IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 27, 2020, accepted June 1, 2020, date of publication June 17, 2020, date of current version July 6, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3002972

Research on Partition Parameter Design Method for Integrated Modular Avionics Based on MOEA/D-ADV

HUAKUN CHEN, WEIGUO ZHANG, AND YONGXI LYU
School of Automation, Northwestern Polytechnical University, Xi'an 710072, China

Corresponding author: Huakun Chen (chenhuakun@mail.nwpu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61573286 and Grant 61374032, in part by the Aeronautical Science Foundation of China under Grant 20180753006, in part by the Fundamental Research Funds for the Central Universities 3102019ZDHKY07, and in part by the Shaanxi Province Key Laboratory of Flight Control and Simulation Technology.

ABSTRACT In the avionics industry, Integrated Modular Avionics (IMA) which introduces the concept of partition has been widely adopted for its isolating capability. However, the real-time performance of the IMA system mainly depends on the partition parameters. This leads to the question of how to design the partition parameters for satisfying the timing requirements of real-time applications. In this paper, the problem of partition parameter design for multiprocessors system is investigated. Firstly, the hierarchical scheduling strategy of IMA is analyzed, and a new schedulability analysis method is proposed to judge the schedulability of the partitions according to the partition period and execution time. Then, an approximation algorithm is developed to minimize the allocated bandwidth of the partitions while simultaneously guaranteeing tasks schedulability within the partitions. The harmonic period partitions, which are used as the constraint of partition parameter design, are realized by considering the scheduling mechanism of intra-partition and inter-partition. The total required bandwidth and the system overhead caused by partition scheduling are regarded as the optimization objective functions. Moreover, Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D) method is improved by applying the Adjustment for the Direction Vectors (ADV) algorithm. Constrained Dominance Principle (CDP) is embedded into the improved algorithm to solve the constrained optimization problem. Consequently, simulation results show that the presented algorithm can achieve better coverage and uniformity than the compared algorithms while obtaining the partition parameters, and the system overhead and total required bandwidth can also be reduced.

INDEX TERMS Integrated modular avionics, hierarchical scheduling, multi-objective optimization, decomposition, partition parameter design, constrained optimization.

I. INTRODUCTION

IMA is an integrated platform of advanced electronic equipment for modern combat aircraft to accomplish communications, navigation, confrontation, control, and other missions. It is a vital system to ensure the safety of aircraft flight and the correct execution of flight function. Under the concept of IMA, the original independent system is integrated into several configurable general processing modules. The specific system functions are achieved by the applications running in the general processing modules.

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

ARINC653 is a real-time operating system standard for IMA software prepared by the Airline's electronic engineering committee. It defines the behavior logic of the real-time operating system under IMA architecture and the interface specification provided for the application program. In the ARINC653, the real-time operating system splits hardware resources into resource partitions with respect to space(memory partitioning) and time(temporal partitioning). Applications are loaded on these resource partitions independent of time and space [1]. Spatiotemporal independence of resource partition can prevent the failure of one partition from spreading to other partitions, and the reliability of applications can be improved. Partition is the core of IMA

software. The fault-tolerant ability of the system is greatly enhanced by using partition management, which achieves time-sharing and partitioned execution of tasks through partition scheduling.

In order to meet the increasing computing requirements of avionics system and the limitation on the size, weight, and power (SWAP) of avionics system, IMA system is committed to integrating different critical levels of system functions on a unified shared resource platform. It means that partitions at different critical levels may be integrated on the same hardware resource. Therefore, IMA brings new problems and challenges to avionics system designers. In the integrated environment with highly shared resources, how to design partitions rationally to ensure the real-time and reliability of the whole system becomes an attractive research point.

A. PROBLEM STATEMENT

As a robust real-time system, IMA requires that all tasks in the system can complete the calculation within deadline time. Task timeout exception can lead to serious consequence. Therefore, the design of partition time scheduling is the core of an IMA system design. In ARINC653 standard, IMA adopts hierarchical scheduling strategy. Partitions are based on non-preemptive strictly periodic scheduling, while the tasks in partition are based on preemptive scheduling with fixed priority. The core task of the partition scheduling design is to design the partition scheduling windows for different modules. The window requirements would define the start time (offset) of each partition and its duration for the entire major frame. The major time frame can be defined as the least common multiple of all partition periods in the module. Each major frame contains identical partition scheduling windows. Fig. 1 shows an example of the partition scheduling windows.

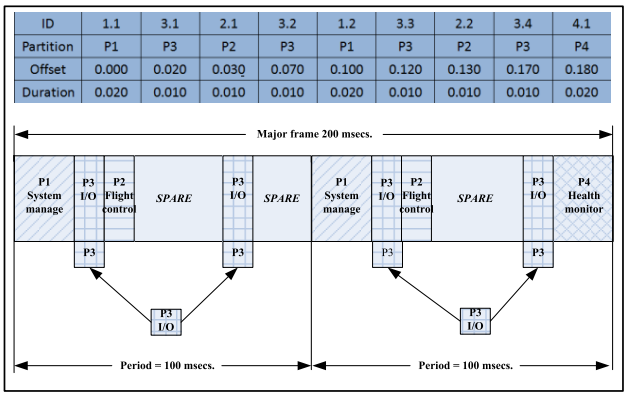

FIGURE 1. Partition windows within a major time frame. There are four partitions in the module, and the major frame for the module is 200 milliseconds. P3 has two windows for each period and 4 windows with in the major frame. P1 and P2 have two windows and P4 has one window within the major frame.

Considering a set of independent partitions $\Gamma = \{\Gamma_1, \cdots, \Gamma_n\}$, each partition Γ_i is characterized by an unknown tuple of (Π_i, Θ_i), where Π_i and Θ_i are the period and the execution length of the partition. In each partition Γ_i, the set $W = \{\tau_1, \cdots, \tau_n\}$ of tasks run in the fixed-priority preemptive schedule. Each task $\tau_i \in \Gamma_i$ is characterized by worst-case execution C_i and period T_i. It is clear that the major frame window for the modules cannot be designed. Therefore, the partition parameters must be obtained before designing partition scheduling windows. The problem of partition parameter design is to find an ''optimal'' schedulable partition parameters $\{(\Pi_i, \Theta_i)\}$ which must satisfy the schedulability of partitions and tasks within partition. The partitions may execute on uniprocessor or multiprocessor platform. Thereore, the method which is used to solve the problem should support the both uniprocessor and multiprocessor platform.

In this paper, we research the problem of partition parameter design. To design the partition parameters, two different objectives are proposed. The first one, denoted as F_1, aims at minimizing the required bandwidth of the partitions. Here, F_1 can be presented by

$$F_1 = \sum_{i=1}^{n} \frac{\Theta_i}{\Pi_i} \tag{1}$$

The required bandwidth represents the overall timing constraint of the partitions along with computational resource requirements. When the required bandwidth is small, it means that computational resource which can be allocated to other application will be larger. In some special case, less bandwidth mean the system requires fewer processors. The second one, denoted as F_2, aims at minimizing system overhead caused by partition scheduling. When all tasks in partitions are schedulable, the lower the system overhead caused by partition scheduling means that the number of times of partition context switching is less in unit time. Therefore, if the system overhead is small, tasks in a partition will not be interrupted frequently due to partition scheduling and system utilization will increase. The partition period is negatively correlated with the times of partition switching. Therefore, the lower system overhead, and the longer partition period.

B. RELATED WORK

In this section, the past works related to the focus areas are introduced from three aspects: schedulability analysis of partition scheduling, schedulability analysis of tasks in partition and partition parameter design.

According to ARINC653 standard, partitions are based on non-preemptive strictly periodic scheduling. Korst *et al.* presented a necessary and sufficient schedulability condition for two tasks, which is a basic theorem to study the schedulability analysis for non-preemptive strictly periodic scheduling problem [2]. For harmonic period tasks, Eisenbrand proposed binary search trees to determine whether partitions are schedulable or not, while Omar kermai proposed FFID method [3], [4]. In addition, Kermia and Sorel proposed a necessary schedulability condition which was proven to be very restrictive [5]. Based on korst' work, Jinchao Chen and Kermai presented sufficient condition for more than two tasks

[6]. In [7], Zhang studied how to select time slots for strictly periodic tasks to make them schedulable and propose an efficient method to solve the problem.

In traditional uniprocessor real-time systems, Liu presented the utilization bounds of a periodic task set under EDF and RM scheduling [8]. On the basis of their research, a large number of scholars have studied and proposed many improved algorithms, but these algorithms are only suitable for single-level scheduling. The tasks in the partition are different from the above research. The impact of partition resources on task schedulability also needs to be considered. Research in hierarchical schedulability analysis for uniprocessor platforms has matured over years. Tei-Wei Huo and Ching-Hui Li developed exact schedulabiltiy conditions for rate-monotonic (RM) system-level scheduler with harmonic task periods [9]. Lipari and Baruah gave exact conditions [10] for dynamic priority (EDF) system-level scheduler. In [11], E. Bini gave conditions for schedulabilitys of an elementary under periodic model, when partition uses EDF or RM respectively. Shin and Lee provided the utilization bounds of a periodic task set over the periodic resource model and the abstraction bounds of periodic interfaces for a periodic task set under EDF and RM scheduling [12].

For the problem of partition parameter design, many scholars proposed different partition resource models and used different methods to solve the problem. Feng and Mok introduced the bounded delay resource model to facilitate hierarchical resource sharing [13]. The bounded delay model uses the maximum time interval to avoid discussing the impact of other partitions on the designed partition, such a bounded delay resource model is not suitable for IMA partition design. The schedulability analysis and partition design problems for real-time applications under the periodic resource model (parameters: period Π and execution time Θ) have been addressed by [12]. Easwaran *et al.* extended the periodic resource model to the explicit deadline period (EDP) model and developed an algorithm to compute a bandwidth optimal EDP model [14]. For the problem of partition parameter design, Nathan Fisher provided a fully-polynomial-time approximate algorithm for bandwidth allocation in a compositional real-time system [15], [16]. In [17], the EDP model was used to analyze hierarchical scheduling and proposed an algorithm to compute the bandwidth. The algorithms proposed by Nathan Fisher and Van den MMHP Martijn need to get period before solving the execution time. However, the impact of other partitions on the same processor is not considered.

For the problem of task assignment, some researchers have studied and proposed algorithms to solve the problem. Al-Sheikh *et al.* analyzed the non-preemptive and strictly periodic scheduling problem in Integrated Modular Avionics (IMA) platform, and presented a best-response algorithm based on the game theoretic approach to give an optimal solution [18], [19]. Pira and [20], Jinchao *et al.* [6] and Kermai [3] did the similar work and presented the Line Search Method (LSM), eigentask and mapping function (EMTA) and TASC respectively to solve the problem. In [4], Eisenbrand considered the problem of scheduling tasks on a minimum processor platform. In addition, they proved that there exists a 2-approximation for the minimization problem.

Similar to this paper, Xiaoguang G described a method to design the IMA partition parameters [21]. The method used the response upper bound of tasks under two-level hierarchical to calculate the partition parameters. With the increase of the number of partitions and the tasks, the calculation error and the required bandwidth also increased. Yoon did the same work and gave Geometric Programming to solve the problem [22]. However, both methods may only be used to design the partition parameters for uniprocessor.

For the case of multiprocessor, several partition models have been proposed in the literature. For the schedulability problem of task sets scheduled on multiprocessors system, many global scheduling algorithms such as global preemptive EDF and global preemptive FP have been proposed for schedulability analysis [23]. Shin *et al.* extended their periodic resource model for multiprocessor platforms [24]. In [25], a Generalized Multiprocessor Periodic Resource model was introduced to calculate the overall resource consumption. IMA Partitions are based on non-preemptive strictly periodic scheduling. Therefore, such algorithms and models are not suitable for IMA partition design.

C. MAIN CONTRIBUTIONS

In this paper, the problem of partition parameter design for multiprocessors system is investigated. The contributions of this work can be summarized as follow:

- A new algorithm is proposed to determine whether a given partition set is schedulable. The algorithm sorts the partitions according to ''Utilization Factor'', and then finds the feasible start time of partitions one by one following the order. Unfortunately, the schedulability condition is just sufficient.
- An approximation algorithm is developed to calculate the minimum execution time of partition for a given partition period while simultaneously guaranteeing tasks schedulability within the partition.
- For problem of partition parameter design, we develop an improved MOEA/D algorithm, called MOEA/D-ADV. In MOEA/D-ADV, in order to maintain better diversity and convergence of the optimal solutions, an adaptive strategy is used to detect the effectiveness of each direction vector. Then positions of the ineffective direction vectors and the size of each subpopulation are adjusted.

D. PAPER ORGANIZATION AND NOTATIONS

This paper is organized as follows. After a review of the related work in Section I, the definitions are formally presented in Section II. In section III-A, a new schedulability analysis method for partition scheduling is proposed. Section III-B presents an algorithm for determining minimum

capacity of partition. Section IV presents MOEA/D-ADV algorithm used to solve problem of partition parameter design. Section V gives some experimentations and results. Section VI finally concludes and gives some perspectives on future work.

Table 1 lists the notations used in this paper. Each partition is associated with a three-tuple (Π_i, Θ_i, S_i), where Π_i, Θ_i and S_i denote the period, the execution time and start time of partition respectively. Without loss of generality, the period of task within partition is assumed to be equal to the deadline of task.

TABLE 1. Summary of key notations.

Notation	Definition
Π_i	Period of partition i
Θ_i	Execution time of partition i
S_i	Start time of partition i
T_j	period of task j
C_j	Execution time of task j
α_i	utilization of partition i, $\alpha_i = \frac{\Theta_i}{\Pi_i}$
u_j	utilization of task j, $u_j = \frac{C_j}{T_j}$
W	task set
Γ	partition set

II. PRELIMINARIES

Definition 1 (Harmonic Period Partitions): Considered a partition set $S = \{\Gamma_1(\Pi_1, \Theta_1), \cdots, \Gamma_n(\Pi_n, \Theta_n)\}$, if $\Pi_i < \Pi_j$ and Π_j is an integral multiple of Π_i, the partition set will be called harmonic period partitions.

Definition 2 (Partition): Partition is a task management unit that the operating system aggregates for a group of attributes. It is a core concept in ARINC653. A partition consists of one or more tasks that are executed concurrently. Tasks within a partition share the system resources occupied by the partition.

Definition 3 (Conflict Intervals): Conflict intervals of partition Γ_i, denoted as CI_i, are intervals which cannot be used to execute Γ_i. Otherwise, partition Γ_i will conflict with partitions which are schedulable on the same processor at some point.

Definition 4 (Available Intervals): Available intervals of partition Γ_i, denoted as AI_i, are intervals exclude the conflict intervals in its period. The available intervals can be allocated to partition Γ_i.

III. SCHEDULABILITY OF IMA HIERARCHICAL SCHEDULING

A. SCHEDULABILITY ANALYSIS OF PARTITION SCHEDULING

For two partition Γ_i and Γ_j, they are schedulable on the same processor if and only if there is no overlapping time unit among their instances. The following theorem was first presented by Korst *et al.* [2]. The theorem provides a sufficient and necessary condition to check the schedulability of two partitions.

Lemma 1: The two partitions $\Gamma_i = (\Pi_i, \Theta_i, S_i)$ and $\Gamma_j = (\Pi_j, \Theta_j, S_j)$ are schedulable on the same processor if and only if

$$\Theta_i \le (S_j - S_i) \bmod (g) \le g - \Theta_j \tag{2}$$

where $g = GCD(\Pi_i, \Pi_j)$.

Lemma 1 does not require the start time of the partition Γ_i is smaller than that of partition Γ_i, and it follows that: $(-a) \bmod (b) = b - (a) \bmod (b)$, if $a, b > 0$.

Corollary 1: The necessary condition to determine whether or not two partitions are schedulable can be given as follow:

$$\Theta_i + \Theta_j \le g \tag{3}$$

Corollary 1 is a necessary condition to determine whether or not two partitions are schedulable when their start time offsets can be freely assigned by the designers. Lemma 1 can be used to check the schedulability of two partitions. However, it requires not only execution time and period of partitions but also start time of partitions. In most cases, start time of partitions are unknown. In [5], Jinchao Chen presented sufficient condition for more than two partitions. However, there are some limitations for the method. The method required that the periods of new partitions are equal to, or multiple of, either one of the periods or the Greatest Common Divisor (GCD) of all periods of the existing partition. The aim of the following section is to propose sufficient condition without constraints on the periods of partitions. The condition allows the system designers to check the schedulability of three or more partitions based on the partition period and execution time.

Theorem 1: Given a partition set $\Gamma = \{\Gamma_1(\Pi_1, \Theta_1), \Gamma_2(\Pi_2, \Theta_2)\}$, Γ_1 is scheduled before Γ_2. In the interval $[0, \Pi_2]$, the total length of conflict intervals of Γ_2 is :

$$len\left(CI_2^1\right) = \frac{\Pi_2}{GCD(\Pi_1, \Pi_2)}\Theta_1 \tag{4}$$

The conflict intervals of Γ_2 can be represented as:

$$CI_2^1 = \bigcup_{k=-\left\lfloor \frac{S_1+\Theta_1-1}{g} \right\rfloor, \cdots, 0, \cdots, \left\lfloor \frac{\Pi_2-S_1}{g} \right\rfloor} \left[S_1 + k^*g, S_1 + k^*g + \Theta_1\right) \cap [0, \Pi_2) \tag{5}$$

Proof: Assume that the start instants of partition Γ_1 and Γ_2 are S_1 and S_2 respectively. They start executing at time instant $T_1 = S_1 + k^*\Pi_1 = S_1 + k^*n_1^*g$ and $T_2 = S_2 + l^*\Pi_2 = S_2 + l^*n_2^*g$ in each of their periods, $g = GCD(\Pi_1, \Pi_2)$. In the interval $[0, \Pi_2]$, the execution time units of Partition Γ_1 and Partition Γ_2 are:

$$I_1 = \bigcup_{k=0, \cdots, \left\lfloor \frac{\Pi_2}{\Pi_1} \right\rfloor} \left[S_1 + k^*n_1^*g, S_1 + k^*n_1^*g + \Theta_1\right),$$

$$I_2 = \bigcup_{l=0,\cdots,\left\lfloor \frac{\Pi_2}{\Pi_1} \right\rfloor} \left[S_2 + l^* n_2^* g, S_2 + l^* n_2^* g + \Theta_2\right).$$

From Condition (2), if the two partitions are schedulable, then $S_2 \subseteq \left[S_1 + m^* g + \Theta_1, S_1 + m^* g + g\right), m \in Z$ and $\Theta_1 + \Theta_2 \leq g$. Therefore, in the interval $[0, \Pi_2]$, conflict intervals of Partition Γ_2 are :

$$CI_2^1 = \bigcup_{k=-\left\lfloor \frac{S_1+\Theta_1-1}{g} \right\rfloor,\cdots,0,\cdots,\left\lfloor \frac{\Pi_2-S_1}{g} \right\rfloor} \left[S_1 + k^* g, S_1 + k^* g + \Theta_1\right) \cap [0, \Pi_2).$$

The conflict interval units of Partition Γ_2 is composed of $\frac{\Pi_2}{g}$ intervals of length Θ_1. Therefore, length of conflict intervals of Partition Γ_2 is:

$$len\left(CI_2^1\right) = \frac{\Pi_2}{GCD\left(\Pi_1, \Pi_2\right)}\Theta_1.$$

According to the definitions of conflict intervals and available intervals, the available intervals of Partition Γ_2 are:

$$\mathrm{A}I_2^1 = [0, T) - CI_2^1.$$

Theorem 2: Given a partition set $\Gamma = \{\Gamma_1(\Pi_1, \Theta_1), \cdots, \Gamma_n(\Pi_n, \Theta_n)\}$, the partitions are schedulable on the same processor. Add a new partition $\Gamma_{n+1}\left(\Pi_{n+1}, 1\right)$ into partition set Γ. The new partition set Γ' is schedulable if :

$$\frac{\Theta_1}{g_{1,n+1}} + \frac{\Theta_2}{g_{2,n+1}} + \cdots + \frac{\Theta_n}{g_{n,n+1}} + \frac{1}{\Pi_{n+1}} \leq 1 \qquad (6)$$

Proof: According to Theorem 1, in the interval $[0, \Pi_{n+1}]$, conflict intervals of Partition Γ_{n+1} are:

$$CI_{n+1}^i = \bigcup_{k_i=-\left\lfloor \frac{S_i+\Theta_i-1}{g} \right\rfloor,\cdots,0,\cdots,\left\lfloor \frac{\Pi_{n+1}-S_i}{g} \right\rfloor} \left[S_i + k_i^* g_{i,n+1}, S_i + k_i^* g_{i,n+1} + \Theta_i\right) \cap [0, \Pi_{n+1}).$$

Available intervals of Partition Γ_{n+1} are:

$$\begin{aligned} \mathrm{A}I_{n+1} &= [0, \Pi_{n+1}) - CI_{n+1}, \\ len\left(\mathrm{A}I_{n+1}\right) &\geq \Pi_{n+1} - \frac{\Theta_1^* \Pi_{n+1}}{g_{1,n+1}} - \frac{\Theta_2^* \Pi_{n+1}}{g_{2,n+1}} - \cdots \\ &\quad - \frac{\Theta_n^* \Pi_{n+1}}{g_{n,n+1}}. \end{aligned}$$

The length of available intervals is greater than or equal to 1. So there is at least one subinterval $[t, t+1) \subseteq AI_{n+1}$. In Partition Γ_{n+1}'s period, the execution time unit of Partition Γ_{n+1} is:

$$I_{n+1} = [S_{n+1}, S_{n+1} + 1), \quad S_{n+1} \in [0, \Pi_{n+1})$$

Therefore, $I_{n+1} \subseteq AI_{n+1}$ and $I_1 \cap I_2 \cdots \cap I_{n+1} = \emptyset$. The new partitions Γ' are schedulable.

Theorem 3 Given a partition set $\Gamma = \{\Gamma_1\left(\Pi_1, \Theta_1\right), \cdots, \Gamma_n\left(\Pi_n, \Theta_n\right)\}$, the partitions are schedulable on the same processor. Add a new partition $\Gamma_{n+1}\left(\Pi_{n+1}, \Theta_{n+1}\right)$ into partition set Γ. The new partition set Γ' is schedulable if :

$$\Theta_{n+1} \leq LLEN\left(AI_{n+1}\right) \qquad (7)$$

where $LLEN\left(AI_{n+1}\right)$ is the longest consecutive duration of available intervals.

Proof: According to Theorem 1, in the interval $[0, \Pi_{n+1}]$, available intervals of Partition Γ_{n+1} are :

$$\begin{aligned} CI_{n+1}^i &= \bigcup_{k_i=-\left\lfloor \frac{S_i+\Theta_i-1}{g} \right\rfloor,\cdots,0,\cdots,\left\lfloor \frac{\Pi_{n+1}-S_i}{g} \right\rfloor} \left[S_i + k_i^* g_{i,n+1}, S_i + k_i^* g_{i,n+1} + \Theta_i\right) \cap [0, \Pi_{n+1}), \\ \mathrm{A}I_{n+1} &= [0, \Pi_{n+1}) - CI_{n+1}. \end{aligned}$$

When $\Theta_{n+1} \leq LLEN\left(AI_{n+1}\right)$, there is at least one interval $[t, t + \Theta_{n+1}] \subseteq AI_{n+1}$. In partition Γ_{n+1}'s period, the execution time unit of Partition Γ_{n+1} is:

$$I_{n+1} = [S_{n+1}, S_{n+1} + \Theta_{n+1}), \quad S_{n+1} \in [0, \Gamma_{n+1}).$$

If $S_i = t$, then $I_{n+1} \subseteq AI_{n+1}$ and $I_1 \cap I_2 \cdots \cap I_{n+1} = \emptyset$. New partition set Γ' is not overlap in the interval $[0, \Pi_{n+1}]$.

In the interval $[0, LCM(\Pi_1, \cdots, \Pi_{n+1})]$, there are a set of intervals $\left[t, t + k^* \Pi_{n+1} + \Theta\right)\left(k = 0, 1 \cdots, \frac{LCM(\Pi_1, \cdots, \Pi_{n+1})}{\Pi_{n+1}}\right)$. $\subseteq AI_{n+1}$. Therefore, new partition set Γ' is schedulable.

Both Theorem 2 and Theorem3 give the condition to check the schedulability of partitions. For schedulability test, Theorem 2 is simpler than Theorem 3. It can check the schedulability of partitions not involving partitions start time parameters. However, Theorem 2 requires that execution time of partitions is equal to 1.

Consider a partition set $\Gamma\left(\Gamma_1, \Gamma_2, \Gamma_3\right)$ with $\Gamma_1\left(4, 1\right)$, $\Gamma_2\left(6, 1\right)$, $\Gamma_3\left(8, 1\right)$. We assume three different partition orders: $\{\Gamma_1, \Gamma_2, \Gamma_3\}$, $\{\Gamma_2, \Gamma_1, \Gamma_3\}$, $\{\Gamma_3, \Gamma_2, \Gamma_1\}$. Thus, when Γ_2 has been scheduled before Γ_1 and Γ_3, the partition set Γ is not schedulable. In some case, the partition scheduling is performed according to an increasing order of partition periods. In fact, sometimes the scheduling according to an increasing order of periods is infeasible. However, according to other orders, the same system is schedulable. For another example, there is a partition set with three partitions $\Gamma\left(\Gamma_1, \Gamma_2, \Gamma_3\right)$ with $\Gamma_1\left(8, 2\right)$, $\Gamma_2\left(16, 2\right)$, $\Gamma_2\left(20, 2\right)$. A partition sort according to an increasing periods order is $\{\Gamma_1, \Gamma_2, \Gamma_3\}$. However this system is not schedulable if Γ_2 has been scheduled before Γ_3. On the other hand, if Γ_3 has been scheduled before Γ_2, the partition set is schedulable. As can be seen from the above two examples, for the same partition set, the partition order affects the final feasibility result.

It is assumed that partitions $\Gamma = \{\Gamma_1, \cdots, \Gamma_{i-1}\}$ are scheduled before Γ_i. In the interval $[0, \Pi_i]$, $\frac{\Pi_i}{g_{i,j}}$ intervals of length Θ_i are considered as busy and cannot be used to schedule Γ_i because of Γ_j scheduling. The sum of busy intervals length can be represented as follow:

$$L_i = \frac{\Theta_1^* \Pi_i}{g_{1,i}} + \frac{\Theta_2^* \Pi_i}{g_{2,i}} + \cdots + \frac{\Theta_{i-1}^* \Pi_i}{g_{i-1,i}} \qquad (8)$$

When $\Theta_i > \Pi_i - L_i$, it is mean that time units may not be sufficient to execute Γ_i. Therefore, it is recommended that partition Γ_i would not be scheduled at last. The "Utilization Factor", denoted as $\mathbb{C}_i$, is proposed to represent the sum of

time fractions utilized to execute Γ_i, which is computed as follows:

$$\mathbb{C}_i = \sum_{\Gamma_k \neq \Gamma_i} \frac{\Theta_k}{g_{k,i}} + \frac{\Theta_i}{\Pi_i} \tag{9}$$

If $\mathbb{C}_i > 1$ and $\mathbb{C}_j \leq 1$, Γ_i would be scheduled before Γ_j. Therefore, ''Utilization Factor'' can be used as a condition to sort partitions. The process of sorting partitions is described as follow:

First, the ''Utilization Factor'' of partitions is computed by equation (9). Then the algorithm browses the partition set Γ to find the partition with $\mathbb{C}_i > 1$ and inserts it to a new partition set Γ_a. After loop, remaining partitions are inserted to another partition set Γ_b. At last, Γ_a and Γ_b are sorted in an increasing periods order respectively. A new partition set Γ' is obtained by inserting Γ'_b to the end of Γ'_a.

Although Theorem 3 gives the condition to check the schedulability of partitions, it can't be used directly to determine whether a given partition set is schedulable. Therefore, the proposed sufficient schedulability test is based on Algorithm1 which can be summarized as follows.

Algorithm 1 Sufficient Schedulability Test

Input: partition set $\Gamma = \{\Gamma_1, \cdots, \Gamma_n\}$
Output: Determination whether Γ is schedulable and valid start time of Γ_i if it is schedulable.

```
1  if condition(3) is not met then
2     return Γ is not schedulable;
3  end if
4  for i = 1; i ≤ n; i++ do //sort partitions
5     C_i is computed by equation (9).
6     if C_i > 1 then
7        Γ_a ← Γ_a ∪ {Γ_i};
8     else
9     Γ_b ← Γ_b ∪ {Γ_i};
10    end if
11 end for
12 Sort Γ_a and Γ_b in an increasing periods order
13 Γ' ← Γ'_a ∪ Γ'_b// insert Γ'_b to the end of Γ'_a
14 for i = 2; i ≤ n; i++ do
15    S_i ← −1, AI_i ← [0, Π_i)
16    for j = 1; j ≤ i; i++
17       CI_i is computed by equation (5).
18       AI_i ← AI_i − CI_i
19    end for
20    if Θ_i ≤ LLEN (AI_i) then
21 [t, t + Θ_i) ⊆ AI_i, S_i ← t
22       end if
23    if S_i = −1
24       return no answer;
25    end if
26 end for
27 Return Γ is schedulable;
```

The Algorithm1 is consisted of two parts. The first part (lines 4-12) is to sort partitions, while the second part (lines 14-26) is to calculate the start time of all partitions. To calculate the start time of all the partitions, the start time of partition are assigned one by one following the order of partition set Γ'. The second part of algorithm 1 has a structure of double closed loops. At each iteration of the outer loop, Partition Γ_i is selected from the partition set Γ' and available intervals of Γ_i are updated (Lines 16-18). Then, available intervals are reviewed to examine if there is any consecutive duration larger than or equal to Θ_i (Lines 20-22). If so, such a consecutive duration is legal for the partition Γ_i execution and the beginning instant is set to Γ_i's start time. Notice that, if more than one intervals satisfy the condition, the first appeared interval will be selected. If Γ_i's start time is equal to -1, the loop is terminated and the algorithm returns ''no answer''.

The idea of Algorithm1 is to compute the necessary available intervals to schedule all the partitions. This can be seen as the classical Bin Packing problem which is an NP-hard problem. Before calculating the start time of partitions, the the partitions order is sorted according the ''Utilization Factor''. Therefore, Algorithm1 is a First Fit Decreasing (FFD) algorithm.

The main computation part of Algorithm 1 is from line 4 to 26, which is consisted of two parts. For the first part, main computation part is Line 5. According to equation (5), complexity of Line 5 is $O(n)$. However, complexity of all greatest common divisor computation should also be considered. Greatest common divisor can be computed very efficiently using Euclid's algorithm, with computational complexity $O(b^2)$ bits for b-bit numbers. Therefore, the computational complexity of line 5 is $O(b^2 n)$, and the computational complexity of the sorting partitions is $O(b^2 n^2)$. For the second part, the computational complexity of calculating the start time of partitions is $O(n^2 + n^* \Pi_{\max})$.The total computational complexity of Algorithm 1 is $O(b^2 n^2 + n^2 + n^* \Pi_{\max})$.

Although Algorithm 1 can be used to check the schedulability of harmonic period partitions and non-harmonic period partitions, there are some limitations of our algorithm:

(1) Note that, Algorithm 1 is First Fit Decreasing (FFD) algorithm, it is not an optimal method. In the process of start time setting for a selected partition, we directly choose the first suitable available intervals for the partition. Therefore, the longest consecutive duration obtained by Algorithm 1 is not the best solution. The partitions which do not satisfy the condition (7) cannot be deduced that the partitions are not schedulable. Therefore, the proposed schedulability test algorithm is a sufficient but not a necessary condition.

(2) The computational complexity of the algorithm presented by Jinchao Chen is $O(\Pi_{\max} n)$, while the computational complexity of the FFID algorithm presented by Kermai is $O\left(2\frac{\Pi_{\max}}{\Pi_{\min}} n\right)$. The computational complexity of algorithm 1 is much greater than the above two algorithms. For the harmonic period partitions, the partitions can be sorted in an increasing periods order, instead of algorithm1. Therefore, it

is not recommended to use algorithm 1 to check the schedulability in harmonic case.

An example is given below to describe the execution process of algorithm 1. In this example, we consider a partition set $\Gamma(\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4)$ with $\Gamma_1(8, 2)$, $\Gamma_2(16, 1)$, $\Gamma_1(16, 2)$, $\Gamma_4(20, 2)$. The process of algorithm1 is given as follows:

First, condition (3) is met for all partitions. Then the utilization factors of partitions are computed according to the equation (9): $\mathbb{C}_1 = 1.125 > 1$, $\mathbb{C}_2 = 0.9375 < 1$, $\mathbb{C}_3 = 0.9375 < 1$, $\mathbb{C}_4 = 1.35 > 1$. Therefore, $\Gamma_a = \{\Gamma_1, \Gamma_4\}$ and $\Gamma_b = \{\Gamma_2, \Gamma_3\}$. Γ_a and Γ_a are sorted in an increasing periods order respectively. The new partition set is: $\{\Gamma_1, \Gamma_4, \Gamma_2, \Gamma_3\}$. In what followed, the start time of partitions are computed, and the start time of Γ_1 is 0.

First iteration: $AI_4 = \{[2,4), [6,8), [10,12), [14,16), [18,20)\}$. Therefore, $LLEN(AI_4) = 2 \geq \Theta_4$ and $S_4 = 2$.

Second iteration: $AI_2 = \{[4, 6), [12, 14)\}$.Therefore, $LLEN(AI_2) = 2 \geq \Theta_2$ and $S_2 = 4$.

Third iteration: $AI_3 = \{[5, 6), [12, 14)\}$. Therefore, $LLEN(AI_3) = 2 > \Theta_3$ and $S_3 = 12$.

Therefore, the partitions $\{\Gamma_1, \Gamma_4, \Gamma_2, \Gamma_3\}$ are schedulable on the processor. However, if the partitions execute in order: $\{\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4\}$, the partitions will be not schedulable on the processor.

B. AN ALGORITHM FOR DETERMINING MINIMUM CAPACITY OF PARTITION

In order to analyze the schedulability of tasks in partition, it is necessary to calculate the resource supply of the scheduling partition. A resource model is to specify resource allocations that are provided to a scheduling partition and to calculate the resource supply to the component. For schedulability analysis, it is important to calculate the minimum supply of partition resources accurately. To solve the problem, Shin and Lee proposed a periodic resource model (Π, Θ) in [12], and its supply-bound function is as follows:

$$sbf(t) = \begin{cases} k\Theta & \text{otherwise} \\ t - (k+1)(\Pi - \Theta) & t \in [(k+1)\Pi - \Theta, (k+1)\Pi] \end{cases} \tag{10}$$

where $k = \lfloor \frac{t}{\Pi} \rfloor$. Since the supply-bound function $sbf(t)$ is a discrete function, its linear low-bound function $lsbf(t)$ is as follows:

$$lsbf(t) = \begin{cases} \frac{\Theta}{\Pi}(t - (\Pi - \Theta)) & t \geq (\Pi - \Theta) \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

For a periodic task set W under RM scheduling, Lehoczky proposed a demand-bound function $rbf(W, t_i)$ that computes the worst-case cumulative resource demand of a task P_i for an interval of length t [26].

$$rbf(W, t, i) = \left\lceil \frac{t}{T_i} \right\rceil C_i + \sum_{P_k \in HP(i)} \left\lceil \frac{t}{T_k} \right\rceil \cdot C_k \tag{12}$$

where $HP(i)$ is the set of higher-priority tasks than P_i in W.

Lemma 2 [12]: Consider task set $W = \{P_1(C_1, T_1), \cdots, P_n(C_n, T_n)\}$, where for each i, $P_i = (C_i, T_i)$ is periodic or sporadic task with relative deadlines equal to periods. W is schedulable in partition, if and only if:

$$\forall P_i \in W \exists t_i \in [0, T_i]\, rbf(W, t_i) \leq sbf(t_i) \tag{13}$$

For schedulability analysis of tasks within partition, Jung-Eun Kim proposed a new method to obtain schedulability bound for tasks in a given IMA partition without any information on task execution time [27]. The computation complexity of method is superior to Lemma 2. For the method, partition period should not be longer than any task's period within the partition. If partition period is larger than any task's period, the method will not be suitable for determining the schedulability of tasks. In many cases, task's period may be larger than the major cycle. The Lemma 2 is used to determine the schedulability of tasks, no matter whether the period of partition is less than or greater than task's period.

Lemma 2 gives sufficient and necessary condition to check whether tasks within a partition are schedulable, but it cannot be used to compute the optimal execution time. In this section, pseudo code of our algorithm is presented. The algorithm takes the task set and partition period as input parameters to calculate the minimum bandwidth allocated to the partition. According to Lemma 2 and the resource supply function of the periodic resource model, the minimum bandwidth satisfies the inequality $rbf(W, t_i) \leq sbf(t_i)$.

From the definition of sbf, when $t \in [k\Pi, (k+1)\Pi - \Theta]$, it follows that $\Theta \geq \frac{rbf(t)}{k}$. When $t \in [(k+1)\Pi - \Theta, (k+1)\Pi]$, it follows that $\Theta \geq \Pi + \frac{rbf(t)-t}{k+1}$.

To calculate $\Theta_{\min}$, the parameters $\Pi, rbf(t), t$ are used as known conditions to determine whether $\Theta \geq \Pi + \frac{rbf(t)-t}{k+1}$ or $\Theta \geq \frac{rbf(t)}{k}$.When $\Pi + \frac{rbf(t)-t}{k+1} \geq \frac{rbf(t)}{k}$, we have $y(k) = k^2\Pi + k(\Pi - t) - rbf(t) \geq 0$. $y(k)$ is a quadratic function. If $k \geq \frac{(t-\Pi)+\sqrt{(\Pi-t)^2+4\Pi rbf(t)}}{2\Pi}$ or $k \leq \frac{(t-\Pi)-\sqrt{(\Pi-t)^2+4\Pi rbf(t)}}{2\Pi}$, then $y(k) \geq 0$ and $\Pi + \frac{rbf(t)-t}{k+1} \geq \frac{rbf(t)}{k}$. Since k is an integer greater than zero, the following conclusions are drawn:

When $k \geq \lfloor n \rfloor + 1$, $\Theta_{\min} = \Pi + \frac{rbf(t)-t}{k+1}$, where $n = \frac{(t-\Pi)+\sqrt{(\Pi-t)^2+4\Pi rbf(t)}}{2\Pi}$. When $k \leq \lceil n \rceil - 1$, $\Theta_{\min} = \frac{rbf(t)}{k}$.

The complexity of algorithm 2 depends on the number of tasks in the task set and the period of each task T_i. The outer loop of algorithm iterates for each task, thus n times in total. The number of iterations of inner loop is related to the period of each task. According to equation (12), complexity of computing the worst-case cumulative resource demand is $O(n)$. Therefore, the computational complexity of algorithm 2 is at most $O\left(n^{2*}T\right)$ (T is the maximum period of tasks) times. In the section, it has been shown that algorithm 2 gives a valid answer when the partition period Π is given. The main problem of algorithm 2 is how to determine the partition period Π. Therefore, the impact of partition period on partition utilization will be introduced in following paragraph.

Algorithm 2 Pesude Code for Computing Minimum Execution

Input: partition set S
Sort all partition in an increasing periods order, then $T_1 \leq T_2 \leq \cdots \leq T_n$
Output: The minimum execution time of partition

$\Theta_{\max} \leftarrow 0$
For all $\tau_i \in T$ do
  $\Theta_{\min} \leftarrow T_i$
  $t \leftarrow T_i$
  While $t > 0$ **do**
    $new\_demand \leftarrow computeRBF\ (i, T_i)$
    if$(new\_demand == old\_demand)$
      Continue
    end if
    $k \leftarrow \frac{t}{\Pi}$
    $\Theta_1 \leftarrow \frac{w(t)}{k}$
    $\Theta_2 \leftarrow \Pi + \frac{w(t)-t}{k+1}$
    $M \leftarrow \left\lfloor \frac{(t-\Pi)+\sqrt{(\Pi-t)^2+4\Pi rbf(t)}}{2\Pi} \right\rfloor + 1$
    if $M \geq k$
      $\Theta \leftarrow \Theta_2$
    else
      $\Theta \leftarrow \Theta_1$
    end if
    $\Theta_{\min} \leftarrow \min(\Theta, \Theta_{\min})$
  end for
  $\Theta_{\max} \leftarrow \max(\Theta_{\max}, \Theta_{\min})$
end for
$\Theta_{opt} \leftarrow \Theta_{\max}$

Theorem 4: Given a task set $W = \{P_1(C_1, T_1), \cdots, P_n(C_n, T_n)\}$ within the partition $\Gamma_{\Pi,t} = (\Pi, \Theta)$, where d_t is worst-case cumulative demand of all tasks, the minimum execution time of partition Θ_{opt} is as follows:

$$\Theta_{opt} \leq Exec(\Pi, t, d_t) = \frac{\Pi - t + \sqrt{(\Pi - t)^2 + 4\Pi d_t}}{2} \tag{14}$$

$$BW_{opt} = \frac{\Theta_{opt}}{\Pi} \leq BW(\Pi, t, d_t) = \frac{\Pi - t + \sqrt{(\Pi - t)^2 + 4\Pi d_t}}{2\Pi} \tag{15}$$

Proof: according to Lemma 2, $rbf(W, t_i) \leq sbf(t_i)$. When $rbf(W, t_i) = sbf(t_i)$, the execution time Θ is equal to the minimum execution time of partition Θ_{opt}. Since $lsbf(t) \leq sbf(t)$, when $rbf(W, t_i) = lsbf(t_i)$, the execution time Θ' can be obtained which is larger than Θ_{opt}.

$$\begin{aligned} lsbf(t) &= rbf(W, t) = d_t \\ &\Leftrightarrow \frac{\Theta}{\Pi}(t - (\Pi - \Theta)) = d_t \\ &\Leftrightarrow \Theta^2 + \Theta(t - \Pi) - \Pi d_t = 0 \\ &\Leftrightarrow \Theta = \frac{\Pi - t \pm \sqrt{(\Pi - t)^2 + 4\Pi d_t}}{2} \end{aligned}$$

Since $\Theta \geq 0$, the above equation has a unique solution:

$$\Theta = \frac{\Pi - t + \sqrt{(\Pi - t)^2 + 4\Pi d_t}}{2}$$

In other words, $\Theta_{opt} \leq Exec(\Pi, t, d_t) = \frac{\Pi - t + \sqrt{(\Pi - t)^2 + 4\Pi d_t}}{2}$.

As a result, the minimum bandwidth of $\Gamma_{\Pi,t}$ is $BW_{opt} = \frac{\Theta_{opt}}{\Pi} \leq BW(\Pi, t, d_t) = \frac{\Pi - t + \sqrt{(\Pi - t)^2 + 4\Pi d_t}}{2\Pi}$.

Theorem 5: The function $BW(\Pi, t, d_t)$ defined in Theorem 4 is increasing on the domain of Π.

Proof:

$$\begin{aligned} &\frac{dBW(\Pi, t)}{\Pi} \\ &= \left(\frac{\Pi - t + \sqrt{(\Pi - t)^2 + 4\Pi d_t}}{2\Pi}\right)' \\ &\Leftrightarrow t\Pi\sqrt{\frac{(\Pi - t)^2 + 4\Pi d_t}{\Pi^2}} + (t - 2d_t) - t^2 \\ &\frac{dBW(\Pi, t)}{\Pi} \geq 0 \\ &\Leftrightarrow t\sqrt{(\Pi - t)^2 + 4\Pi d_t} \geq (t - 2d_t) - t^2 \\ &\Leftrightarrow \left(\sqrt{(\Pi - t)^2 + 4\Pi d_t}\right)^2 \geq \left(\frac{(t - 2d_t) - t^2}{t}\right)^2 \\ &\Leftrightarrow \left((\Pi - t)^2 + 4\Pi d_t\right)t^2 - \left((t - 2d_t) - t^2\right)^2 \\ &= 4d_t\Pi t^2 - 4d_t^2\Pi t \\ &\Leftrightarrow 4d_t\Pi t(t - d_t) \end{aligned}$$

Since $t \geq d_t$, $\frac{dBW(\Pi,t)}{\Pi} \geq 0$, $BW(\Pi, t)$ is increasing with the increase of the partition period.

For a task set W, Theorem 5 shows that the larger the partition period, the greater the partition utilization. When partition period is very small, the scheduling overhead is large because of the increase of context switching. On the contrary, when the partition period becomes larger, the bandwidth requirement of the partition will become larger and the bandwidth left for other partitions will become smaller. Without considering the impact of partition scheduling, we suggest selecting the partition period close to the minimum period among all tasks in W.

IV. PARTITION PARAMETER DESIGN METHOD

A. OBJECTIVE FUNCTIONS AND CONSTRAINTS FOR PARTITION PARAMETER DESIGN

This section demonstrates the objective functions and constraints for partition parameter design. Two different objectives have been considered in the optimization procedure.

As a first optimization goal, minimizing the required bandwidth is considered. When tasks execute in partition, it is hoped that tasks execution are not interrupted frequently by partition scheduling. As a second optimization goal, we consider minimizing the system overhead caused by partition

scheduling, which is defined as

$$F_2 = \sum_{j=1}^{n} \sum_{i}^{k} \left\lceil \frac{R_i}{\alpha_j \cdot \Pi_j} \right\rceil \cdot \frac{T_q}{\Pi_j} \tag{16}$$

where α_j is the bandwidth of partition j, R_i is the worst-case response time of task i, T_q is task-switching time spending and Π_j is the period of partition j. Through the analysis of IMA hierarchical scheduling, the partition period is negatively related to the system overhead, while the partition period is positively related to the required of the partitions. When designing the partition parameters, two opposite objectives must be balanced:

1) The required bandwidth F_1 should be small, and the total processors capacity would not be wasted.

2) The value of F_2 should be small, then the partition period should be large. Otherwise, the time wasted in context switches performed by the global scheduler will be too high. The platform with a low value of F_2 has a higher chance to schedule tasks due to the lower degree of fragmentation of the overall computing capacity.

In fact, when designing the partition parameters, the computing capacity of the system must be considered. The partitions are mapped to the limited number of processors, and the constraint condition is defined as follows:

$$g\left(\Pi_1, \cdots \Pi_n, \Theta_1, \cdots \Theta_n\right) \leq M \tag{17}$$

where M is the number of processors.

Through the schedulability analysis of hierarchical scheduling in the past section, partition period Π needs to be designed before the execution time of partition can be obtained. The partition period is a key factor affecting the schedulability of partitions. According to Lemma 1, it can be easily deduce that when $GCD\left(\Pi_i, \Pi_j\right)$ is a small value, the likelihood of meeting Γ_i and Γ_i strict periodicity and deadlines are also small. Therefore, $GCD\left(\Pi_i, \Pi_j\right)$ should be designed to be as large as possible. If so, the available intervals for other partitions will be larger. According to the partition period, the partition set can be divided into harmonic period partitions and non-harmonic period partitions. For the problem of partition assignment, the computational complexity of algorithm used for non-harmonic period partitions is much greater than that of algorithm used for harmonic period partitions. When the system is designed as harmonic period partitions, the number of required processors is m. When system is designed as a non-harmonic period partitions, the number of required processors is m'. In most cases, m is less than or equal to m'. The difference between m and m' becomes larger with the increase of the number of partitions. Therefore, it is recommended that all partitions are designed as harmonic period partitions. In [3], experiments show that the performance of TASC algorithm in terms of execution time and number of required processors is better than EMTA, Line Search Method and Best-response algorithm for harmonic case. Therefore, the TASC algorithm is used to calculate the number of required processors in the optimization procedure.

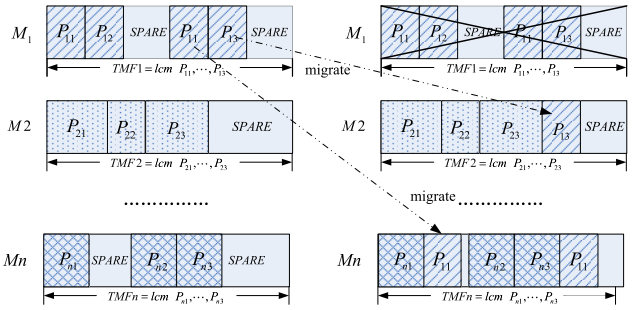

FIGURE 2. Process of partition reconfiguration.

The reliability of integrated avionics system is critical to the flight safety of aircraft. Therefore, when a processor in the IMA system fails, the critical partitions running on the processor need to be migrated to other processors. It is assumed that the IMA system has n processors, and the processor $M1$ has two partitions P_{11} and P_{13} which are related to the safety of the aircraft. Therefore, when the processor $M1$ fails, it is necessary to migrate P_{11} and P_{13} to other processors. The migration process is shown in the Fig. 2. In fact, partition migration is related to the remaining time space and partition period of other processors. In this example, because the period P_{11} is smaller, it can only be migrated to the processor Mn, while the number of processors to which P_{13} can be migrated is relatively larger. Therefore, the smaller partition period is, the harder it will be to find sufficient processor resources in the reconfiguration process. Therefore, the difference between the maximum partition period and minimum partition period is not too large. It is suggested that $\frac{\Pi_n}{\Pi_1} \leq 4$($\Pi_1$ and Π_n are the minimum and maximum periods of the partition).

Therefore, the objective functions and constraints of partition parameter design are as follows:

$$\begin{aligned} &F_1\left(\Pi_1, \Pi_2, \cdots, \Pi_n, W\right) \\ &\quad = \min\left(\sum_{j=1}^{n} \alpha_j\right) \end{aligned} \tag{18}$$

$$\begin{aligned} &F_2\left(\Pi_1, \Pi_2, \cdots, \Pi_n, W\right) \\ &\quad = \min\left(\sum_{j=1}^{n} \sum_{i}^{k} \left\lceil \frac{R_i}{\alpha_j \cdot \Pi_j} \right\rceil \cdot \frac{T_q}{\Pi_j}\right) \end{aligned} \tag{19}$$

$$\begin{aligned} &\text{Subject to } \frac{\Pi_i}{\Pi_1} = k, \quad (k = 1, 2, 4) \\ &\qquad T_{\min} \leq \Pi_1 \leq T_{\max} \\ &\qquad g\left(\Pi_1, \cdots \Pi_n, \Theta_1, \cdots \Theta_n\right) \leq M \end{aligned}$$

where W represents the task set, $T_{\min}$ is the minimum period among all tasks, and $T_{\max}$ is the maximum period among all tasks. Therefore, the problem of partition parameter design is transformed into a constrained multi-objective

problem. The input parameters for calculating objective functions F_1 and F_2 are partition periods $\Pi_1, \Pi_2, \cdots, \Pi_n$ and task set W. The parameter α_j in the objective function $F_1(\Pi_1, \Pi_2, \cdots, \Pi_n, W)$ is solved by the algorithm 2 in section III-B. For R_i, N. Audsley gave an iterative formula for computing the worst execution time of a task [28].

$$w_i^{n+1} = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i^n}{T_j} \right\rceil C_j \tag{20}$$

where $hp(j)$ is the set of tasks with priorities higher than the task i. Recurrence begins with $w_i^0 = C_i$ and ends when $w_i^{n+1} > D_i$ in which case the task is not schedulable or when $w_i^{n+1} = w_i^n$ in which case w_i^{n+1} gives the worst-case response time of the task.

B. THE MOEA/D-ADV ALGORITHM

Based on the above analysis, the problem of partition parameter design can be translated to constrained multi-objective optimization problems (CMOPs). Then the objective function and constraints can be converted to the following formulas:

$$F_1(\Pi, k_1, \cdots, k_n, W) = \min\left(\sum_{j=1}^{n} \alpha_j\right) \tag{21}$$

$$F_2(\Pi, k_1, \cdots, k_n, W) = \min\left(\sum_{j=1}^{n}\sum_{i}^{k} \left\lceil \frac{R_i}{\alpha_j \cdot k_j \cdot \Pi} \right\rceil \cdot \frac{T_q}{\Pi}\right) \tag{22}$$

$$\text{Subject to } k_1, \cdots, k_n = 1, 2, 4$$
$$T_{\min} \le \Pi \le T_{\max}$$
$$\text{TASC}(\Pi_1, \cdots \Pi_n, \Theta_1, \cdots \Theta_n) - M \le 0$$

where TASC is the algorithm proposed by Kermai to calculate the number of required processors.

CMOPs are consisted of a few objectives and constraints. To solve CMOPs, the constraint-handling technique should be applied to the framework of multi-objective evolutionary algorithm (MOEA).

According to different selection strategies, MOEAs mainly can be classified into three types: (1) Pareto-domination-based; (2) decomposition-based; (3) indicator-based. The typical examples of domination-based MOEAs include NSGA-II [29], PAES-II [30], SPEA-II [31]. For indicator-based MOEAs, they use a scalar metric to assist the selection, typical examples of this type are IBEA [32], HypE [33]. In recent years, decomposition-based MOEAs attract a lot of attention. The most representative algorithm of this type is Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D) which was proposed by Qingfu Zhang *et al.* in 2007 [34]. Given a series of uniform distributed weight vectors, the algorithm decomposed the multi-objective problem into N scalar subproblems, and each scalar subproblems related to one solution.

The decomposition-based multi-objective evolutional algorithm (MOEA/D-M2M), introduced by Liu *et al.* [35] in 2014, is a variant of MOEA/D. However, unlike MOEA/D using aggregation methods, MOEA/D-M2M decomposes the multi-objective optimization problems (MOPs) in to a number of simple multi-objective optimization subproblems, and then solves them in a collaborative way [36]. Compared with SPEA-II, NSAG-II, and so forth, MOEA/D-M2M protects every subpopulation by decomposition and solves every subproblem using multi-objective optimization approaches, which balances the diversity and convergence simultaneously at each generation. Recently, MOEA/D-M2M has shown its great advantages in dealing with many kinds of continuous MOPs. For problem of partition parameter design, as the number of partition and processor increases, the optimal solutions of many subproblems are not Pareto-optimal. It may affect the diversity and convergence performance of MOEA/D-M2M. To maintain better diversity and convergence of the optimal solutions, in this paper, we propose a variant of MOEA/D, called MOEA/D-ADV. In MOEA/D-ADV, an adaptive strategy is used to detect the effectiveness of each direction vector. Then positions of the ineffective direction vectors and the size of each subpopulation are adjusted.

To solve CMOPs, the constraint-handling technique should be applied to the framework of the multi-objective evolutionary algorithm (MOEA). According to [37], the constraint-handling techniques mainly include six categories: constrained dominance principle (CDP), penalty functions, stochastic ranking, ε constrained method, multi-objective optimization-based methods, and hybrid methods [38]. Compared with other constraint-handling techniques mentioned above, CDP is simple, flexible, and non-parametric, making it easy to be embedded into MOEA. In this paper, we embed the CDP into MOEA/D-ADV to solve the problem of partition parameter design.

The pseudo code of the proposed method is shown in Algorithm 3. The MOEAD-ADV shares a common framework that is employed by many evolutionary algorithms. First, a population with N solutions is randomly initialized in the whole decision space. Next, more potential solutions are selected into the mating pool according to the constraint violation value of each individual. Then a set of offspring solutions Q is generated by applying crossover and mutation operations. In what followed, direction vectors and the size of subpopulation are adjusted according to the union set of P and Q. Finally, N solutions are selected from the union set of P and Q by adopting an association selection procedure. In the following paragraphs, the four main components in Algorithm3 will be introduce, i.e., constrained tournament selection, adjustment for the direction vectors, adjustment for the size of the subpopulations and association selection.

1) CONSTRAINED TOURNAMENT SELECTION

For the problem of partition parameter design, the constraint violation value of a solution x, denotes as $CV(x)$, is

Algorithm 3 General Framework of MOEA/D-ADV

Input: $Gen_{\max}$: the maximum number of generation
S : the size of each subpopulation
K : the number of the subproblems
G : parameters update frequency
Output: a set of feasible nondominated solutions

$[P, W] \leftarrow Intialization()$; P is the parent population, W is K unit direction vectors $W^1, \cdots, W^K$
While termination criterion is not fulfilled do
$P' \leftarrow Constrained\_Tournament\_Selection\ (P)$;
$Q \leftarrow Variation\ (P')$;
$P \leftarrow P \cup Q$
if $Mod(gen, G) == 0$ **then**
$DV \leftarrow DV\_Adjustment\ (P, W, S)$
$S' \leftarrow Subsize\_Adjustment\ (P, DV, S)$
else
$DV \leftarrow W, S' \leftarrow S$
end if
$P \leftarrow Association\_Selection\ (P, DV, S')$
end for
Find all the non-dominated and feasible solutions and output them

calculated by the following form:

$$CV\ (x) = \begin{cases} TASC\ (x) - M & TASC\ (x) > M \\ 0 & TASC\ (x) \leq M \end{cases} \tag{23}$$

It is obvious that the smaller is the $CV\ (x)$, the better is the quality of x, and the CV of a feasible solution is always 0.

The pseudo code of the constrained tournament selection procedure is given in Algorithm 4. Given any two solutions $p1$ and $p2$, the better one is chosen as the mating parent. When both of $p1$ and $p2$ are feasible, the one which dominates the other is chosen. If $p1$ and $p2$ are non-dominated solutions, we choose one at random. When at least one of $p1$ and $p2$ are infeasible, the one with a smaller CV is chosen. If both of $p1$ and $p2$ have the same constraint violation value, we choose one at random. In order to select other mating parents, another pair of solutions is randomly chosen, so on and so forth, until N solutions are selected.

2) ADJUSTMENT FOR THE DIRECTION VECTORS

With the update of the population, many ineffective direction vectors may exit in MAOPs with irregular Pareto Front (PF). And much search effort will be unavoidably wasted since many single objective subproblems will share the same optimal solutions. These ineffective direction vectors are needed to be adjusted. Based on this consideration, direction vectors are adjusted according to the population at each generation. The direction vectors are adjusted by calling Algorithm 6.

In Algorithm 6, the effective and ineffective direction vectors are first detected. The detection of the effective direction vectors is given in Algorithm 5. Each nondominated solution is associated with a direction vector. If a direction vector

Algorithm 4 Constrained Tournament Selection Procedure

Input: population P and Constrained condition CV
Output: mating parent P'

$tour1 \leftarrow randperm(N), tour2 \leftarrow randperm(N)$
for $i \leftarrow 1$ to N **do**
$p1 \leftarrow tour1\ (i)\ , p2 \leftarrow tour2\ (i)$
if $CV\ (p1) \leq 0 \& CV\ (p2) \leq 0$ **then**
if $p1 \prec_\theta p2$ or $p2 \prec_\theta p1$
$P'\ (i) \leftarrow p1$ or $P'\ (i) \leftarrow p2$
else
$P'\ (i) = RANDOM\_PICK\ (p1, p2)$
end if
else
if $CV\ (p1) < CV\ (p2)$ or $CV\ (p1) > CV\ (p2)$
$P'\ (i) \leftarrow p1$ or $P'\ (i) \leftarrow p2$
else
$P'(i) = RANDOM\_PICK(p1, p2)$
end if
end if
end for

contains no associated nondominated solutions, this direction vector is considered to be ineffective; otherwise it is considered to be effective.

Algorithm 5 Detection of the Effective Direction Vectors (DETECTION)

Input: P : current population;
$W = \{w^1, \cdots w^K\}$: direction vectors;
Output: effective direction vectors DV

$P_{dom} = NONDOMINATED\ (P)$
for each x^i **in** P_{dom} **do**
$k \leftarrow \arg\min_{w^k} \arccos\left(\frac{x^i \cdot w^k}{\|x^i\| \|w^k\|}\right)$
$P_k \leftarrow P_k \cup \{x^i\}$
end for
for $k = 1$ to K **do**
if $P_k \neq \emptyset$ **then**
$DV \leftarrow DV \cup w^k$
end if
end for

After detecting effective direction vectors DV, the rest $(K - |DV|)$ direction vectors are generated by inserting new ones at the midpoints between every two effective direction vectors. It can be known that possible pairs of effective direction vectors, whose indexes can be denoted by

$$Pair = \{(1, 2)\ , \cdots, (i, j)\ , (|DV| - 1, |DV|)\}\ . \tag{24}$$

The number of the solutions in each subregion can be denotes as $Subreg \leftarrow \{Subreg_{(1,2)}, \cdots, Subreg_{(i,j)}, Subreg_{(|DV|-1,|DV|)}\}$. There are $|DV| - 1$ subregions in the objective space. The subregion index set, denoted by Len, in which the number of solution is greater than or equal to 2^*S. $|Len|$ new direction vectors are generated by inserting midpoints of

Algorithm 6 Adjustment for Direction Vectors (DV_Adjustment)

Input: P : current population;
$W = \{w^1, \cdots, w^K\}$: direction vectors;
Output: direction vectors DV

1 $DV \leftarrow DETECTION\ (P, W)$
2 $P_{dom} = NONDOMINATED(P)$, $S = \lfloor |P|/|W| \rfloor$
3 **While** $|DV| < K$ **do**
4 $Subreg \leftarrow \{Subreg_{(1,2)}, \cdots, Subreg_{(i,j)}, Subreg_{(|DV|-1,|DV|)}\}$//determine the number of nondominated solutions in each subregion
5 $Len \rightarrow find\ (Subreg \geq 2^*S)$, $r \rightarrow |Len|$ or $r \leftarrow K - |DV|$
6 **for** $k = 1$ **to** r // Generate new direction vectors
7 $w^* \leftarrow \left(w^{Subreg(k).1} + w^{Subreg(k).2}\right)/2$
8 $DV = DV \cup w^*$
9 **end for**
10 $Pair \leftarrow \{(1,2), \cdots, (i,j), (|DV|-1, |DV|)\}$
11 **for** $i = 1$ to $|DV|$ **do**
12 **for** $j = i + 1$ to $|DV|$ **do**
13 $d_{i,j} \leftarrow \left|w^i - w^j\right|$
14 **end for**
15 $d_{\min}(i) \leftarrow \min\left(d_{i,j}\right)$
16 **end for**
17 $d_{\max} \rightarrow \max\left(d_{\min}(1), \cdots, d_{\min}(|DV|)\right)$
18 $[d, I] \leftarrow sort\ (d)$, $Pair \leftarrow Pair\ (I)$
// Select $K - |DV|$ or $|Mid|$ pairs of the direction vectors
19 $Mid \leftarrow find(d == d_{\max})$, $r \leftarrow \max(Mid)$, $l \leftarrow \min(Mid)$
20 **While** $r - l + 1 < K - |DV| \& l > 1$
21 $r{+}{+}, l{-}{-}$
22 **end**
23 **for** $k = 1$ **to** r
24 $w^* \leftarrow \left(w^{Pair(k).1} + w^{Pair(k).2}\right)/2$
25 $DV = DV \cup w*$
26 **end for**
27 **end for**

vector pairs (Line 5-Line 9). Most of time, $|Len|$ is less than $K - |DV|$. Therefore, another method to find the new direction vectors will be introduced in following paragraph.

The distance of every *ith* direction vector to its nearest neighbor, denoted by $d_{\min}(i)$, is computed (lines 11–16). The pair index set, denoted by Mid, that has the maximum distance $d_{\max}$ out of all possible $d_{\min}(i)$, is selected (lines 17–19). The range between the low index of Mid and the upper index of Mid are expanded, first move l backwards and then move r forwards, until the range $|r - l + 1|$ reaches $K - |DV|$ (lines 19–22). $K - |DV|$ new direction vectors are generated by inserting midpoints of (upper-lower) vector pairs in DV (lines 23–26).

3) ADJUSTMENT FOR THE SIZE OF EACH SUBPOPULATION

After the direction vectors are adjusted, the number of solutions S_k for subregion Ω_k is needed to determine. The pseudo code of this update procedure is present in Algorithm 7. At lines 1, the solution set P is allocated to each subregion according to direction vectors W. In order to select the next generation solutions from the double population, S_k can be set to $|P_k|/2$ in each subregion. If $|P_k| \leq 1$, S_k is set to 1. When the whole population size $\sum_{k=1}^{K} S_k$ is larger than $|P_k|/2$, it indicates some subproblems have been over-crowed. So the number of solution S_k in each subregion is set to m. When the whole population size $\sum_{k=1}^{K} S_k$ is less than $|P_k|/2$, the number of solution S_k is updated at lines 12-14.

Algorithm 7 Adjustment for the Size of Each Subpopulation(Subsize_Adjustment)

Input: P : current population;
$W = \{w^1, \cdots, w^K\}$: direction vectors;
m : the size of the subpopulation
Output: number of the solution in subregions S

1 $[P_1, \cdots, P_K] = Assocation\ (P, W)$
2 **for** $i = 1$ **to** K **do**
3 **if** $P_k == \emptyset$ **or** $|P_k| == 1$ **then**
4 $S_k \leftarrow 1$
5 **else**
6 $S_k \leftarrow \lfloor |P_k|/2 \rfloor$
7 **end if**
8 **end for**
9 **if** $\sum_{k=1}^{K} S_k > |P_k|/2$ **then**
10 the number of solution S_k in each subregion is set as m
11 **else**
12 Sort $S_k\ (k = 1, \cdots, K)$, such that $0 < S_{k_i} \leq S_{k_{i+1}} \cdots \leq S_{k_K}$;
13 **While** $\sum_{k=1}^{K} S_k < |P|/2$ **do**
14 $S_{k_i} \leftarrow S_{k_i} + 1, i \leftarrow i + 1$;
15 **end**
16 **end if**

4) ASSOCIATION SELECTION

After combining the parent population P with the offspring population Q, association selection(Algorithm 8) is called, where solutions are selected from the merged population as follow.

First and foremost, each solution is associated with its closest direction vector. After that, the remaining $S_k - |P_k|$ solutions are randomly chosen from the combined population when the number of solutions of P_k is less than S_k. When the number of solutions of P_k is greater than S_k, we select S_k

Algorithm 8 Association Selection

Input: P : current population;
$W = \{w^1, \cdots, w^K\}$: direction vectors;
$S = \{S_1, \cdots S_k\}$: number of the solution in each subregion;
Output: the selected population Q
1 $[P_1, \cdots, P_k] = Assocation\,(P, W)$ // the codes are the same as Line2-Line5 in Algorithm 5;
2 **for** $k = 1$ **to** K **do**
3 **if** $|P_k| < S_k$ **then**
4 randomly select $S_k - |P_k|$ solutions from P_k and add them to Q
5 **end if**
6 **if** $|P_k| > S_k$ **then**
7 $P_k^{infeasible} \leftarrow \max(P_k) + CV_{Infeasible}$
8 $P_k \leftarrow P_k^{feasible} \cup P_k^{Infeasible}$
9 $(FrontNo, MaxFont) \leftarrow ENS\_SS\,(P_k, S_k)$,
10 $F_l \leftarrow P(MaxFront)$, Distance $\leftarrow$ CrowingDistance(F_l)
11 $P_s \leftarrow Select(Front, Dis\tan ce)$//Select solutions from P_k by non-dominated sorting and crowding distance and add them to Q
12 **end if**
13 **end for**

solutions from P_k by non-dominated sorting and crowding distance. It is better to select the solutions which have small acute angles with the direction vector in the objective space, since they can conduct to a better front for the subregion Ω. Thus, the method to allocate the remaining individuals in this situation can be defined as the following steps.

Step1. Find the infeasible solutions from the current population, and calculate the new objective value of the infeasible solutions (Line 7 in Algorithm 8). The objective value can be formulated as:

$$f\left(x^i\right) = \max(f\,(P)) + CV(x^i) \tag{25}$$

Step2. A non-dominated sorting procedure is used to rank the solutions P_k by the ENS_SS method [39], and the solutions with the last acceptable rank are determined (lines 9–10).

Step3. Solutions from the best ranks are chosen until all members of the last acceptable rank cannot be all chosen. The remaining solutions having worse ranks are deleted. Solutions from the last acceptable rank are assigned a crowding distance value based on their sparsity in the objective space, and solutions with higher crowding distance values are chosen.

5) COMPUTATIONAL COMPLEXITY

The computational time complexity of MOEA/D-ADV is dependent mainly on three main components: adjustment for the direction vectors, adjustment for the size of the subpopulations and association selection. The adjustment for direction vectors (Algorithm 6) includes two steps: 1) the detection of the effective direction vector (Algorithm 5) and 2) the subsequential adjustments. The computational complexity of direction vectors adjustment is $O\,(mNK)$, where m is the number of objectives, N is the current population size, K is the number of the subproblems. The computational of association selection (Algorithm 8) is $O\left(mN^2/K\right)$, while the computational of adjustment for the size of the subpopulations (Algorithm 7) is $O\,(NK)$.In summary, the worst computational complexity of MOEA/D-ADV within one generation is $O\,(mNK)$.

V. EXPERIMENT SIMULATION AND RESULT ANALYSIS

To explain the effectiveness of the proposed partition parameter design method, this section presents a number of experiments aiming at comparing the performance of our partition parameter design method. For the sake of generality, all timing parameters are expressed in generic time units, which are microsecond. In experiments, it is assumed that T_q in (11) is equal to 1.

A. EVALUATION OF THE ALGORITHM TO CALCULATE THE MINIMUM EXECUTION TIME OF PARTITION

The algorithm 2 is used to calculate the minimum execution time of partition while simultaneously guaranteeing tasks schedulability. To illustrate the effectiveness of the algorithm, this section presents two experiments aimed at comparing the performance of the algorithm.

Consider an ARIN653 partition consisting of three tasks, where $P_1 = (2, 35)$, $P_2 = (3, 45)$, $P_3 = (2, 50)$. Algorithm 2 is used to calculate the minimum execution time and bandwidth in different partition periods, and the range of period is $5 \le \Pi \le 100$. In the first experiment, we compare the bandwidth obtained by our algorithm with the algorithm in [40], which is denoted as Exhaust in the plot. The bandwidth of partition is represented as follows:

$$BandWith = \sum_{i=1}^{N} \frac{c_1^*\delta + c_2^*\Theta_i}{\Pi_i} \tag{26}$$

where $\delta = 1, c_1 = 1, c_2 = 1$.

To calculate the worst-case response time of the task, R.I.Davi gave an iterative formula for computing the worst execution time of hierarchical scheduling [41].

$$L_i\,(w) = C_i + \sum_{\forall j \in hp(j)} \left\lceil \frac{w}{T_j} \right\rceil C_j \tag{27}$$

$$w = L_i\,(w) + \left\lceil \frac{L_i\,(w)}{\Theta} \right\rceil (\Pi - \Theta) \tag{28}$$

where $L_i\,(w)$ is the task load, $\left\lceil \frac{L_i(w)}{\Theta} \right\rceil (\Pi - \Theta)$ is the gaps of the task, and $hp(j)$ is the set of tasks with priorities higher than task i. Recurrence begins with $w_i^0 = C_i + \left\lceil \frac{C_i}{\Theta} \right\rceil (\Pi - \Theta)$ and ends when $w_i^{n+1} > D_i$ in which case the task is not schedulable or when $w_i^{n+1} = w_i^n$ in which case w_i^{n+1} gives the worst-case response time of the task.

As shown in Fig. 3, the worst respond time of three tasks are less than the deadline of tasks, so the tasks are schedulable

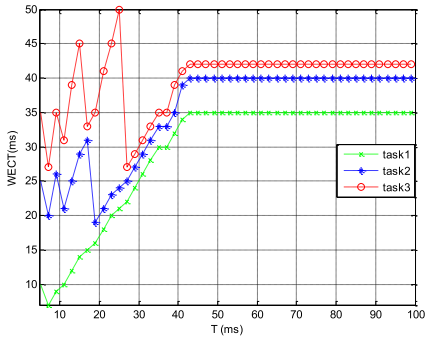

FIGURE 3. Worst-case response time of the task with different partition periods.

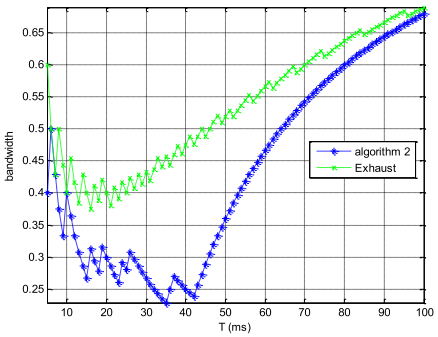

FIGURE 4. Bandwidth of partition obtained by algorithm 2 and Exhaust.

on the partition which is designed by algorithm 2. In Fig. 4, the minimum bandwidths calculated by the two algorithms are plotted as a function of partition period. The results indicate that the bandwidths achieved by algorithm 2 are smaller than bandwidths achieved by Exhuast. With the increase of system utilization, the difference between the algorithm 2 and Exhuast reduces. For algorithm 2, the bandwidth increases rapidly when $\Pi \geq 45$, while the bandwidth does not change much when $15 \leq \Pi \leq 45$. Without considering partition scheduling, it is a great option to set partition period as $T_{\min}$.

In the second experiment, we show the simulation results for our proposed algorithm and compare it with the heuristic algorithm proposed by Dewan and Fisher [16], the algorithm presented by Yoon *et al.* [22] and the sufficient algorithm proposed by Shin I [12]. These algorithms are denoted as Heuristic, GP and Suff respectively in the plots. The experiment parameters and value ranges are shown below:

1) The number of tasks executed in the partition is taken from the set $\{4, 6, 8, \cdots, 18\}$.

2) The tasks utilization is taken from the range [0.1, 0.7] at 0.05-increments and individual task utilizations are generated using UUniFast algorithm [41].

3) Task period T_i is generated randomly in the interval [20, 100] and is subject to a logarithmic uniform distribution.

4) The parameter k is set to be 3 for Suff, while the parameter Π is set to be the minimum period of the tasks for algorithm 2 and Heuristic.

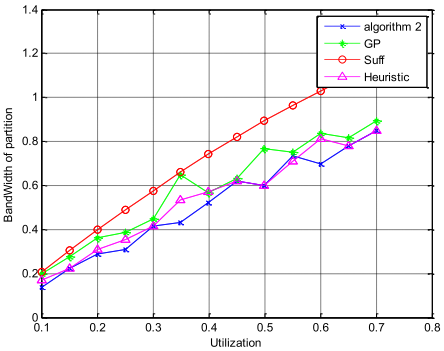

FIGURE 5. Bandwidth of partition with different task system utilizations.

In Fig. 5, the bandwidths of partition calculated by the four algorithms are plotted as a function of task system utilization. The performance of algorithm 2 and Heuristic are better than GP than Suff. In most cases, the bandwidths obtained by algorithm 2 are less than the bandwidths obtained by compared algorithms When task system utilization is greater than 0.5, the bandwidths obtained by Suff are larger than 1. The bandwidths obtained by algorithm 2 and Heuristic are almost same. With the increase of system utilization, the difference between the algorithm 2 and Suff becomes larger.

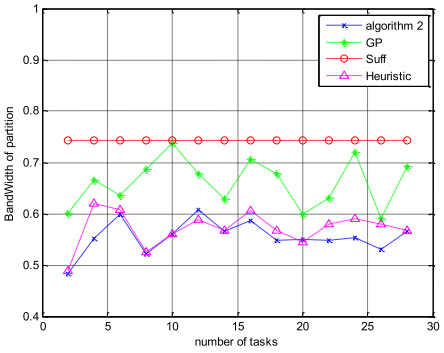

FIGURE 6. Bandwidth of partition with different task system sizes.

The Fig. 6 shows the effect of task system size on the bandwidth for the four algorithms. In this plot, system utilization is $u = 0.4$, and task system size is varied from 2 to 28. It can be seen that the task system size has no influence on bandwidths obtained by Suff. The task system size increases from 2 to 15, and the bandwidths obtained by algorithm 2 and Heuristic are almost same. In some cases, Heuristic performs better than algorithm 2. When task system size is greater than 15, algorithm 2 perform better than other algorithms. The result shows that the task system size does not have a significant effect on algorithm 2 and Heuristic. This is due to the fact that for algorithm 2 and Heuristic, bandwidths depend on the minimum period of the tasks. From the experiment, it is found that the algorithm 2 works well in both task system utilization and task system size increasing cases.

Since the supply-bound function is a discrete function, it is difficult to obtain the minimum execution time through a

numerical analysis. GP and Suff use the linear low-bound function to obtain the minimum execution time through the numerical analysis, while algorithm 2 and Heuristic use the supply-bound function to obtain the minimum execution time through heuristic method. Therefore, the bandwidth achieved by GP and Suff is greater than the bandwidth achieved by algorithm 2 and Heuristic. In Heuristic, Request-bound function is replaced by approximate cumulative request-bound function to reduce the number of points in the testing set. This may affect the bandwidths obtained by Heuristic. Therefore, in many cases, algorithm 2 performs better than Heuristic. Although the algorithm 2 performs better than the compared algorithms, the computation complexity of algorithm 2 is higher than the compared algorithms.

B. EVALUATION OF THE PARTITION PARAMETER DESIGN METHOD

In this section, we present simulation results and compare the performance of our proposed partition parameter design method with the exact algorithm proposed by Xiaoguang *et al.* [21], Tianran and Xiong [42] which is denoted as POA.

1) EXPERIMENT OF PARTITION PARAMETER DESIGN

In the first experiment, we considered the application shown in Table 2, consisting of three partitions. We should design the parameters of three partitions to confirm that the partitions are schedulable on the same processor.

TABLE 2. Partitions within system and task parameters within partitions.

ID	Tasks		
	C (ms)	T (ms)	D (ms)
G1	2	20	20
	4	25	25
	1	50	50
G2	4	40	40
	4	50	50
G3	5	100	100
	6	150	150
	5	200	200

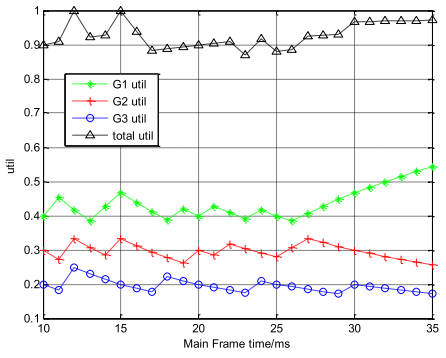

FIGURE 7. System schedulability partition parameters.

For the application given in Table 2, the POA algorithm is used to search the partition parameters satisfying the system scheduling. The results are shown in Fig. 7. From the figure, it is found that the minimum value of the bandwidth requirement of the partitions is $\sum_{j=1}^{3} \alpha_j = 0.87$, while the max value of the major frame time is 35. In order to reduce the task context switching, the major frame time should be larger. Therefore, for the system given above, the major frame time of the system is $MainFrame = 35$, and the bandwidths of three partitions are $\alpha_1 = 0.54, \alpha_2 = 0.25, \alpha_3 = 0.17$. Equation (15) is used to obtain the system overhead which is $F_2 = 0.628$.

For the MOEA/D-ADV, constraint condition of the parameter Π in the objective function F_1 and F_2 is $20 \leq \Pi \leq 60$. The parameters of MOEA/D-ADV algorithm are set as follows:

- The population size is 80;
- Maximum number of iterations is 100;

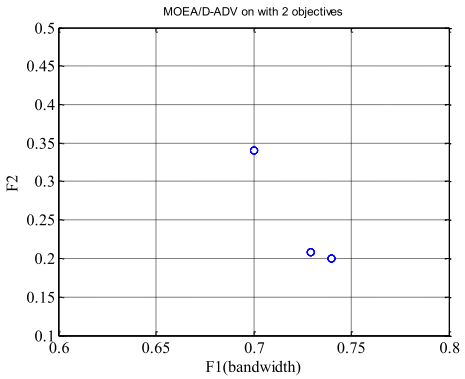

FIGURE 8. Pareto optimal solutions obtained by MOEA/D-ADV algorithm.

Pareto optimal solutions obtained by MOEA/D-ADV are shown in Fig. 8, and the minimum value of the required bandwidth of the partitions is $\sum_{j=1}^{3} \alpha_j = 0.7$. To compare with the POA algorithm, we select a Pareto optimal solution with the smallest value of F_1, where $G1 = (25, 8), G2 = (50, 14), G1 = (100, 14), \alpha_1 = 0.32, \alpha_2 = 0.28, \alpha_3 = 0.14, F_2 = 0.34$. The major frame time of the system is $MainFrame = 100$. Compared with POA algorithm, both the system overhead and the total bandwidth of the system designed by our algorithm are smaller. This is because the periods of partitions designed by POA algorithm are the same. In this experiment, the minimum period among all tasks in G1 is 20 ms, while the minimum period in G3 is 100 ms. The periods of G1 and G3 designed by POA algorithm are 35 ms, For G1, the partition period is too large, while for G3, the partition period is too small. Time wasted in context switches performed by the global scheduler is high. The period of G1 and G3 designed by our algorithm are 25 ms and 100 ms, which are close to the minimum period among all tasks in partitions. If the difference of minimum period among tasks

within different partitions is large, for the system designed by the POA algorithm, time wasted in context switches performed by the global schedule will be high. Therefore, the more difference between minimum periods among the tasks within different partitions is, our algorithm performs better. For multiprocessor systems, the POA algorithm is not suitable for partition parameter design, while our algorithm can be extended to multiprocessors system design by modifying the limitation of processor number.

2) ALGORITHM PERFORMANCE EVALUTION

When designing partition parameters, it is hoped that both the total bandwidth requirement and the system overhead are small. For this test, the total bandwidth requirement and the system overhead designed by two algorithms is compared. The experiment parameters and value ranges are shown below:

1) The number of partitions executed on the processor is taken from the set $\{2, 4, 6\}$.

2) The task system utilization is taken from the range $[0.3, 0.7]$ at 0.1-increments and individual task utilizations are generated using UUniFast algorithm.

3) The generated tasks are randomly assigned to different partitions, and the number of tasks in one partition is less than 5.

In this experiment, according to given partition size and task system utilization, we randomly generate task parameters C_i and T_i for each task and the number of tasks in the partition. The MOEA/D-ADV and POA algorithm are implemented to obtain the partition parameters. Our comparison metrics is the ratio of the system overhead. For the plots which vary the system utilization, each point in the plots represents mean of 30 simulation runs.

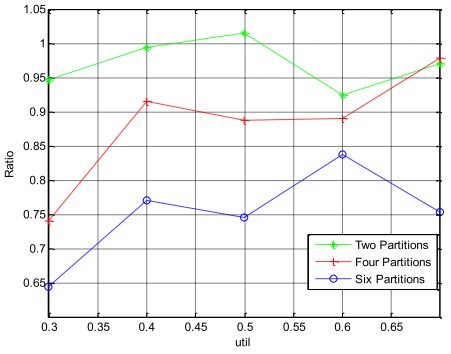

FIGURE 9. Ratio of the cost of system overhead between MOEA/D-ADV algorithm and POA algorithm.

In Fig. 9, the ratio of the system overhead for the two algorithms is plotted as a function of task system utilization, where ratio denotes the ratio of the system overhead between MOEA/D-ADV and POA algorithm. In two partitions case, the ratio is almost equal to 1. In four partitions case and six partitions case, the ratios are less than 1, and the ratio in four partitions case is larger than that in the six partitions case. Therefore, the ratios decrease with the increase in the number of partitions, while the ratios increase with the increase the system utilization. It means that our algorithm performs better with increasing the number of the partition.

3) SUCCESS RATIO EVALUATION

In this subsection, we focus on the evaluation of the proposed approach in terms of success ratio, i.e., the percentage of partition sets designed by POA and MOEA/D-ADV to be schedulable upon a limited processors platform. A higher success ratio indicates a more accurate and useful method. To this end, a series of partition sets with cardinality from 4 to 12 are generated, and the system utilization is 0.6. For the exact algorithm, if the bandwidth requirement is greater than $1(\sum_{j=1}^{n} \alpha_j > 1)$, the partitions are not schedulable. For the MOEA/D-ADV algorithm, if the number of required processors by TASC is greater than 1, the partitions are not schedulable.

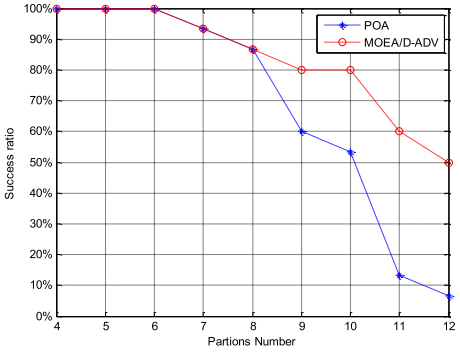

FIGURE 10. Success ratios of MOEA/D-ADV and POA when the system utilization is 0.6.

Fig. 10 presents the success ratio of POA algorithm and MOEA/D-ADV algorithm. At first glance, success ratios decrease with the increase of partition sets cardinalities. While the number of partition increases, the required bandwidth of partitions also increases. Then it may be hard to find the parameters to satisfy the partition scheduling condition. Both of the algorithms perform well when a number of partitions are less than 7, nevertheless, our approach performs better than POA when the number of partitions is larger than 7. It can be found that the difference in success ratios between POA and our approach enlarge gradually while the number of partitions increases. The largest difference values of success ratios are 43%.

In summary, it can be observed that the partition parameters designed by our method are better than that designed by the POA algorithm, on all the experiments. The partition parameters designed by our method have smaller bandwidth and the system overhead. The computational complexity of our method is $O(2NKGMT)$, where N is the current population size, K is the number of the subproblems, G is the number of the iterations, M is the number of the partitions, and T is the maximum period of tasks. The computational complexity

of POA algorithm is $O\left(M^2T^2\right)$. However, the computational complexity of our method is greater than POA algorithm.

C. EVALUATION OF THE MOEA/D-ADV METHOD

To illustrate the effectiveness of the MOEA/D-ADV algorithm, in this section, MOEA/D-ADV is compared with nine multi-objective evolutionary algorithms: 1) MOEA/D-M2M; 2) NSGAII-CDP; 3) MOEA/DD [43]; 4)MOEA/D-SR [44]; 5) SPEA2-CDP [31], [45]; 6) C-RVEA [46]; 7) NSGAIII-CDP [47]; 8) VaEA [48]; 9) HypE. In this experiment, five groups of partition sets will be generated according to the method in Section V-B. The experiment parameters and value ranges are shown below:

1) The number of partitions executed on the processors is taken from the set {8, 20, 24, 32, 40} and system utilization is taken from the set {1.5, 3, 4.2, 4.8, 6.0}. The number of processors is as follow 3, 5, 6, 8, 9. According to the number of partitions and processors, the test instances are denoted as: M3P8, M5P20, M6P24, M8P32, M9P40.

2) The population size of all algorithms is 100, and the maximum number of iterations is 300. All algorithms use the same analog binary crossover operator and polynomial mutation operator.

3) Probability of selecting individuals in the neighborhood is $\delta = 0.9$;

4) Parameters setting in MOEA/D-ADV and MOEA/D-M2M are $K = 50$ and $S = 2$;

5) Parameter of MOEA/D-SR is $S_r = 0.005$;

6) Parameter of CRVEA is $f_r = 0.01$;

The performance of the multi-objective algorithm is evaluated by hyper volume (HV). The hyper volume evaluates the quality of Pareto optimal solution set by calculating the volume of the hypercube surrounded by the given reference points. Hyper volume is a comprehensive measure of the approximation, comprehensive and uniformity of the algorithm. The larger the value of the hyper volume is, the better the performance of the algorithm is. The value of the reference point for calculating hyper volume is set to the maximum value of the union of Pareto optimal solution sets obtained by ten algorithms on each optimization objective.

In the experiment, MOEA/D-ADV and all the comparing EMO algorithms independently run 15 times for each test problem. The performance of ten compared algorithms in terms of HV is presented in Table 3, where the performance of the algorithm with the best mean HV value is highlighted in boldface. It can be observed that, MOEA/D-ADV significantly outperforms other compared algorithms on all the test problems in terms of HV mean, except for M3P8. NSGAIII-CDP achieves the best performance on M3P8. It should be pointed out that there is no significant difference among the performance of ten compared algorithms on M3P8. On the other hand, the standard deviation of MOEA/D-ADV is obviously smaller than other nine algorithms, which indicates that the performance of MOEA/D-ADV is more stable. From the simulation result, it can be observed that MOEA/D-ADV is significantly better than other nine algorithms.

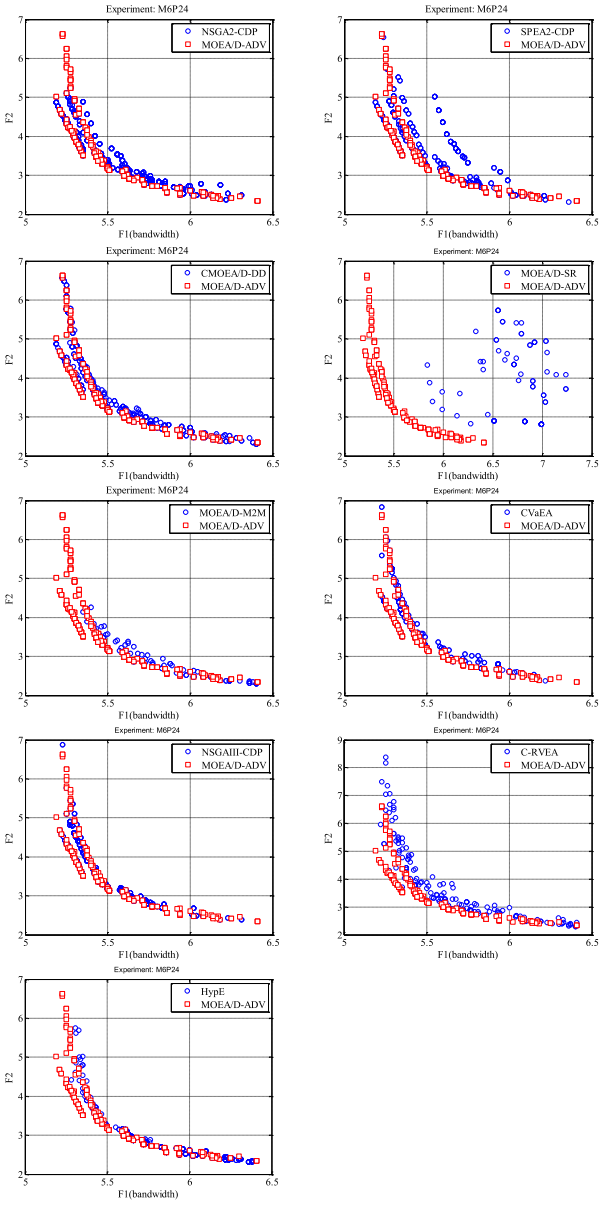

FIGURE 11. Pareto optimal solutions obtained by ten compared algorithms (partition number: 24, system utilization: 4.2, processor number: 6).

To visually view the performance of those algorithms, the Pareto optimal solutions on M6P24 obtained by the ten compared algorithms are plot in Fig. 11. As can be seen, solutions distribution of MOEA/D-SR and SPEA2-CDP are worse than other algorithms. Compared with other algorithms, the solutions of MOEA/DD, NSGAIII-CDP, CVaEA and MOEA/D-ADV are distributed more uniformly. It is obvious that MOEA/D-ADV has the best diversity and convergence among the ten algorithms.

VI. CONCLUSION AND FURTHER WORK

While the IMA architecture has great advantages, it is quite challenging for system integrators to integrate partitions

TABLE 3. Mean values of HV mean, obtained by MOEA/D-ADV, NSGAII-CDP, SPEA2-CDP, NSGAIII-CDP, C-RVEA, MOEAD-SR, CVaEA, MOEAD-M2M, CMOEAD/D, and HypE.

Test Instances		MOEA/D-ADV	NSGA-II-CDP	SPEA2-CDP	NSGAIII-CDP	C-RVEA	MOEA/D-SR	CVaEA	MOEAD-M2M	CMOEA/DD	HypE
M3P8	$\bar{H}$	2.4652	2.4633	2.4625	**2.4734**	2.4302	2.4172	2.4574	2.4266	2.4723	2.3538
	σ	**0.0017**	0.086	0.0845	0.0095	0.0829	0.0845	0.0081	0.0029	0.0058	0.0905
M5P20	$\bar{H}$	**7.9019**	7.8776	7.518	7.8756	7.6147	4.1358	7.8334	7.6791	7.7477	7.7199
	σ	0.1064	0.1508	0.2071	0.1079	0.1987	1.4750	0.1437	0.1356	0.2695	**0.1046**
M6P24	$\bar{H}$	**11.2811**	11.0739	10.9073	11.1067	10.807	3.3285	11.1102	10.6832	11.1918	10.9015
	σ	**0.1297**	0.2803	0.2681	0.2305	0.2305	2.2760	0.2829	0.2529	0.2660	0.2002
M8P32	$\bar{H}$	**14.6333**	13.4564	13.9408	13.9331	13.4586	2.2981	13.7643	13.8182	13.7591	13.5699
	σ	**0.0824**	0.6549	0.2681	0.2793	0.2611	2.4033	0.1729	0.1652	0.4796	0.2681
M9P40	$\bar{H}$	**26.4707**	23.0059	23.5590	24.1151	24.2661	9.9223	23.9480	26.0182	23.4135	24.3633
	σ	**0.5982**	1.7248	0.9793	1.0822	0.6575	2.7183	0.9007	1.0824	1.7812	0.8601

into IMA architecture efficiently. In this paper, the sufficient schedulability condition of partitions scheduling is proposed, which can be applied to industrial applications such as TTEthernet protocol. The complexity of the sufficient schedulability condition for harmonic period partitions is linear, while that for non-harmonic period partitions is pseudo-linear. If partitions are designed as non-harmonic period partitions, it will be hard to assign partitions to a number of processors. We suggest designing IMA partitions as harmonic period partitions.

TABLE 4. List of acronyms.

Acronym	Meaning
IMA	Integrated Modular Avionics
ARINC	Aeronautical Radio, Incorporated
MOEA/D	Multi-objective Evolutionary Algorithm Based on Decomposition
GCD	Greatest Common Divisor
RM	Rate Monotonic policy
CDP	Constraints Domination Principle
SR	Stochastic Ranking
ADV	Adjustment for the Direction Vectors
MOEA/D-CDP	MOEA/D Algorithm Based on Constraints Domination Principle
MOEA/D-SR	MOEA/D Algorithm Based on Stochastic Ranking
MOEA/D-M2M	decomposition-based multi-objective evolutional algorithm
RVEA	a reference vector guided evolutionary algorithm

In this paper, the MOEA/D-ADV is introduced to solve the problem of partition parameter design. Our proposed method can be extended to multiprocessor systems design by modifying the limitation of processor number. The experiment results show that the solutions obtained by MOEA/D-ADV have better convergence and diversity than compared algorithms. However, there are two limitations of our research:

(1) Although the partition parameters designed by our method are better than that designed by the POA algorithm, the method is more complex and computationally expensive.

(2) The mainly goal of this paper is to propose methods to optimize the partition parameters under the schedulability constraints of partitions and tasks. However, in the optimization procedure, we use the sufficient condition to determine the schedulability of partitions.

There are several possible directions for further work. First, to improve the analysis precision of Algorithm 1, we will investigate how to assign feasible start time of partition more effectively instead of directly choosing the first suitable time unit. Second, we would like to study how to distribute the designed partitions to different processors, in order to meet the real-time and reliability requirements of IMA.

APPENDIX

See Table 4.

REFERENCES

[1] *Avionics Application Software Standard Interface*, document ARINC653P1-2, Aeronautical Radio, Annapolis, MD, USA, Mar. 2006, pp. 2–45.

[2] J. Korst, E. Aarts, J. Lenstra, and J. Wessels, "Periodic multiprocessor scheduling," in *Parallel Architectures and Languages Europe*, vol. 505. Berlin, Germany: Springer, 1991, pp. 166–178.

[3] O. Kermia, "An efficient approach for the multiprocessor non-preemptive strictly periodic task scheduling problem," *J. Syst. Archit.*, vol. 79, pp. 31–44, Sep. 2017.

[4] F. Eisenbrand, K. Kesavan, R. S. Mattikalli, M. Niemeier, A. W. Nordsieck, M. Skutella, J. Verschae, and A. Wiese, "Solving an avionics real-time scheduling problem by advanced IP-methods," in *Proc. ESA*. Berlin, Germany: Springer, 2010.

[5] O. Kermia and Y. Sorel, "Schedulability analysis for non-preemptive tasks under strict periodicity constraints," in *Proc. 14th IEEE Int. Conf. Embedded Real-Time Comput. Syst. Appl.*, Aug. 2008, pp. 25–32.

[6] J. Chen, C. Du, F. Xie, and Z. Yang, "Schedulability analysis of non-preemptive strictly periodic tasks in multi-core real-time systems," *Real-Time Syst.*, vol. 52, no. 3, pp. 239–271, May 2016.

[7] T. Zhang, N. Guan, Q. Deng, and W. Yi, "Start time configuration for strictly periodic real-time task systems," *J. Syst. Archit.*, vols. 66–67, pp. 61–68, May 2016.

[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] T.-W. Kuo and C.-H. Li, "A fixed-priority-driven open environment for real-time applications," in *Proc. 20th IEEE Real-Time Syst. Symp.*, Madrid, Spain, Dec. 1999, pp. 256–267.
[10] G. Lipari and S. K. Baruah, "Efficient scheduling of real-time multi-task applications in dynamic systems," in *Proc. 6th IEEE Real-Time Technol. Appl. Symp. (RTAS)*, May/Jun. 2000, pp. 166–175.
[11] G. Lipari and E. Bini, "Resource partitioning among real-time applications," in *Proc. 15th Euromicro Conf. Real-Time Syst.*, Jul. 2003, pp. 151–158.
[12] I. Shin and I. Lee, "Compositional real-time scheduling framework with periodic model," *ACM Trans. Embedded Comput. Syst.*, vol. 7, no. 3, pp. 1–39, Apr. 2008.
[13] X. Feng and A. K. Mok, "A model of hierarchical real-time virtual resources," in *Proc. 23rd IEEE Real-Time Syst. Symp. (RTSS)*, Dec. 2002, pp. 26–35.
[14] A. Easwaran, M. Anand, and I. Lee, "Compositional analysis framework using EDP resource models," in *Proc. 28th IEEE Int. Real-Time Syst. Symp. (RTSS)*, Dec. 2007, pp. 129–138.
[15] N. Fisher and F. Dewan, "A bandwidth allocation scheme for compositional real-time systems with periodic resources," *Real-Time Syst.*, vol. 48, no. 3, pp. 223–263, May 2012.
[16] F. Dewan and N. Fisher, "Bandwidth allocation for fixed-priority-scheduled compositional real-time systems," *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 4, pp. 1–29, Dec. 2014.
[17] M. M. H. P. van den Heuvel, R. J. Bril, and J. J. Lukkien, "Budget allocations for hierarchical fixed-priority scheduling of sporadic tasks with deferred preemptions upon EDP resources," *ACM SIGBED Rev.*, vol. 12, no. 1, pp. 19–27, Mar. 2015.
[18] A. Al Sheikh, O. Brun, P. E. Hladik, and B. J. Prabhu, "A best-response algorithm for multiprocessor periodic scheduling," in *Proc. 23rd Euromicro Conf. Real-Time Syst.*, Jul. 2011, pp. 228–237.
[19] A. Al Sheikh, O. Brun, P.-E. Hladik, and B. J. Prabhu, "Strictly periodic scheduling in IMA-based architectures," *Real-Time Syst.*, vol. 48, no. 4, pp. 359–386, Jul. 2012.
[20] C. Pira and C. Artigues, "Line search method for solving a non-preemptive strictly periodic scheduling problem," *J. Scheduling*, vol. 19, no. 3, pp. 227–243, Jun. 2016.
[21] G. Xiaoguang, X. Yayong, and W. Zengkui, "Task schedulability analyzing method of two-level hierarchical scheduling algorithm in integrated modular avionics," *Acta Aeronautica Et Astronautica Sinica*, vol. 36, no. 2, pp. 585–595, 2015.
[22] M.-K. Yoon, J.-E. Kim, R. Bradford, and L. Sha, "Holistic design parameter optimization of multiple periodic resources in hierarchical scheduling," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2013, pp. 18–22.
[23] H. S. Chwa, J. Lee, J. Lee, K.-M. Phan, A. Easwaran, and I. Shin, "Global EDF schedulability analysis for parallel tasks on multi-core platforms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 5, pp. 1331–1345, May 2017.
[24] I. Shin, A. Easwaran, and I. Lee, "Hierarchical scheduling framework for virtual clustering of multiprocessors," in *Proc. Euromicro Conf. Real-Time Syst.*, Jul. 2008, pp. 181–190.
[25] A. Burmyakov, E. Bini, and E. Tovar, "Compositional multiprocessor scheduling: The GMPR interface," *Real-Time Syst.*, vol. 50, no. 3, pp. 342–376, May 2014.
[26] J. P. Lehoczky and S. Ramos-Thuel, "An optimal algorithm for scheduling soft-aperiodic tasks in fixed-priority preemptive systems," in *Proc. Real-Time Syst. Symp.*, Phoenix, AZ, USA, Dec. 1992, pp. 110–123.
[27] J.-E. Kim, T. Abdelzaher, and L. Sha, "Schedulability bound for integrated modular avionics partitions," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Grenoble, France, Mar. 2015, pp. 37–42.
[28] 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, pp. 284–292, Sep. 1993.
[29] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[30] D. W. Corne, N. R. Jerram, J. D. Knowles, and M. J. Oates, "PESA-II: Region-based selection in evolutionary multiobjective optimization," in *Proc. 3rd Annu. Conf. Genetic Evol. Comput.*, 2001, pp. 283–290.
[31] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," in *Evolutionary Methods for Design, Optimisation and Control*. 2002, pp. 95–100.
[32] E. Zitzler and S. Kunzli, "Indicator-based selection in multiobjective search," in *Proc. 8th Int. Conf. Parallel Problem Solving Nature*, 2004, pp. 832–842.
[33] J. Bader and E. Zitzler, "HypE: An algorithm for fast hypervolume-based many-objective optimization," *Evol. Comput.*, vol. 19, no. 1, pp. 45–76, Spring 2011.
[34] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.
[35] H.-L. Liu, F. Gu, and Q. Zhang, "Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 450–455, Jun. 2014.
[36] H.-L. Liu, L. Chen, K. Deb, and E. D. Goodman, "Investigating the effect of imbalance between convergence and diversity in evolutionary multiobjective algorithms," *IEEE Trans. Evol. Comput.*, vol. 21, no. 3, pp. 408–425, Jun. 2017.
[37] Jordehi, and A. Rezaee, "A review on constraint handling strategies in particle swarm optimisation," *Neural Comput. Appl.*, vol. 26, no. 6, pp. 1265–1275, 2015.
[38] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Trans. Evol. Comput.*, vol. 4, no. 3, pp. 284–294, Sep. 2000.
[39] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "An efficient approach to nondominated sorting for evolutionary multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 2, pp. 201–213, Apr. 2015.
[40] R. I. Davis and A. Burns, "Hierarchical fixed priority pre-emptive scheduling," in *Proc. 26th IEEE Int. Real-Time Syst. Symp. (RTSS)*, Miami, FL, USA, Dec. 2005, pp. 389–398.
[41] E. Bini and G. C. Buttazzo, "Biasing effects in schedulability measures," in *Proc. 16th Euromicro Conf. Real-Time Syst. (ECRTS)*, Catania, Italy, 2004, pp. 196–203.
[42] T. Zhou and H. Xiong, "Design of energy-efficient hierarchical scheduling for integrated modular avionics systems," *Chin. J. Aeronaut.*, vol. 25, no. 1, pp. 109–114, Feb. 2012.
[43] K. Li, K. Deb, Q. Zhang, and S. Kwong, "An evolutionary many-objective optimization algorithm based on dominance and decomposition," *IEEE Trans. Evol. Comput.*, vol. 19, no. 5, pp. 694–716, Oct. 2015.
[44] M. A. Jan and R. A. Khanum, "A study of two penalty-parameterless constraint handling techniques in the framework of MOEA/D," *Appl. Soft Comput.*, vol. 13, no. 1, pp. 128–148, Jan. 2013.
[45] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, Nov. 1999.
[46] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "A reference vector guided evolutionary algorithm for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 5, pp. 773–791, Oct. 2016.
[47] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using Reference-Point-Based nondominated sorting approach, part I: Solving problems with box constraints," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 577–601, Aug. 2014.
[48] Y. Xiang, Y. Zhou, and M. Li, "A vector angle-based evolutionary algorithm for unconstrained many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 21, no. 1, pp. 131–152, Feb. 2017.

HUAKUN CHEN received the M.S. degree in navigation, guidance, and control from Northwestern Polytechnical University, in 2007, where he is currently pursuing the Ph.D. degree. His main research interests include integrated modular avionics, airborne networks, avionics fault diagnosis, and health management.

WEIGUO ZHANG was born in February 1956. He received the D.Eng. degree from Northwestern Polytechnical University. He currently supervises 11 master's degree students 19 Ph.D. degree students. He is currently a Professor with Northwestern Polytechnical University. He presided over a number of projects such as the National Natural Resources Fund and the Aviation Fund, and published more than 50 scientific research articles and two monographs. His main research works include active control, flight control system design and optimization, control allocation, non-linear control, and robust control.

YONGXI LYU was born in June 1990. He received the D.Eng. degree. He is an Assistant Researcher. He has participated in a number of research projects such as the National Natural Resources Fund, and he also published nearly ten papers in academic conferences and journals, including two SCI indexed four EI indexed. He holds two patents. His research interests include high-angle-of-attack flight control technology, high-angle-of-attack aerodynamic modeling technology, and multi-control plane control allocation technology.