0, 1 \cdots m_i - 1) \\ [(a-n)T + t_i + C_i, (a+1-n)T + t_i - C_j] \\ \quad (a = -1, 0, 1 \cdots m_j - 2) \quad (n+1)T - C_i < t_i < (n+1)T \\ \quad (n = 0, 1 \cdots m_i - 2) \end{cases} \tag{21}$$

b) $C_j + C_i \le T < 2C_j + C_i$

$$t_{j1} \in \begin{cases} [(a-n)T + t_i + C_i, (a+1-n)T + t_i - C_j] \\ \quad \cup [(m_j - 1 - n)T + t_i + C_i, m_jT - C_j] \\ \quad (a = 0, 1 \cdots m_j - 2) \quad nT \le t_i \le (n+1)T - C_i - C_j \\ \quad (n = 0, 1 \cdots m_i - 1) \\ [(a-n)T + t_i + C_i, (a+1-n)T + t_i - C_j] \\ \quad (a = 0, 1 \cdots m_j - 2) \quad (n+1)T - C_i - C_j \\ \quad < t_i < nT + C_j \quad (n = 0, 1 \cdots m_i - 1) \\ [0, t_i - nT - C_j] \cup [(a-n)T + t_i + C_i, \\ \quad (a+1-n)T + t_i - C_j] \quad (a = 0, 1 \cdots m_j - 2) \\ \quad nT + C_j \le t_i \le (n+1)T - C_i \quad (n = 0, 1 \cdots m_i - 1) \\ [(a-n)T + t_i + C_i, (a+1-n)T + t_i - C_j] \\ \quad (a = -1, 0, 1 \cdots m_j - 2) \quad (n+1)T - C_i \\ \quad < t_i < (n+1)T \quad (n = 0, 1 \cdots m_i - 2) \end{cases} \tag{22}$$

c) $C_j + C_i > T$

$$t_{j1} \in \emptyset \tag{23}$$

• The analysis of t_{j2}

Assume the first release time of both partitions P_i and P_j are zero. All the release time points of the two partitions in $MTF' = LCM(m_iT, m_jT)$ are as following.

$$\begin{array}{lllllll} P_i & 0, & m_iT, & 2m_iT, & \cdots & (m_j-1)m_iT, & m_jm_iT \\ P_j & \multicolumn{5}{l}{\underbrace{0, \quad m_jT, \quad \cdots \quad (m_i-1)m_jT,}_{MTF'}} & m_im_jT \end{array}$$

The number of releases for partition P_j is m_i. For each $qm_jT(q = 0, 1, 2 \cdots m_i - 1)$, there exists a $pm_iT(p = 0, 1, 2 \cdots m_j - 1)$ making $0 \le pm_iT - qm_jT < m_iT$.

Take $m_i = 5$ and $m_j = 7$ as an example. All the release time points of the two partitions in $MTF' = LCM(5T, 7T) = 35T$ are as following.

$$\begin{array}{lllllllll} P_i & 0, & 5T, & 10T, & 15T, & 20T, & 25T, & 30T, & 35T \\ P_j & \multicolumn{7}{l}{\underbrace{0, \quad 7T, \quad 14T, \quad 21T, \quad 28T,}_{MTF'}} & 35T \end{array}$$

For each one in $\{0, 7T, 14T, 21T, 28T\}$, there exists an appropriate value in $\{0, 5T, 10T, 15T, 20T, 25T, 30T\}$ which makes $0 \le s = 5pT - 7qT < 5T$ $(p \in \{0, 1, 2, 3, 4, 5, 6\}; q \in \{0, 1, 2, 3, 4\})$. s denotes the difference of $5pT - 7qT$ and $s = rT(r = 0, 1, 2, 3, 4)$. It is remarkable that each q corresponds to a different r because of the equal numbers of q and r. In fact, each $pm_iT - qm_jT(q = 0, 1, 2 \cdots m_i - 1)$ will be equal to a different one of $s = rT$ $(r = 0, 1, 2 \cdots m_i - 1)$ for any two coprime numbers m_i and m_j. The proof is as following.

Assume there exist two equal $pm_iT - qm_jT(q = 0, 1, 2 \cdots m_i - 1)$, namely

$$\begin{aligned} & p_{n1}m_iT - q_{n1}m_jT = p_{n2}m_iT - q_{n2}m_jT \quad (n1 \ne n2) \\ & \Rightarrow p_{n1}m_iT - p_{n2}m_iT = q_{n1}m_jT - q_{n2}m_jT \quad (n1 \ne n2) \\ & \Rightarrow (p_{n1} - p_{n2})m_iT = (q_{n1} - q_{n2})m_jT \quad (n1 \ne n2) \end{aligned} \tag{24}$$

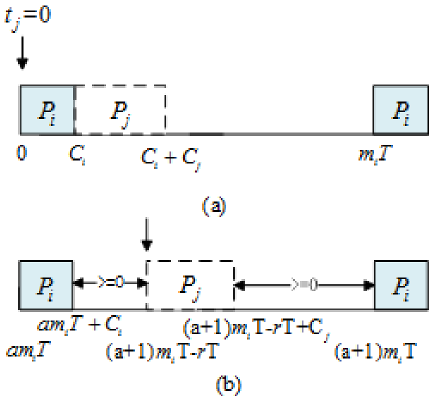

FIGURE 10. The necessary and sufficient condition of the existence of t_{j2}.

However, $p_{n1} - p_{n2} < m_j$; $q_{n1} - q_{n2} < m_i$ because of $p = 0, 1, 2 \cdots m_j - 1$; $q = 0, 1, 2 \cdots m_i - 1$. The Eq. (24) is in contradiction with the premise that m_i and m_j are coprime. Therefore, arbitrary two $pm_iT - qm_jT(q = 0, 1, 2 \cdots m_i - 1)$ are not equal when q is different, and each $pm_iT - qm_jT(q = 0, 1, 2 \cdots m_i - 1)$ can be equal to a different one of $s = rT$ $(r = 0, 1, 2 \cdots m_i - 1)$.

If the time that the two different partitions are released simultaneously is not zero, it means that all the release time points of the two partitions in $MTF' = LCM(m_iT, m_jT)$ are as following.

$$\begin{array}{ll} P_i & t_i,\ \ t_i + m_iT, \cdots t_i + (m_j - 1)m_iT,\ \ t_i + m_jm_iT \\ P_j & \underbrace{t_j,\ \ t_j + m_jT, \cdots t_j + (m_i - 1)m_jT,}_{MTF'}\ \ t_j + m_jm_iT \end{array}$$

$s = t_i + pm_iT - (t_j + qm_jT)$ will equal to mod$[(r + a)T/m_i](r = 0, 1 \cdots m_i - 1; a = \{0, 1 \cdots m_i - 1\})$, which is always equal to $s = rT(r = 0, 1, 2 \cdots m_i - 1)$ no matter what a is.

Therefore, we can obtain the necessary and sufficient condition of the existence of t_{j2} using the case that the first release time of the two partitions P_i and P_j are both zero.

As Fig. 10(a) shows, Eq. (25) will be obtained when $pm_iT - qm_jT = 0$.

$$C_i + C_j \leq m_iT \tag{25}$$

As Fig. 10(b) shows, Eq. (26) will be obtained when $pm_iT - qm_jT = rT(r = 1, 2 \cdots m_i - 1)$.

$$\begin{cases} C_i \leq (m_i - r)T \\ C_j \leq rT \end{cases} \tag{26}$$

Therefore, the necessary and sufficient condition of the existence of t_{j2} is the intersection of Eq. (25) and Eq. (26), which is $C_m \leq T$ $(m = i, j)$. The available t_{j2} can be obtained based on the following formula when the condition is satisfied.

$$t_{j2} = \{t_{j2} \big| t_i + n_im_iT = t_{j2} + n_jm_jT,\ n_i = 0, 1, 2 \cdots m_j - 1; \\ n_j = 0, 1 \cdots m_i - 1;\ 0 \leq t_{j2} \leq m_jT - C_j\} \tag{27}$$

$$\Rightarrow 0 \leq t_{j2} = t_i + n_im_iT - n_jm_jT \leq m_jT - C_j \tag{28}$$

Assume that t_i is equal to 0, and $0 \leq t_{j2} = n_im_iT - n_jm_jT \leq (m_j - 1)T$ has the same solution with Eq. (28). We can obtain $t_{j2} = bT(b = 0, 1, 2 \cdots m_i - 1)$ when $t_i = 0$. Therefore, the general t_{j2} can be calculated as following.

$$t_{j2} = t_i + bT \begin{cases} b = 0, 1 \cdots m_j - 1 \quad t_i \in [0, T - C_j] \\ b = 0, 1 \cdots m_j - 1 - n \\ \quad t_i \in [nT - C_j, (n+1)T - C_j] \\ \quad (n = 1, 2 \cdots m_i - 2) \\ b = 0, 1 \cdots m_j - 1 - m_i - 1 \\ \quad t_i \in [(m_i - 1)T - C_j, m_iT - C_i - C_j] \\ b = 0, 1 \cdots m_j - 1 - m_i \\ \quad t_i \in [m_iT - C_i - C_j, m_iT - C_i] \end{cases} \tag{29}$$

Therefore, t_j can be obtained by using Eq. (20).

d: m_i AND m_j HAVE COMMON FACTOR GREATER THAN ONE AND m_i IS NOT A DIVISOR OF m_j

Assume that R is the greatest common divisor of m_i and m_j. m_i/R and m_j/R are coprime. Therefore, the analysis is the same as the case that m_i and m_j are coprime when T is replaced by RT, m_i is replaced by m_i/R and m_j is replaced by m_j/R.

Therefore, the schedulability of the arbitrary two partitions can be obtained based on the above analysis.

3) INTERRUPTION ANALYSIS FOR ARBITRARY PARTITION SET

Since, $MTF' = LCM(m_iT, m_jT)$ is a divisor of $MTF = LCM(m_1T, m_2T, m_3T, \cdots, m_nT)$, the two partitions P_i and P_j in a partition set $\Pi = \{P_1, P_2, P_3 \cdots P_n\}$ can be scheduled without interruptions in MTF if they are schedulable without interruptions in MTF'. Moreover, the runtime operation in MTF will periodically repeat for the two partitions P_i and P_j and the repetition period is exactly MTF'.

For an arbitrary partition set $\Pi = \{P_1, P_2, P_3 \cdots P_n\}$, we first sort the partitions in ascending order of periods. Then, we set the first release time of partition P'_1 in the ordered partition set $\Pi' = \{P'_1, P'_2, P'_3 \cdots P'_n\}$ as zero, and analyze the schedulability without interruptions for P'_1 and P'_2. If they are non-schedulable, other partitions do not need to be considered, and we can derive that the partition set is non-schedulable without interruptions. Otherwise, we conduct the available time analysis for partition P'_3 with partition $P'_i(i < 3)$ based on the available first release time of P'_2 obtained in the previous analysis. For example, if the available first release time of P'_2 is t_2, we first analyze the schedulability of P'_3 and P'_1, and we can obtain the available t'_3 if they are schedulable. Then we analyze the schedulability of P'_3 and P'_2 based on t_2. Assume that the available time for P'_3 exists and can be denoted by t''_3. If the intersection of t'_3 and t''_3 is nonempty, we can derive that P'_1, P'_2, P'_3 is schedulable without interruptions. In addition, we can also obtain the available time for P'_2 and P'_3. Similar to the above process, we can analyze more partitions one by one until all partitions

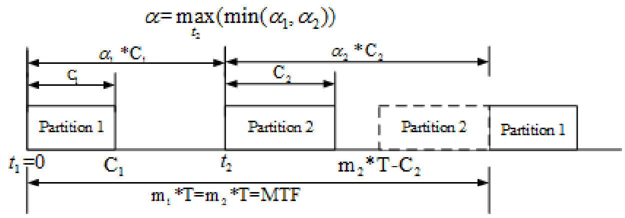

FIGURE 11. The impact of α.

are considered. The partition set $\Pi = \{P_1, P_2, P_3 \cdots P_n\}$ can be scheduled without interruptions if the intersection of all available time is nonempty.

The above analysis process will be terminated once the partition set can be determined to be non-schedulable. However, every two partitions will be analyzed if the partition set is schedulable without interruptions. Therefore, the time complexity of the analysis is $O(n^2)$ in the worst situation.

B. OPTIMIZATION STRATEGY FOR THE PARTITION SCHEDULING PROBLEM

1) OPTIMIZATION SCHEDULING STRATEGY FOR SCHEDULABLE PARTITIONS WITHOUT INTERRUPTIONS

For the schedulable partitions without interruptions, there will be many *FRTs* that can make the number of interruptions equal to zero, Therefore, the two optimization goals proposed in section II.B cannot be used to determine the best *FRT*. Sheikh *et al.* [12] use a coefficient α to evaluate *FRT*. The maximum α determines the maximum idle time that is allocated to each partition as much as possible based on their execution time. The impact of α is shown in Fig. 11.

Assume there are two partitions and their periods are the same, namely $m_1T = m_2T$. Therefore, $MTF = LCM(m_1T, m_2T) = m_1T = m_2T$. If $t_1 = 0, t_2 \in [C_1, m_2T - C_2]$. Based on the definition, we can obtain the following equations.

$$\alpha_1 = t_2/C_1 \tag{30}$$

$$\alpha_2 = (MTF - t_2)/C_2 \tag{31}$$

$$\alpha = \max_{t_2}(\min(\alpha_1, \alpha_2)) \tag{32}$$

α will reach the maximum value when t_2 make $\alpha_1 = \alpha_2$, and the idle time is fairly allocated to the two partitions based on their execution time.

Sheikh *et al.* [12] also proposed a best-response algorithm inspired by the non-cooperative game theory to optimize α. The algorithm is effective to dispatch the schedulable partitions without interruptions.

2) OPTIMIZATION SCHEDULING STRATEGY FOR NON-SCHEDULABLE PARTITIONS WITHOUT INTERRUPTIONS

Many optimization algorithms, like genetic algorithm (GA) [25], [26], tabu search (TS) [27], [28], neural networks (NN) [29], particle swarm optimization (PSO), can be used in our optimization framework to optimize the partition sets which are not schedulable without interruptions. Among the many optimization algorithms, PSO has been widely used to solve various optimization problems because of its simplicity and fast convergence. In this paper, we use PSO in the optimization framework to optimize the first release time of all partitions. We improve the update strategy for the particles beyond search space and round all particles before calculating the fitness value in PSO to optimize *FRT* for searching the minimum *NI* and *SET* in partition scheduling.

a: THE ENCODING AND THE INITIALIZATION OF THE PARTICLES

For a partition set $\Pi = \{P_1, P_2, P_3 \cdots P_n\}$, the first release time of the partition with the minimum period is zero. Therefore, the optimization of *FRT* is $(n-1)$ dimensional. The value of partition P_j's first release time is an arbitrary real number between 0 and $m_jT - C_j$. As a result, each particles' position x_i can be indicated by $[t_1, t_2, t_3 \cdots t_{n-1}]$ when partition P_n has the minimum period and the range of t_j is $[0, m_jT - C_j]$. Based on the number of partitions, we choose different numbers of particles constituted by candidate solutions. In addition, the initial position $x_{*j}(0)$ and velocity $v_{*j}(0)$ of the *j-th* dimension for all particles are chosen randomly from $[0, m_jT - C_j]$. Each particle updates velocity and position based on its own previous experience and the swarm's experience. The update formulas and the parameters are the same as standard PSO [21]–[23], [30], [31].

b: THE PROCESS OF PARTICLES BEYOND SEARCH SPACE

To prevent the particles from flying beyond the boundary and obtaining invalid candidate solutions, the appropriate limits for the velocity and the position need to be set. The maximum velocity is chosen as following.

$$v_{*jmax} = (m_jT - C_j)/2 \tag{33}$$

Here, v_{*jmax} is the maximum velocity of the *j-th* dimension of all particles, If $v_{*j}(t)$ is greater than v_{*jmax}, set $v_{*j}(t)$ as v_{*jmax}. If $v_{*j}(t)$ is less than $-v_{*jmax}$, set $v_{*j}(t)$ as $-v_{*jmax}$. For the positions of the particles, the maximum x_{*jmax} is $(m_jT - C_j)$, and the minimum x_{*jmin} is zero. If x_{*j} is beyond its interval in any dimension, the position for this particle will be chosen randomly from all candidate solutions.

c: THE ROUNDING FOR CANDIDATE SOLUTIONS

In general, the case in which some partitions are released simultaneously can decrease the number of interruptions compared to the case in which all partitions are released at different time based on the definition of the interruption. However, if the updated positions are directly used to calculate the two fitness values, the influence caused by the fractional part will make the simultaneous release almost impossible. For example, assume that there are three partitions in a partition set, and the first release time is [0, 5.1, 10.7]. The release time of the partitions P_2 and P_3 will never be simultaneous if the three partitions' periods are integers. The experiment results in section IV.B also show that rounding is at least an effective measure to obtain better candidate solutions with the partitions which are released at the same time.

d: THE CALCULATION OF FITNESS VALUES

The runtime operation of all partitions in MTF needs to calculate NI and SET. Here, we take the partition set shown in Table 1 with FRT_1 as an example to show the analysis method of the runtime operation of all partitions. In Table 1, the period of P_1 is the smallest one among the three partitions, which means the first release time of P_1 is zero. The execution time of P_1 is 5ms, and there are no other releases in the time-interval $[0, 5ms]$. Therefore, MTF allocates $[0, 5ms]$ to partition P_1. Similar to the above process, $[5ms, 11ms]$ will be allocated to partition P_2. The closest release for all partitions is at 12ms, and there are no partitions, which are interrupted before, waiting for execution in $[11ms, 12ms]$. Therefore, $[11ms, 12ms]$ will be the idle time without execution for all partitions. Similar to the above process, $[12ms, 19ms]$ and $[20ms, 25ms]$ will be allocated to partitions P_3 and P_1, respectively. The second execution of partition P_2 begins at 35ms. However, the third release of partition P_1 is at 40ms. Based on the execution rules, the MTF will allocate the time window to the new coming partition. Therefore, partition P_1 will interrupt the execution of partition P_2 at 40ms. The third execution of partition P_1 will be finished at 45ms, when the second execution of partition P_2 will continue for 1ms. Therefore, MTF will allocate $[35ms, 40ms]$ and $[45ms, 46ms]$ to partition P_2 and $[40ms, 45ms]$ to partition P_1. Similar to the above process, we can obtain the runtime operation of all partitions in MTF.

Therefore, NI and SET for the example shown in Table 1 with FRT_1 can be calculated by Eq. (4) and Eq. (6).

Based on the above analysis, the algorithm to calculate NI and SET for an arbitrary partition set $\Pi = \{P_1, P_2, P_3 \cdots P_n\}$ with the arbitrary $FRT[t_1, t_2, t_3 \cdots t_n]$ is shown as the pseudo code in algorithm 1. In step 5, RT means the release time points for each partition. In step 9, If some release time points for different partitions are equal, sort them in descending order of corresponding periods. In step 10, WET saves the time that all interrupted partitions need to finish their own complete execution. For example, if partition P_i with $C_i = 5ms$ is interrupted by other partition's release when P_i has been executed exactly for 3ms, the corresponding position of WET will be $5 - 3 = 2ms$. If a partition is not interrupted, the corresponding position of WET is zero.

Algorithm 1 Calculate NI and SET for Arbitrary Partition Set With Arbitrary $FRT = [t_1, t_2, t_3 \cdots t_n]$

Input: $[t_1, t_2, t_3 \cdots t_n]$, $[C_1, C_2, C_3 \cdots C_n]$ and $[m_1T, m_2T, m_3T \cdots m_nT]$
Output: NI and SET
1: $MTF = LCM(m_1T, m_2T, m_3T \cdots m_nT)$
2: **for** $i = 1 \rightarrow n$ **do**
3: $N_i = MTF/m_iT$
4: **for** $j = 1 \rightarrow N_i$ **do**
5: $RT[ij] = t_i + (j-1) * m_iT$
6: **end for**
7: **end for**
8: $SN = \sum_{i=1}^{n} N_i$
9: $RT = SortAscending(RT_{11} \cdots RT_{1N_1} \cdots RT_{nN_n})$
10: $WET = [0, 0, 0 \cdots 0]$
11: **for** $i = 1 \rightarrow SN$ **do**
12: $s =$ the first number of the subscript for $RT[i]$
13: **if** $RT[i+1] - RT[i] <= C_s$ **then**
14: Assign time window $[RT[i], RT[i+1]]$ to partition s
15: $WET[s] = RT[i+1] - RT[i] - C_s$
16: **else**
17: Assign time window $[RT[i], RT[i+1]]$ to partition s
18: Sort WET in ascending order of each partition's next release time
19: Execute the corresponding partitions in the order WET in $RT[i+1] - RT[i] - C_s$
20: update the corresponding WET
21: **end if**
22: **end for**
23: $NI = STW - SN$
24: $SET = SAC + \sum_{i=1}^{n} k_iC_i$

IV. EXPERIMENT AND ANALYSIS

In this section, we first introduce the simulation environment that we have developed. Using the simulation environment, we conduct the experiments to show the function of rounding all candidate solutions in each iteration. Then, we conduct the scheduling experiments for different partition sets which cannot be scheduled without interruptions based on our proposed scheduling model, and give the scheduling results and the comparisons with other models. In addition, we also analyze the properties of the optimization problem and show the rationality of the improved PSO. Finally, we summarize the experiments and conclusions.

A. SIMULATION ENVIRONMENT

IMA runs in the embedded operation system named VxWorks, which abides by the basic software framework of the industry standard ARINC 653. However, the scheduling model and the algorithm are the underlying work of the embedded development platform. As a result, they are generally not open as universal interfaces and we cannot load our proposed model and algorithm in ARINC653 directly.

For the construction of the simulation environment, a software language named Architecture Analysis and Design Language (AADL) [32] is popular in the field of embedded systems. However, it also does not have the open interface or software development kit (SDK) available. The scheduling models it can simulate are limited, and it cannot achieve the schedulability analysis and optimization of the model and the algorithm.

TABLE 2. Optimal solutions with rounding for the partition set in Table 1.

No.	NI	SET/ms	t_1/ms	t_2/ms	t_3/ms
1	1	81	0	10	20
2	1	81	0	10	25
3	1	81	0	10	5
4	1	81	0	20	25
5	1	81	0	20	20
6	1	81	0	10	25
7	1	81	0	10	20
8	1	81	0	10	25
9	1	81	0	10	25
10	1	81	0	10	20

TABLE 3. Optimal solutions without rounding for the partition set in Table 1.

No.	NI	SET/ms	t_1/ms	t_2/ms	t_3/ms
1	3	92	0	15.4533	27.362
2	3	92	0	16.3721	27.385
3	3	92	0	6.30861	28.115
4	3	92	0	15.6081	27.138
5	3	92	0	5.14822	7.0414
6	3	92	0	15.911	8.7173
7	3	92	0	6.7329	28.948
8	3	92	0	15.9489	27.626
9	3	92	0	6.64520	28.891
10	3	92	0	15.1872	27.6111

In order to verify our model and optimization framework, we construct a simple simulation platform based on MATLAB environment and MySQL database ourselves. Our simulation platform has a more convenient interface to support the addition of the running rules of the scheduling model. First, it calculates the execution time-windows of all partitions by simulating the system operation. Then, it can further calculate the schedulability of the system and the fitness values of the candidate solutions. Besides, the optimization algorithm is an independent module and has a common interface in the platform. We can embed any improved algorithms based on the primary platform. Overall, our simulation platform is an integrated system that supports model addition, algorithm optimization and the graphical display of simulation results. We can use it to verify the validity of the model, calculate the schedulability of the system and analyze the performance of the algorithm. For the following experiments, the simulation platform runs in a computer with Intel(R) Core(TM) i5-4590T CPU and 4GB RAM.

B. THE FUNCTION OF ROUNDING

We use the partition set shown in Table 1 to demonstrate the function of rounding. The number of particles is 50 and the number of iterations is 200. The optimization results with rounding the positions in PSO are shown in Table 2, while the optimization results without rounding is shown in Table 3. A total number of 10 runs for each case are conducted.

There exist more than one optimal solutions with the same *NI* and *SET* whether we round the candidate solutions or not.

TABLE 4. All release time points of all partitions in Table 1 when *FRT* = [0, 10, 25].

Partition	All release time points					
P_1	0	20	40	60	80	100
P_2	10	40	70	100		
P_3	25	65	105			

TABLE 5. All release time points of all partitions in Table 1 when *FRT* = [0, 15, 27].

Partition	All release time points					
P_1	0	20	40	60	80	100
P_2	15	45	75	105		
P_3	27	67	107			

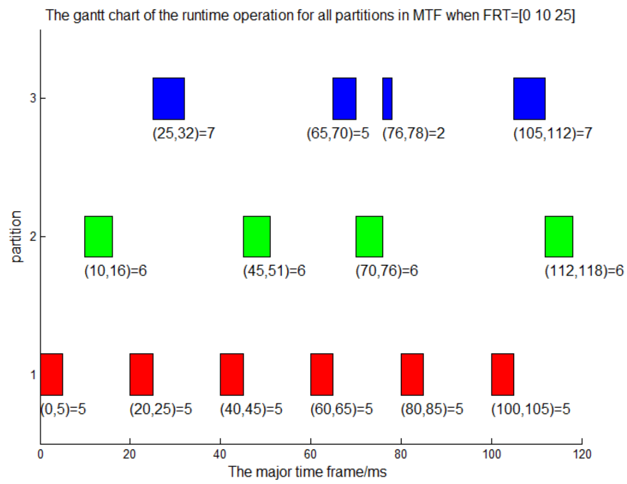

FIGURE 12. The gantt chart of the runtime operation for all partitions with *FRT* = [0, 10, 25].

For each optimal solution, the candidate solutions in its small neighborhood are worse than it when we round the candidate solutions in each iteration. However, there may exists a small neighborhood with the same fitness values as the optimal solution when we do not execute rounding. For example, $NI \neq 1$, $SET \neq 81$ when *FRT* equals to [0, 10.1, 20] and [0, 15, 27] can still make $NI = 3$, $SET = 92$. Based on the two fitness values, the scheduling results with rounding are much better than those without rounding. We use [0, 10, 25] and [0, 15, 27] to explain the reasons. As Table 4, Table 5, Fig. 12 and Fig. 13 show, the case that some partitions are released simultaneously in *MTF* can decrease *NI* and *SET*. Because of the fact that all the release time points of the partition with the minimum period are integers, rounding all candidate solutions is a good strategy to make more partitions released simultaneously.

The candidate solutions may still obtain real numbers along with the evolution of the particles when integer-number encoding is used, and they also need to be rounded. Therefore, we adopt the real-number encoding for *FRT* directly, and round the candidate solutions before calculating the fitness values.

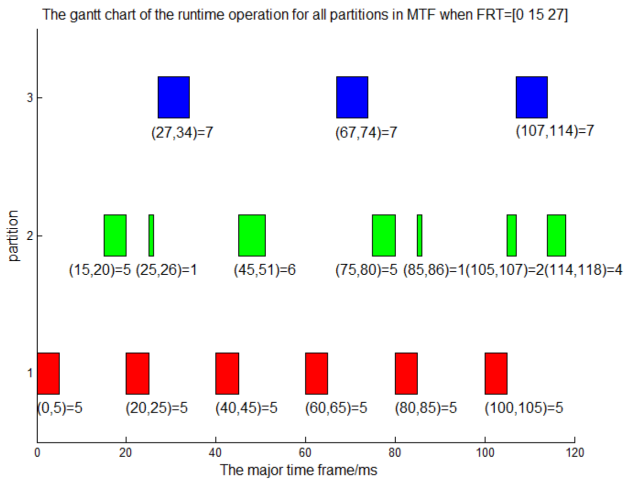

FIGURE 13. The gantt chart of the runtime operation for all partitions with *FRT* = [0, 15, 27].

TABLE 6. Partition parameters of the three partitions.

Partition	P_1	P_2	P_3
m_iT/ms	20	32	38
C_i/ms	5	7	8

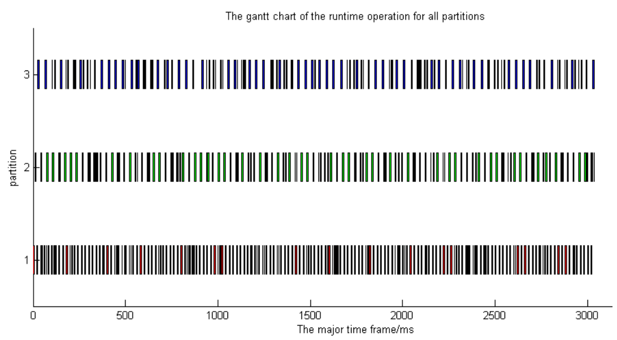

FIGURE 14. The gantt chart of the runtime operation in *MTF* for all three partitions in Table 6.

C. THE ANALYSIS OF THE PROPOSED MODEL BASED ON DIFFERENT SETS

1) AN EXAMPLE TO SHOW THE INFEASIBILITY OF COMPLEX PARTITION PERIODS

The complex partition periods, which means m_i and m_j are coprime, will cause the complicated operation of all partitions. It is more likely to cause a large number of interruptions and sharply reduce the schedulability of the system. Besides, the optimization will also be much more difficult. For the three partitions shown in Table 6, the optimization result is presented in Fig. 14. The complex periods of all partitions cause very complex operation for the IMA system. Therefore, the simple periods of all partitions are more reasonable when designing partition parameters. In fact, the periods of all partitions are even equal in the real IMA system.

TABLE 7. Partition parameters of the three partitions.

Partition	P_1	P_2	P_3
m_iT/ms	20	30	40
C_i/ms	5	8	9

TABLE 8. Partition parameters of the four partitions.

Partition	P_1	P_2	P_3	P_4
m_iT/ms	20	30	30	40
C_i/ms	3	5	6	7

TABLE 9. Partition parameters of the five partitions.

Partition	P_1	P_2	P_3	P_4	P_5
m_iT/ms	20	20	30	40	60
C_i/ms	4	5	4	6	10

TABLE 10. Optimal solutions obtained by the improved PSO for the partition set in Table 7.

No.	NI	SET/ms	t_1/ms	t_2/ms	t_3/ms
1	2	117	0	20	11
2	2	117	0	10	11
3	2	117	0	10	11
4	2	117	0	20	11
5	2	117	0	10	11
6	2	117	0	10	11
7	2	117	0	10	11
8	2	117	0	0	11
9	2	117	0	10	11
10	2	117	0	20	11

2) EFFECTIVENESS ANALYSIS OF ALGORITHM AND MODEL

The following three experiments with three, four, and five partitions are conducted to illustrate the effectiveness of our proposed partition scheduling model and the optimization framework. The partition parameters are shown in Table 7, Table 8 and Table 9. The numbers of the particles are 10, 50 and 50, respectively, for the three experiments, and the numbers of iterations are 50, 100 and 200, respectively.

Each experiment runs 10 times and the optimal solutions are shown in Table 10, Table 11 and Table 12. From Table 10, we can find that *NIs* and *SETs* of the 10 runs are the same, while *FRTs* are different. It means that there exist more than one solutions which can make *NI* equal to 2 and *SET* equal to 117ms. Therefore, *FRTs* of the 10 runs are equivalent. Taking [0, 10, 11] as an example, the scheduling gantt chart is shown in Fig. 15. From the results shown in Table 11, all *NIs* are equal to 1, while *SETs* are not the same. 70% of the 10 runs reach the optimal solutions searched by the improved PSO. The scheduling gantt chart using an optimal *FRT* which is equal to [0, 10, 20, 3] is shown in Fig. 16. In Table 12, The searched optimal solution is [0, 14, 10, 20, 24]. However, the ratio of obtaining the optimal solution in the ten runs is only 10%. The improved PSO has an obvious decline in performance along with the increasing of the particle's dimension. The scheduling gantt chart for the partition set in Table 9 with *FRT* = [0, 14, 10, 20, 24] is shown in Fig. 17.

TABLE 11. Optimal solutions obtained by the improved PSO for the partition set in Table 8.

No.	NI	SET/ms	t_1/ms	t_2/ms	t_3/ms	t_4/ms
1	1	92	0	4	14	4
2	1	86	0	10	20	3
3	1	86	0	10	0	3
4	1	86	0	10	20	3
5	1	92	0	14	24	4
6	1	86	0	0	20	23
7	1	86	0	10	20	3
8	1	86	0	10	20	3
9	1	92	0	4	14	24
10	1	86	0	10	20	3

TABLE 12. Optimal solutions obtained by the improved PSO for the partition set in Table 9.

No.	NI	SET/ms	t_1/ms	t_2/ms	t_3/ms	t_4/ms	t_5/ms
1	1	127	0	15	5	9	20
2	1	117	0	4	10	14	10
3	1	127	0	4	10	14	20
4	2	118	0	5	20	5	10
5	2	118	0	5	20	5	10
6	1	117	0	4	10	14	14
7	1	121	0	0	25	29	0
8	1	127	0	4	20	14	4
9	1	127	0	4	20	14	4
10	1	116	0	14	10	20	24

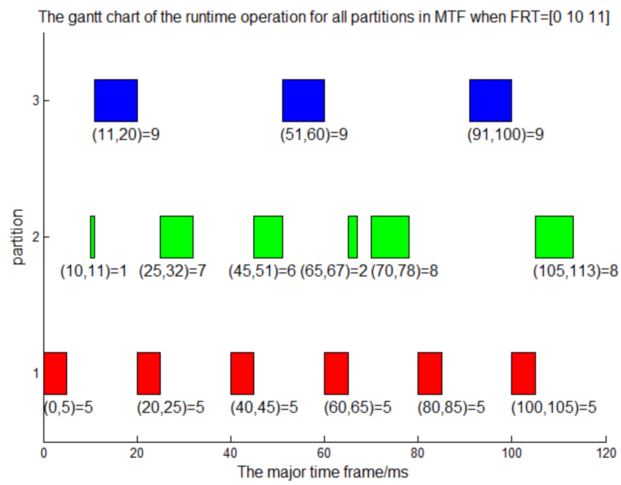

FIGURE 15. The gantt chart of the runtime operation for all partitions with $FRT = [0, 10, 11]$.

In order to prove that our improved PSO used in the framework can obtain at least a sub-optimal solution, we search all the integer solutions in each experiment. The comparison results of the solutions obtained by our improved PSO and traversal search for the partition sets in Table 7, Table 8 and Table 9 are shown in Table 13.

From Table 13, the best one of the optimization results of our improved PSO for all three experiments is the same with the corresponding result obtained by the traversal search. It means that the improved PSO can obtain the optimal solution within the integer range for the three partition sets. Therefore, the improved PSO can at least obtain the sub-optimal solutions.

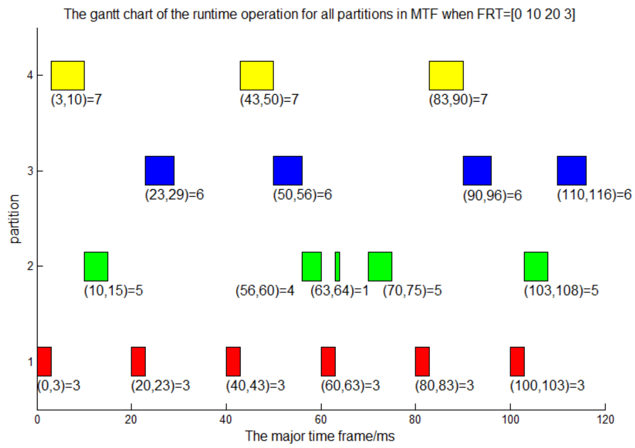

FIGURE 16. The gantt chart of the runtime operation for all partitions with $FRT = [0, 10, 20, 3]$.

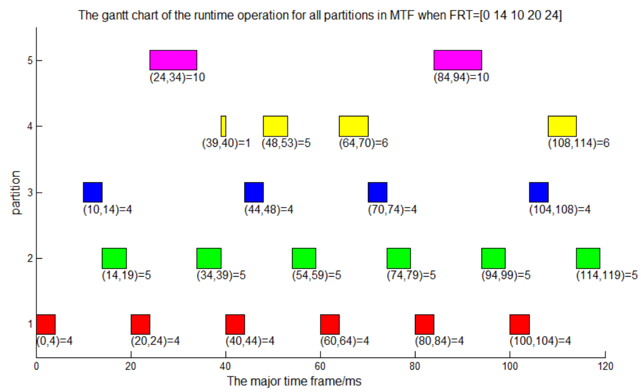

FIGURE 17. The gantt chart of the runtime operation for all partitions with $FRT = [0, 14, 10, 20, 24]$.

TABLE 13. The optimization results obtained by improved PSO and traversal search.

Process set	Our improved PSO (the best one of 10 runs)		Traversal search	
	NI	SET	NI	SET
Table 7	2	117	2	117
Table 8	1	86	1	86
Table 9	1	116	1	116

3) THE COMPARISON OF OUR MODEL AND THE TWO EXISTING MODELS WE COMBINED

The above three partition sets shown in Table 7, Table 8 and Table 9 will be non-schedulable based on the model proposed by Sheikh *et al.* [12], which is combined in our model to deal with the schedulable partition sets without interruptions. Therefore, our combined model retains the reliability for the schedulable partition sets without interruptions, and extends the schedulability for the partition sets which cannot be scheduled without interruptions because of the rationality of the interruptions in the model we proposed. In order to make a comparison with the model proposed by Gui *et al.* [15], we do not optimize *FRT* and choose *FRTs* randomly for the

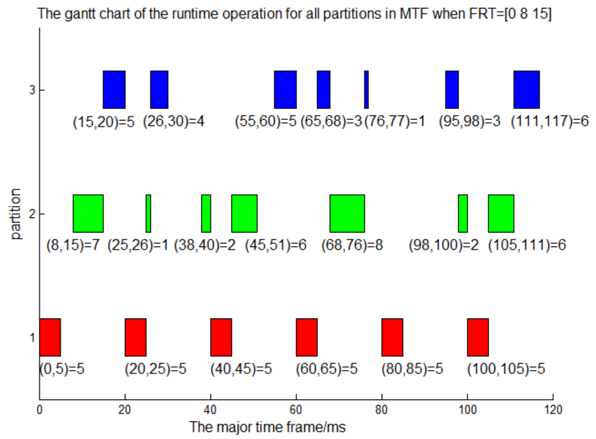

FIGURE 18. The gantt chart of the runtime operation for all partitions with $FRT = [0, 8, 15]$.

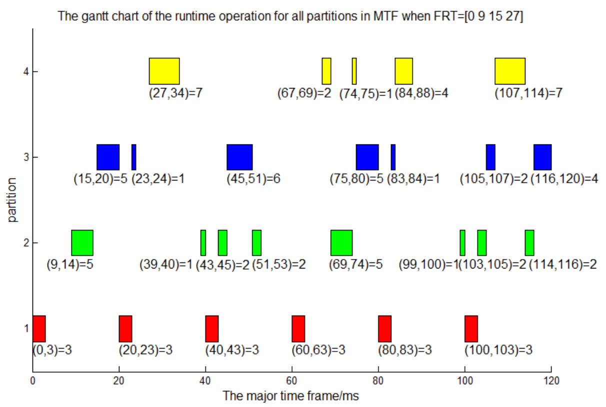

FIGURE 19. The gantt chart of the runtime operation for all partitions with $FRT = [0, 9, 15, 27]$.

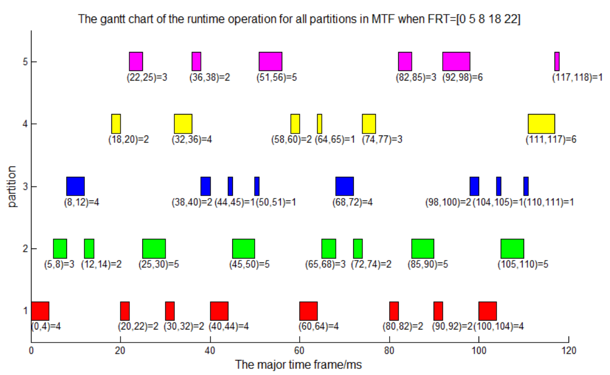

FIGURE 20. The gantt chart of the runtime operation for all partitions with $FRT = [0, 5, 8, 18, 22]$.

three partition sets, such as [0, 8, 15], [0, 9, 15, 27] and [0, 5, 8, 18, 22]. The scheduling gantt charts for them are shown in Fig. 18, Fig. 19 and Fig. 20.

The detailed comparison results for the three experiments are shown in Table 14. Sheikh *et al.*'s model cannot schedule the partition sets shown in Table 7, Table 8 and Table 9, and the results in Table 14 for it are empty. Compared with Gui *et al.*'s model, though our model has higher time cost, the smaller NIs, the smaller SETs and the simpler scheduling charts prove that our model is better. For the partition sets which are schedulable without interruptions, our scheduling model can retain the advantages of Sheikh *et al.*'s model. For the partition sets which are not schedulable without interruptions, our scheduling model makes significant improvement in the reliability of the processor's operation. Therefore, our model has a wider application scope for arbitrary partition sets.

TABLE 14. The comparison results for our model and the two existing models.

Process set	Our model			Sheikh et al.'s model			Gui et al.'s model		
	NI	SET	Time cost	NI	SET	Time cost	NI	SET	Time cost
Table7	2	117	0.58s	-	-	-	7	141	0.11s
Table8	1	86	6.49s	-	-	-	8	133	0.13s
Table9	1	116	13.78s	-	-	-	15	225	0.14s

D. ANALYSIS FOR THE PROPERTIES OF THE CANDIDATE SOLUTIONS

The partition set shown in Table 7 is used as an example to illustrate the properties of the search space for the partition scheduling problem. There are three partitions and the search space is two-dimensional. We traverse all integer candidate solutions. NI and SET of the optimal FRT are equal to 2 and 117ms, respectively. In order to combine NI and SET, we use Z to denote the new fitness value which can be calculated by the following formula.

$$Z(FRT) = -(NI_{(FRT)} \times W + SET_{(FRT)}) \tag{34}$$

Here, W is equal to 100. Through traversing all candidate solutions, the maximum and the minimum SET are equal to 167ms and 104ms, respectively, without considering NI. The difference is 63ms, which is less than 100ms. Therefore, the influence from SET will not change the dominant function of NI. In addition, we set NI as 9 and SET as 170ms for FRT which cannot make the partitions schedulable. They are greater than the maximum $NI = 8$ and the maximum $SET = 167ms$, respectively. The fitness values of FRTs which cannot make the partition set schedulable are less than that of FRTs which can make the partition set schedulable. Fig. 21 shows the fitness landscape of the partition set shown in Table 7 and Fig. 22 reflects the top view of Fig. 21.

From Fig. 21 and Fig. 22, we can obtain the following properties of the partition scheduling problem based on our model.

(1) There may exist more than one optimal solutions for the partition scheduling problem. However, the number of the optimal solutions is still very small compared with the search space. For the partition set in Table 7, there are four optimal solutions while all integer candidate solutions are $23 \times 32 = 736$. In addition, we also traverse all integer candidate solutions for the partition sets shown in Table 8 and Table 9. The optimal solutions are 10 and 4, respectively,

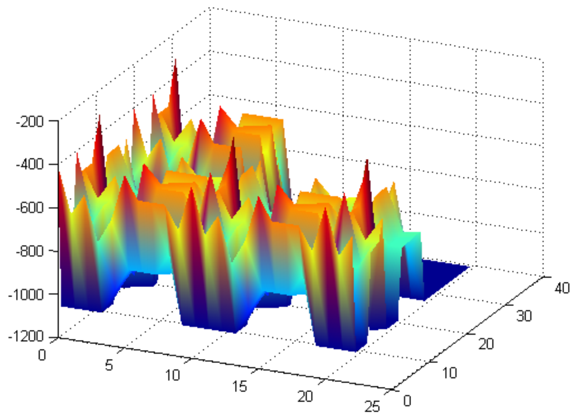

FIGURE 21. The fitness landscape of the partition set shown in Table 7.

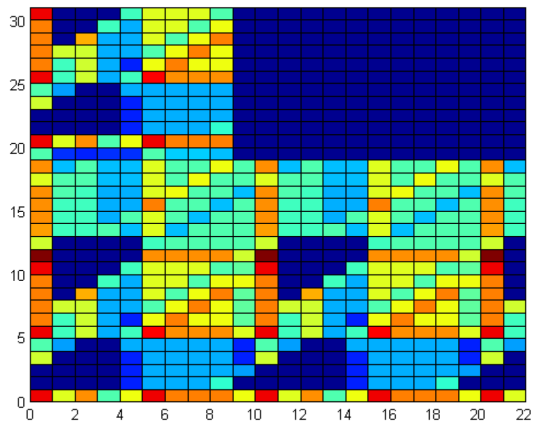

FIGURE 22. The top view of Fig. 21.

while all integer candidate solutions are $26 \times 25 \times 34 = 22100$ and $16 \times 27 \times 35 \times 51 = 771120$, respectively. Moreover, the optimal solutions have a decentralized distribution in the search space.

(2) The search space is very rough, and there are many local optimal solutions for the partition scheduling problem. Therefore, improving the diversity of the optimization algorithm is critical to searching the optimal solutions. The operation that updates the positions randomly for the particles beyond the search space is an effective measure to prevent premature termination for the improved PSO.

(3) The fitness landscape for the partition scheduling problem lacks gradient information of the neighborhood. This characteristic also can be seen from the definition of the interruption. If we just change a partition's first release time for a certain partition set, *NI* and *SET* for the scheduling scheme will not change unless the change for the partition's first release time goes beyond the critical point. Therefore, the traditional optimization algorithms, such as gradient descent algorithm, branch and bound algorithm, will be inappropriate for this problem. However, the meta-heuristic algorithms, like GA, PSO and TS, will be suitable to optimize the scheduling scheme.

E. SUMMARY OF THE EXPERIMENTS

Overall, the core idea of our model is that interruption is allowed but it should be avoided as much as possible. The model inherits the advantages of the two existing models. Compared with Sheikh *et al.*'s model [12], our model improves the schedulability. It also improves reliability compared with Gui *et al.*'s model [15]. From the scheduling results and the gantt charts, it can be concluded that our model can deal with arbitrary partition sets whether they are schedulable without interruptions or not, and give at least a near-optimal scheduling scheme. Therefore, our model is more effective than the existing models.

In addition, we also propose an optimization framework based on our model. We first analyze the schedulability for the partition sets without interruptions. For the partition set which cannot be scheduled without interruptions, we use improved PSO in the framework to show the complete optimization process and the details that need to be considered. In the improved PSO, the rounding for all positions of the particles is a critical process to search the optimal solutions, including the case in which different partitions can be released at the same time. In addition, the random assignment for the particles beyond the search space guarantees the fairness of all candidate solutions and improves the diversity of the optimization algorithm. From the scheduling charts shown in Fig. 15, Fig. 16 and Fig. 17, it is clear that the obtained solutions can achieve the scheduling goals of minimizing the number of interruptions and all partitions' run time when the partition sets cannot be scheduled without interruptions. Therefore, we can conclude that the optimization framework for our model is effective.

V. CONCLUSIONS

This paper focuses on the partition scheduling problem in IMA systems. Compared with the existing partition scheduling models, our proposed model retains the execution stability for the partition sets which can be scheduled without interruptions. In addition, our model increases the schedulability and ensures the execution stability as much as possible for the partition sets which can only be scheduled with interruptions. In the optimization framework, we first determine whether they are schedulable without interruptions for arbitrary partition sets. Furthermore, we use two different optimization strategies to obtain a good partition scheduling scheme based on the result of schedulability analysis. Therefore, the shedulability analysis is an essential contribution for the partition scheduling problem. The experiment results show that the solutions obtained by the improved PSO, which is used as an example in the framework, meet the goals of the model. In summary, the scheduling model that we proposed are more reasonable than the two existing models and the optimization framework that we proposed for our model is effective. In addition, other meta-heuristic algorithms can be embedded into the framework as the solution method.

Future work will focus on the scheduling model, scheduling algorithms and scheduling platform. For the scheduling model, more practical constraints, like the jitter uncertainty, should be considered in our proposed scheduling model. In addition, the partition scheduling model can be extended for multi-core processors. With regard to scheduling algorithms, other kinds of meta-heuristic algorithms can be considered based on our proposed solution platform and to find the best performance. For the scheduling platform, we have developed one based on Matlab and MySQL. The platform will be ported to the Web environment for increasing the generality and information sharing. Finally, the application of the proposed partition scheduling model and scheduling process in real IMA systems is a problem deserving further research.

REFERENCES

[1] C. B. Watkins and R. Walter, "Transitioning from federated avionics architectures to integrated modular avionics," in *Proc. IEEE/AIAA 26th Digit. Avionics Syst. Conf.*, Oct. 2007, pp. 2.A.1-1–2.A.1-10.

[2] C. B. Watkins, "Integrated modular avionics: Managing the allocation of shared intersystem resources," in *Proc. IEEE/AIAA 25TH Digit. Avionics Syst. Conf.*, Nov. 2006, pp. 1–12.

[3] H. Ju, S. Wang, and T. Zhao, "A modeling method of IMA dynamic reconfiguration based on AADL," *Proc. 1st Int. Conf. Rel. Syst. Eng. (ICRSE)*, Oct. 2015, pp. 1–5.

[4] C. Wilkinson, "IMA aircraft improvements," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 20, no. 9, pp. 11–17, Sep. 2005.

[5] W. Ruan and Z. Zhai, "Kernel-level design to support partitioning and hierarchical real-time scheduling of ARINC 653 for VxWorks," in *Proc. IEEE 12th Int. Conf. Depend., Auton. Secure Comput. (DASC)*, Aug. 2014, pp. 300–388.

[6] *Avionics Application Software Standard Interface*, document ARINC653 P1-2, Aeronautical Radio Inc., Mar. 2006, pp. 2–45.

[7] Y. Seo and H. S. Kim, "Partitioning strategy of flight software for the IMA system," in *Proc. IEEE/AIAA 34th Digit. Avionics Syst. Conf. (DASC)*, Sep. 2015, pp. 1–21.

[8] 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.

[9] M. L. Dertouzos and A. K. Mok, "Multiprocessor online scheduling of hard-real-time tasks," *IEEE Trans. Softw. Eng.*, vol. 15, no. 12, pp. 1497–1506, Dec. 1989.

[10] J. Xu, Y. Li, and Y. Meng, "Research and implementation of ARINC653 configuration tool based on AADL," in *Proc. IEEE Symp. Comput. Informat. (ISCI)*, Mar. 2011, pp. 479–483.

[11] G. Xiao, Z. Qu, and F. He, "Design and realization of IMA simulation platform based on CPCI bus using VxWorks653 RTOS," in *Proc. IEEE/AIAA 34th Digit. Avionics Syst. Conf. (DASC)*, Sep. 2015, pp. 10A1-1–10A1-8.

[12] A. A. Sheikh, O. Brunl, P. L. Hladik, and B. J. Prabhu, "Strictly periodic scheduling in IMA-based architectures," *Real-Time Syst.*, vol. 48, no. 4, pp. 359–386, Jul. 2012.

[13] Y.-H. Lee, D. Kim, M. Younis, J. Zhou, and J. McElroy, "Resource scheduling in dependable integrated modular avionics," in *Proc. Int. Conf. Depend. Syst. Netw. (DSN)*, 2000, pp. 14–23.

[14] X. Tao *et al.*, "Designing ARINC653 partition constrained scheduling for secure real time embedded avionics," in *Proc. IEEE 2nd Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)*, Nov. 2015, pp. 213–217.

[15] S.-L. Gui, L. Lou, S.-S. Tang, and Y. Meng, "Optimal static partition configuration in ARINC653 system," *J. Electron. Sci. Technol.*, vol. 9, no. 4, pp. 373–378, 2011.

[16] F. Eisenbrand, N. Hähnle, M. Niemeier, M. Skutella, J. Verschae, and A. Wiese, "Scheduling periodic tasks in a hard real-time environment," in *Proc. 37th Int. Colloq. Automata, Lang. Program.*, 2010, pp. 299–311.

[17] O. Kermia and Y. Sorel, "A rapid heuristic for scheduling non-preemptive dependent periodic tasks onto multiprocessor," in *Proc. ISCA Int. Conf. Parallel Distrib. Comput. Syst. (ISCAPDCS)*, 2007, pp. 1–6.

[18] M. Wan and S. Tian, "Research on schedulability of partition scheduling for IMA," in *Proc. 4th Int. Symp. Comput. Intell. Design (ISCID)*, 2011, pp. 322–325.

[19] J. Lehoczky, L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: Exact characterization and average case behavior," in *Proc. Real Time Syst. Symp.*, 1989, pp. 166–171.

[20] M. Marouf and Y. Sorel, "Scheduling non-preemptive hard real-time tasks with strict periods," in *Proc. IEEE Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2011, vol. 47. no. 10, pp. 1–8.

[21] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Hum. Sci. (MHS)*, 1995, pp. 39–43.

[22] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proc. Congr. Evol. Comput. (CEC)*, vol. 3. 1999, p. 1950.

[23] F. Marini and B. Walczak, "Particle swarm optimization (PSO). A tutorial," *Chemometrics Intell. Lab. Syst.*, vol. 149, pp. 153–165, Dec. 2015.

[24] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE ICNN*, vol. 4. Nov./Dec. 1995, pp. 1942–1948.

[25] G. R. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," in *Proc. IEEE World Congr. Comput. Intell., IEEE Int. Conf. Evol. Comput.*, May 1998, pp. 523–528.

[26] B. L. Miller, D. E. Goldberg, and G. Algorithms, "Selection schemes, and the varying effects of noise," *Evol. Comput.*, vol. 4, no. 2, pp. 113–131, 1996.

[27] F. Glover, "Tabu search—Part I," *ORSA J. Comput.*, vol. 1, no. 3, pp. 190–206, 1989.

[28] F. Glover, "Tabu search—Part II," *ORSA J. Comput.*, vol. 2, no. 1, pp. 4–32, 1990.

[29] C. Zhang and W. Xu, "Neural networks: Efficient implementations and applications," in *Proc. IEEE 12th Int. Conf. ASIC (ASICON)*, Oct. 2017, pp. 1029–1032.

[30] H.-H. Chen, G.-Q. Li, and H.-L. Liao, "A self-adaptive improved particle swarm optimization algorithm and its application in available transfer capability calculation," in *Proc. 5th Int. Conf. Natural Comput.*, vol. 3. 2009, pp. 200–205.

[31] Y. H. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Proc. Int. Conf. Evol. Program.*, 1998, vol. 1447. no. 25, pp. 591–600.

[32] P. H. Feiler, B. A. Lewis, and S. Vestal, "The SAE architecture analysis & design language (AADL) a standard for engineering performance critical systems," in *Proc. IEEE Conf. Comput. Aided Control Syst. Design, IEEE Int. Conf. Control Appl., IEEE Int. Symp. Intell. Control*, Oct. 2006, pp. 1206–1211.

HUI LU received the Ph.D. degree in navigation, guidance and control from Harbin Engineering University, Harbin, China, in 2004. She is currently a Professor with Beihang University, Beijing, China. Her research interests include information and communication system and intelligent optimization and practical application.

QIANLIN ZHOU received the B.Sc. degree from the School of Electronic and Information Engineering, Beihang University, Beijing, China, in 2015, where he is currently pursuing the master's degree. His main research areas include automatic test system and optimization.

ZONGMING FEI received the Ph.D. degree in computer science from the Georgia Institute of Technology, Atlanta, GA, USA, in 2000. He is currently a Professor with the University of Kentucky, Lexington, KY, USA. His research interests include networking protocols and architectures, multimedia networking, and smart grid communications.

RONGRONG ZHOU received the B.Sc. degree from the School of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2016, where she is currently pursuing the master's degree. Her main research areas include optimization algorithm design and application in automatic test system.