Received January 15, 2020, accepted February 3, 2020, date of publication February 17, 2020, date of current version March 3, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.2974557*

Independent Temporal Integration of

ARINC653 Conformed Architecture

A Search Based Solution

KUI ZHANG1, JUNCHEN LIU1, JIAN REN1, JINGHUI HU2, AND CHAO LIU1



* State Key Laboratory of Software Development Environment, School of Computer Science and Engineering, Beihang University, Beijing 100083, China

2Aeronautical Key Laboratory for Digital Manufacturing Technology, AVIC Manufacturing Technology Institute, Beijing 100024, China

Corresponding author: Jian Ren (renjian@buaa.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61602021, and in part by the State Key Laboratory of Software Development Environment under Grant SKLSDE-2019ZX-10.

 **ABSTRACT** ARINC653-based integrated modular avionics (IMA) architecture has been widely adoptedin the design of modern civil and military aircraft. IMA imposes various requirements on the underlying operating system, of which the temporal and spatial separation requirements are essential to task allocation. In practice, nding the optimal allocation con gurations of tasks to enable processing modules to satisfy various temporal constraints is one of the greatest challenges. For that purpose, hundreds of tasks must be mapped into given processing modules, which has been proven to be a nonpolynomial problem. This paper introduces a search-based approach to aid in nding effective solutions for the task allocation problem in polynomial time. Two search techniques based on both population search (genetic algorithm) and neighbor search (simulated annealing), along with their multicore versions, are presented. A heuristic is designed speci cally to validate whether candidate solutions ful ll various constraints implied by IMA, and thus to evaluate the tness. Furthermore, the multicore version is designed to reduce the time delay of obtaining a new optimized con guration. The results show that both algorithms can ultimately nd optimized solutions with utility rates above 90% in all con gurations and can support the optimization over 100 tasks, which is an outstanding result. The result also reveals that simulated annealing can produce a better solution under limited resources, while the genetic algorithm will determine a valid solution within a shorter time period. Moreover, simulated annealing outperforms the genetic algorithm in terms of both effectiveness and ef ciency with respect to solving this allocation problem with complicated constraints.



 **INDEX TERMS** Temporal allocation, Heuristic model, IMA, system integration.



**I. INTRODUCTION**

In recent years, software integration has received much atten-tion from both industry and research institutions. During the integration process, many attributes must be taken into consideration, such as interface compatibility, coherence of the context, reliability, and temporal performance [1] [3]. The temporal attributes represent a crucial requirement that must be satis ed in a real-time system, since the correctness of output is not only determined by the result but also depends on the time the result was generated [4] [6].

Avionics represents one of the most important types of real-time systems, which is now becoming an obligatory component of modern aircraft [7]. Avionics have been used in many applications of aircraft, such as ight control [8],

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



autopilot, collision-avoidance [9], weather systems used in civil aircraft, weapons systems, electronic support mea-sures (ESM) and defensive aids systems (DAS) in the military, etc.

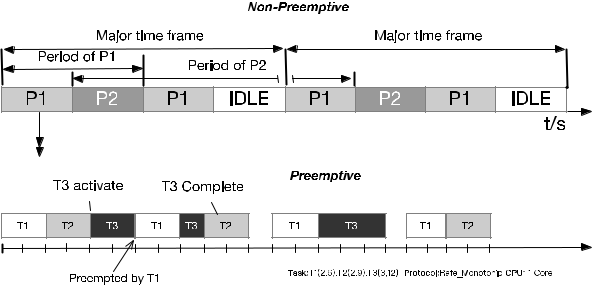
With the requirement of producing better performance while reducing weight, IMA architecture standardized by ARINC653 has been proposed and widely adopted in the design of modern avionics. It was rst presented by Honeywell as an Airplane Information Management Sys-tem (AIMS) on the Boeing 777 aircraft in 1995 [10]: after that, it has been extensively implemented in avionics designs of the US Air Force F-22 commonly integrated proces-sor [11], Airbus 380 [12], Boeing 787 Dreamliner [9], Boeing C130, Gulf-Stream G280, etc.

The importance of temporal correction makes the veri-cation and validation process an essential activity in the avionics development life cycle. In the early design phase,

|  |  |  |  |
| --- | --- | --- | --- |
| VOLUME 8, 2020 | This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/ | 38333 |  |
|  |  |

K. Zhang *et al.*: Independent Temporal Integration of ARINC653 Conformed Architecture Search Based Solution

**FIGURE 1.** An example of a two-layer scheduler.



conduction of a thorough analysis of integrated applications under the design model to evaluate whether or not the design can meet the temporal requirements under IMA architec-ture has been described in many works [13] [15]. Fur-thermore, during the implementation, worst-case execution time (WCET) analysis at the code level produced an esti-mation of the execution time in order to assure the satisfac-tion of temporal requirements under the worst-case scenario. The execution order model of a speci c CPU was modeled to obtain a static estimation of the execution time of each instruction [16], [17]. Meanwhile, the latency and delay of the IMA network characterized by a bandwidth allocation gap (BAG) were also studied in many works [14], [18], [19]. To ensure that the process execution inter- or inner-IMA partitions can meet their deadlines with given WCET and worst-case communication time (WCCT), another impor-tant issue is the schedule algorithm design of the IMA sys-tem [20], [21]: a typical two-layer schedule in the IMA system has also been modeled to verify the real-time feature of the system [22], [23], as shown in Fig. [1:](#page2)

At the top of Fig. [1](#page2) is the module-level scheduler schedule of the partitions on the processor for a xed cyclic length de ned by a MAjor Frame (MAF). Each MAF is a periodic schedule where slots are assigned to the partitions based on their period and WCET. Partition slots within MAF are cyclically scheduled nonpreemptively, in which the duration and period are de ned by the task ’period and WCET within the partition. Furthermore, as shown in the lower part of Fig. [1,](#page2) the tasks scheduled within the partition are preemptive, and need to be scheduled within the temporal slot allocated to the partition.

In contrast with the conventional federated architecture, IMA presents the airplane manufacturer with a number of challenges not previously encountered during integration. For example, as implemented on the Boeing 787 Dreamliner, where it is referred to as the Common Core System (CCS), the platform will initially host approximately 70 applications for 20-25 independent suppliers but is capable of hosting over 100 applications [9]. These real-time applications must meet their own temporal requirements while integrating within the context of sharing hardware resources (e.g., processors and communication networks) and software platform (e.g., inter-face). The system integrator needs to develop a so-called

38334

blueprint integration plan to lay out the path to bring each hosted application together and make sure that all resources are properly allocated in order to meet application and func-tionality requirements and a temporal requirement. Thus, temporally related characteristics of the prospective systems need to be evaluated on the basis of the characteristics of the platform in order to evaluate the complete integration plan.

Many works have been conducted for this issue, and mod-eling and simulation of the IMA architecture is conducted to evaluate the performance of system [24], [25] at the design level. Considering the detailed design of the architecture, the dynamic schedule algorithm in single and multiprocessors was studied at the partition level [23], and a MILP algorithm was also proposed for formalization of the two-layer sched-uler problem and to attempt to produce a possible schedule for the redundancy consideration, in which the exact MILP for-mulation fails in converging for fairly large problem instances within an acceptable time limit [20]; after that, an algorithm inspired by Game Theory, called the best-response algorithm, was designed for performance improvement [26]. Moreover, an in-depth search algorithm was used to search for the possible con guration of task mapping [15].

Although the model and algorithm previously mentioned both offer numerous contributions to the solution to the task allocation problem from the view of scheduling, little attention has been paid to task mapping with the optimiza-tion requirement. With continuous development in avionics, the increasing embedded functions present the integration and con guration with numerous dif culties. Moreover, the evaluation criteria during the integration can be antag-onist (such as CPM utilization, communication delay and safety level assurance). In that case, a multiobjective opti-mization must be introduced for the trade-off. Furthermore, the use of multicore platform and Field-Programmable Gate Array (FPGA) in avionics enables greater exibility of the IMA system [27], [28] Thus, how to obtain an optimized con-guration with respect to a given optimization target within an acceptable timeframe is a problem that must be answered with the full consideration of scalability of both the number of applications and the computing resources.

In this article, the heuristic algorithm was designed to over-come the problems. This algorithm takes both the scheduling constraint and its potential con icts into account. A classic objective in both research and industry was used as the opti-mization target. Several comparative studies were conducted on the industry data for the validation of the effectiveness and ef ciency of the proposed algorithms under the context of scalability of both the problem domain and computing resources.

The primary contribution of this paper is the presentation of a heuristic solution for task allocation under the optimiza-tion requirement during the integration phase. The solution can be more scalable with respect to both task numbers and partition numbers with harmonic or nonharmonic peri-ods and computing resources compared with the previously proposed conventional compositional optimization solution.

VOLUME 8, 2020

K. Zhang *et al.*: Independent Temporal Integration of ARINC653 Conformed Architecture Search Based Solution

Moreover, we conduct a comparative study of the two algo-rithms with respect to effectiveness and ef ciency for industrial cases.

The rest of this paper is organized as follows. Section 2 will present the related contributions, terminologies and research work performed within the domain. The problem de nition is elucidated in Section 3. However, the research method-ology used in the research study is elaborated in Section 4. Section 5 explains the experimental results and our interpre-tation, and a discussion on the research analysis will also be argued. In the end, Section 6 summarizes our conclusions and highlights future directions.

**II. RELATED WORK**

1. **TEMPORAL ALLOCATION IN IMA INTEGRATION**

Integrated modular avionics (IMA) architectures have pro-moted a new issue regarding avionics integration [10]. Numerous works have been proposed with respect to different facets of IMA system integration, including incremental inte-gration and certi cation [29] [32]. The sharing of hardware platforms makes the architecture integration into a virtual process, which shifts the temporal veri cation task from the supplier to the integrator [33]. Modeling is a promising method to settle these problems by simulating and validating the virtual integration plan [34], [35]: meanwhile, tools such as the AADL-based toolsets are designed for the veri cation of the design [29], and the utilization of timed automata for IMA temporal requirement veri cation was designed and proven to be helpful. Furthermore, CPM utilization and communication robustness performance criteria are used to quantify the quality of a set of valid temporal allocations [36].

1. **SCHEDULE AND RESOURCE ALLOCATION**

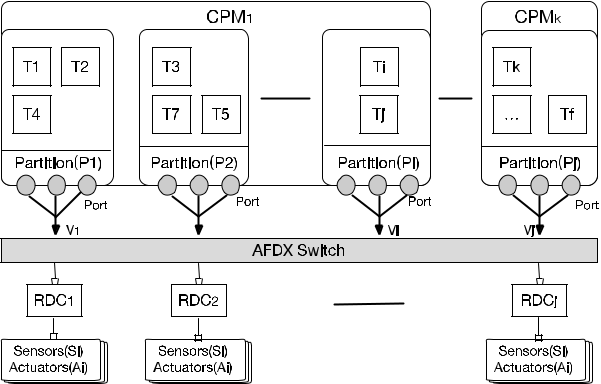
The precondition of temporal allocation is the schedulabil-ity of each task and the whole platform: Al Sheikh et al. formulated this mapping and scheduling problem as a mixed-integer linear programming challenge [37]. Further-more, they addressed this problem by proposing a Game Theory-based algorithm that not only nds a feasible sched-ule of tasks but also maximizes the relative distances between them [26]. The authors then extended it to sup-port harmonic and near-harmonic periods in the context of IMA [38]. A pseudo-polynomial-time lower-bound algorithm which functions by translating the task assignment problem into the so-called k-cut problem of a graph was proposed in [18] to address the partition allocation problem. Moreover, an in-depth search algorithm is introduced in [15] to allocate the tasks with communication and WCET into partitions.

1. **SEARCH-BASED SOFTWARE DESIGN OPTIMIZATION**

Harman and Jones [39] proposed search-based software engi-neering in 2001, reformulated classic software engineering problems as search problems and provided the key ingredi-ents for successful reformulation and evaluation criteria for search-based software engineering. IMA integration can be

VOLUME 8, 2020

**FIGURE 2.** IMA example architecture.



considered as the possible combination of ARINC653 con-formed applications in the whole platform, which has been proved to be an NP-hard problem [40]. The search-based algorithm has also been proven to represent a promising method to settle these kinds of problems. Numerous search algorithms have been proposed, such as classic Linear pro-gramming (LP) search, the metaheuristic search algorithm genetic algorithm (GA) [41], simulated annealing (SA) [42], and the Hill Climbing algorithm. The extension of the classic search algorithm and their applications in architecture design optimization with consideration of different quality attributes have been extensively studied, as presented in the systematic literature review in [43].

**III. PROBLEM DEFINITION**

**A. FORMAL DEFINITION OF IMA ARCHITECTURE**

From the static view of architecture, typical distributed inte-grated modular avionics can be illustrated in Fig. [2.](#page3) The architecture can be decomposed into three parts: Real-time operating system (RTOS), APEX (Application Execution Interface) and Remote devices.

From the top of Fig. [2,](#page3) RTOS will be deployed into CPM to manage the computing resources and to support the temporal and spatial separation within the CPM. The avionics appli-cation represents that the concrete function of the aircraft will be deployed into the RTOS partition (illustrated in Fig. [2](#page3) as *Ti* is deployed in *Pi*). The communication channel within the applications is through the virtual links (*Vi*) and port de ned in different partitions, which can be data or command links between Task *Ti* and end devices through Avionics Full Duplex Switched Ethernet (AFDX). APEX standardized by ARNIC653 de nes the interface for both RTOS and avionics applications, which facilitates the distributed development of both avionics applications and RTOS. The CPM will connect and control the remote external devices (e.g., sensors (*Si*) and actuators (*Ai*)) through the remote device control (RDC) component.

A preliminary formal de nition of these components rep-resented in Fig. [2](#page3) can be described as a series of host tasks *Task* D *T*1; : : : ; *Ti* which will be allocated into a set of partitions *P* D *P*1; : : : ; *Pj*; after that, the partitions

38335

Considering the task schedule in the IMA, we must de ne the relation between task and partition and the relation between partition and CPM. The length of the smallest sched-ule unit in a speci c IMA system is de ned in Equation (5):

**C. SCHEDULABLE CONSTRAINT**

**A typical two-layer schedule model in IMA architecture has been extensively investigated by many researchers [20], [21]. One important validation for the correct allocation of the task into partitions is that the temporal requirement has been satis ed. As de ned before, the partitioned host in the CPM is scheduled in a xed and nonpreemptive cyclic MAF, which is pre-de ned in the blueprint. The length of MAF is de ned as Equation** [**(3)**](#page4)**. However, in industry, to ensure the maxi-mum utilization of the computing resources and simplify the scheduling algorithm, the periods of partitions in the same CPM are generally harmonic, so that Equation** [**(3)**](#page4) **can be rewritten as followsV**

**Meanwhile, the temporal attribute of *CPM* can be described by the MAjor time Frame (MAF), which will cyclically execute the partition scheduled in a static sequence during MAF, where the length of MAF is de ned by the least common multiple of the partition host in the CPM, as shown in Equation (3):**

***MAFlength* D *LCM*(*TjP*);**

***i*D1**

***TjP* D *GCD*(*TiF* );**

***must be allocated and scheduled on a set of parallel CPMs CPM* D *CPM*1; : : : ; *CPMk* .**

***B. TEMPORAL CONSTRAINT IN ARINC653***

***As described in the architecture de nition, the temporal def-initions of tasks, partitions, and CPMs correlate with each other. Formally, from the temporal facet, the taski2Task can be described by its worst-case execution time, period and deadline and can then be de ned with (CiT ; TiT ; DTi ). The worst-case execution time represents the maximum temporal requirement that the task needs to nish its required function, the period attribute is the cycle interval of the task to be reactivated, and the deadline is a relative offset from the start of the task, after which the task must be terminated. Similarly, the temporal de nition of the partition Pj2P can be described by the partition execution duration CjP and the period TjP, which can be represented in (CjP; TjP). However, the attributes of a partition (CjP; TjP) are de ned by the tasks (CiT ; TiT ; DTi ) hosted in the partition. The relation between them can be illustrated by Equation (1): CjP is the sum of the WCET tasks ‘while TjP is the greatest common divisor (GCD) of the task ’periodV***

***N***

***X***

K. Zhang *et al.*: Independent Temporal Integration of ARINC653 Conformed Architecture Search Based Solution

*CiP* D *CiF*

*where Ti*2*Pj*

*where Pj*2*CPMk*

*MAFlength* D *MAXPj*2*CPMk* (*TjP*)

*intervallength* D *MINPj*2*CPMk* (*TjP*)

38336

Moreover, as de ned in schedule theory, the necessary but not suf cient condition is that the utilization of the assigned time should be less than the computing time [22] when the system designer must allocate a set of tasks into partitions, to ensure the schedulability of the allocated tasks in the partition, and their relation can be described in Equation [(7):](#page4)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | *CiF* | < 1 | (6) |  |
|  |  |

*Fi*2*interval intervallength*

To ensure that all tasks can be schedulable in the partitions, the task assigned in this interval should satisfy the following constraintV

X*n CP*

*i* < 1 (7)

*i*D1 *TiP*

**D. OPTIMIZATION OBJECT**

The con guration of assigned tasks into partitions can vary with respect to different design considerations. Thus, the ef - cient use of the CPM computation power can be an important concern during system design. In this section, the evalua-tion criteria for the task allocation con guration which is frequently used in both industry and research areas [36] are

1. de ned as follows.

In general, the allocation needs to achieve a given utiliza-

tion of computing resources, which can be maximized under

(2)

the constraint of a certain safe margin. In this paper, the CPU utilization of the CPM is our primary concern: we consider the CPM utilization from a global view, which can be de ned as CPM utilization *Qavg*. For each CPM, the utilization factor *Qi* is de ned as the percentage of time the CPM is executingwithin a partition, formallyV

P*Pj*2*CPMk CjP*

(3) *Qi* D (8)

*MAFk*

where *CPMk* is the allocated computing unit partition, and *MAFk* is the duration of *CPMk* ; thus, *Qavg* can be de ned as

followsV

P

*Qavg* D *i*2[1:::*m*] *Qi* (9)

*m*

where *m* is the number of CPMs in the system, and *Qi* and *Qavg* may represent indexes of whether the allocation strategycan ensure suf cient use of CPM.

In practice, thousands of communications and hundreds of CPMs are integrated into an avionics IMA platform. Thus, managing and optimizing the temporal allocation is a very large and complex problem. In one facet, it must satisfy the constraint of the schedulable requirement from the real-time

1. requirement, and in other facet, it must nd a suf ciently effective con guration for better CPM utilization under an acceptable limit time. In solving this combinatorial opti-mization problem, conventional methods such as branch and bound or the MILP algorithm have been proven inef cient at large scale, or even for fairly complex problems, with respect
2. to CPU time or memory space. Thus, whether this complex

VOLUME 8, 2020

K. Zhang *et al.*: Independent Temporal Integration of ARINC653 Conformed Architecture Search Based Solution

problem can be solved by the heuristic algorithms, and how effectively it may be solved, are the questions that must be answered. In that case, the research questions can be de ned as follows:

**E. RESEARCH QUESTIONS**

**RQ1 (Sanity Check): How do GA and SA perform as compare with random search?** In any attempt at anSBSE formulation of a problem, this is a standard ‘baseline ’question asked. If a proposed formulation does not allow an intelligent computational search technique to convincingly outperform a random search, then there is clearly something wrong with the formulation. This question is thus adopted in SBSE research as a preliminary ‘sanity check ’ [44].

**RQ2 (Effectiveness Comparison): How does GA per-form with respect to effectivity compared with SA?** Thesealgorithms may outperform random search and may present different feature in the performance of solution search. Effec-tiveness is an index of whether the solution can produce a workable con guration with optimized utilization, which is also an important index to judge the validation of the solution. In this research study, effectiveness is an important question that must be answered.

**RQ3 (Ef ciency Comparison): How does GA perform with respect to ef ciency compared with SA?** Ef ciencyis another important index for solving the task allocation problem, which will show how rapidly the algorithm can obtain a solution. For that, there must be a balance between the effectivity and ef ciency in order to generate a more practicable solution for a certain scenario. In this research study, ef ciency is also another important question that must be answered.

**RQ4 (Multicore Acceleration Ratio Comparison): How does GA perform under the multicore scenario compared**

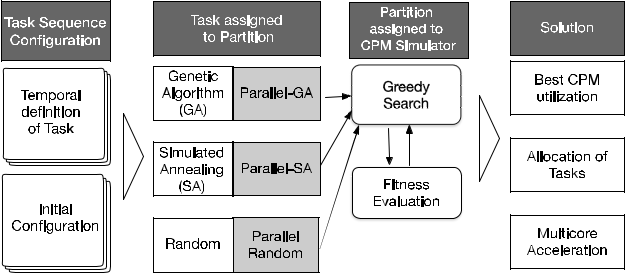
**with SA?** The acceleration ratio is an important indicator ofthe paralleling ability of both algorithms, which can be used to balance the effectivity and ef ciency. Moreover, under the scenario of limited time consumption allowed for obtaining the solution (e.g., online recon guration), a solution with a higher acceleration ratio can be easier to apply with multicore platform support.

**IV. METHODOLOGY**

The objective of the current research study is to nd a possible temporal allocation con guration and achieve a de ned opti-mization objective, maximized CPM utilization, for a given task sequence with the constraint of the schedulable require-ment. The methodology used in this research study answers the research question by formulating the temporal allocation task into a search-based format, and then searching for the solution regarding the given target. A heuristic algorithm was chosen to solve the temporal allocation, while the random search was also conducted as a control group to compare the effectiveness and ef ciency of the two algorithms. The par-alleling heuristic algorithm under multicore context was con-ducted for the scalability of the proposed problem, while the

VOLUME 8, 2020

**FIGURE 3.** Main Procedure of Methodology.



acceleration ratio of ef ciency and effectiveness were also examined and compared with random search. Speci cally, the genetic algorithm (GA) and simulated annealing (SA) algorithm are classic algorithms used to nd the solution to the allocation scenario which were selected to determine the temporal allocation in the context of IMA integration. More-over, an experiment in the industrial context was used to prove the effectiveness and ef ciency of the chosen algorithm, with each con guration, including 30 epochs of almost 2 hours each, conducted to eliminate the random error. Three typical con gurations with the same population and generation as in GA, SA, and Random algorithms were used as the same input of three algorithms to compare the ef ciency of the different problems. The main procedure of the methodology is illustrated in Fig. [3:](#page5)

As shown, the procedure can be divided into three main parts allocation of the task into partitions, allocation parti-tion into CPM, and evaluation of the tness to obtain the nal allocation con guration. On the left side of the gure, the rst part is the input of the model, which is the tem-poral attribution of tasks that need to be integrated into the IMA platform with an initial con guration of the application. The second part is the search solution part, which searches for the proper allocation con guration to allocate the previously generated tasks into partitions under the schedulable con-straint. The allocation of tasks into partitions has been proven to be an NP-hard problem, for which it is dif cult to nd the best solution in polynomial time, so that the heuristic algo-rithm is introduced to settle this problem. The seed solution of the algorithm can be generated by random search. However, in industry, the required time consumption to obtain an allo-cation con guration is also an important factor that must be taken into consideration (e.g., online recon guration). In this research study, a multicore version of the time allocation algorithm was designed to determine whether the algorithm can be accelerated with parallel computing so that it can produce an available solution in a considerably short time. The parallelization of the algorithm is applied to GA and SA so that they can execute paralleling on the multicore platform to reduce the calculated time. The time consumption will be compared with the single-process algorithm to obtain an acceleration ratio. The third part is the simulator, which allo-cates the previously generated partitions into CPMs. Since different allocations with the same number of CPMs leads to

38337

where

38338

8*x*(*S*1(*x*) ! 8*q*8*r*(*S*2(*q*) ^ *S*2(*r*) ! :(*S*3(*q*; *x*) ^ *S*3(*r*; *x*))))

8*c*;

For all CPMs, let *Taskc* D f*Ti*j*M*(*i*; 3) D *c*g, representing all tasks which belong to *CPMc*, where 1 *c j*. The schedulable constraint Equation [(12)](#page6) can then be written as followsV

The constraints can be rewritten as follows. The numbers of partitions and CPMS are limited, so that

*i* <D *MaximumPartitionLimit j* <D *MaximumCPMLimit*

*Max*f*Qavg*g

*The second row will be provided by the search algorithm (TAGA and TASA), while the third row, fourth row and the value of the matrix can be obtained by the simulator. The optimized target can be written as followsV*

K. Zhang *et al.*: Independent Temporal Integration of ARINC653 Conformed Architecture Search Based Solution

the same best *Qavg* for the same tasks, and each partition has two independent attributes, offsets to start and CPMs to which it belongs, the greedy algorithm is applied to acquire the best *Qavg*. In the nal part, the ultimate optimization object andthe best *Qavg* will be achieved, which will be used for the effectiveness and ef ciency comparison of GA and SA.

1. **FORM OF THE SOLUTION AND OPTIMIZED TARGET**

Search-based temporal allocation requires the formulation of mapping tasks into the form of search tasks, with correspond-ing constraints and optimization target. Speci cally, the solu-tion of the problem involves determining the mapping relation from tasks to partitions and partitions to CPMs, which contain the temporal con gurations of partitions into which a task is allocated, along with its offset in the allocated CPM. In this research study, a 4 *N* matrix *M* is de ned as the form of the solution, where *N* is the number of tasks. The rst row of the matrix *M* represents *Task*1 to *TaskN* , while the second row contains the partition to which the corresponding task is allocated, which can be de ned as *P*1 *Pi*. The third row is the CPM number to which the partition belongs, which is

de ned as *CPM*1 *CPMj*. The fourth row is the partition’s offset in that CPM. The value of matrix *M* is *val*(*M*) D *Qavg*, which is the optimization targetV

The schedulable constraint Equation [(14)](#page6) can then be writ-ten as followsV

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Ti*2X |  | *CP* | |  |  |  |
| *c* | | *iP* | < 1; f8*c*j; 1 *c j*g | (13) |  |
|  |  | *T* |  |
| *Task* |  |  | *i* |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  | (14) |  |

The optimization target in Equation [(16)](#page8) can be written as followsV

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Qavg* DP | *P*)=(*T* *P*) | |  |  |
| (*CiM* | *i* | (15) |  |

As illustrated, fewer CPMs leads to a higher *Qavg* under the same input, while the same amount of CPMs will have the same *Qavg* under the same input partition list, no matter how the partitions are scheduled.

1. **TASK ALLOCATION WITH GENETIC ALGORITHM (TAGA)**

The genetic algorithm is applied to allocate the tasks into partitions. In this research study, the task of TAGA is mapping tasks into partitions, which corresponds to the second row of

the solution matrix *M*. The search task optimized by GA is !

formulated as follows: Letting *X* (*i*) be the population of the *ith* iteration, where S, C and M are operators, the algorithmcan be written as followsV

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 2 | *T*1 | *T*2 |  | *Tn* |  | 3 |  |
|  |  | 6 |  | 7 |  |
| *M* | D | *Pi*1 | *Pi*2 |  | *Pik* |  |  |
|  | *CPMj*1 | *CPMj*2 |  | *CPMjk* | |  |
|  |  | *offset*1 | *offset*2 | *offset* | *n* |  |
|  |  | 6 |  |  |  |  | 7 |  |
|  |  | 4 |  |  |  |  | 5 |  |

1 *c j*

|  |  |
| --- | --- |
| *S*1 | (*x*) V *x* 2 *intervalc* |
| *S*2 | (*x*) V *Tx* 2 *Taskc* |
| *S*3 | (*x*; *y*) V *TxP*j(*y M*(*x*; 4)) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ! *X* (*i* | C | 1) | D | *S* | 2 |  | *M* |  | *C* |  | *S* | !*X* (*i*); *i* | D | 0; 1 |  | *n* |  |
|  |  |  |  |  |  | 1 |  |  |  |

(10)

where

*S*1: Select individuals from population

*C*: Crossover from selected individuals

*M*: Mutation

*S*2: Select best individuals

As is shown in Fig. [4,](#page7) transition, crossover, mutation, and tness evaluation are four main processes of TAGA used to produce an optimized offspring for each individual. The

1. evolution of the offspring is generated sequentially, which means that the algorithm will evaluate each individual and optimize that individual towards the optimization target by four processes in each epoch. After each epoch, the algorithm will select the best individual for this iteration and continue until the iteration is nished.

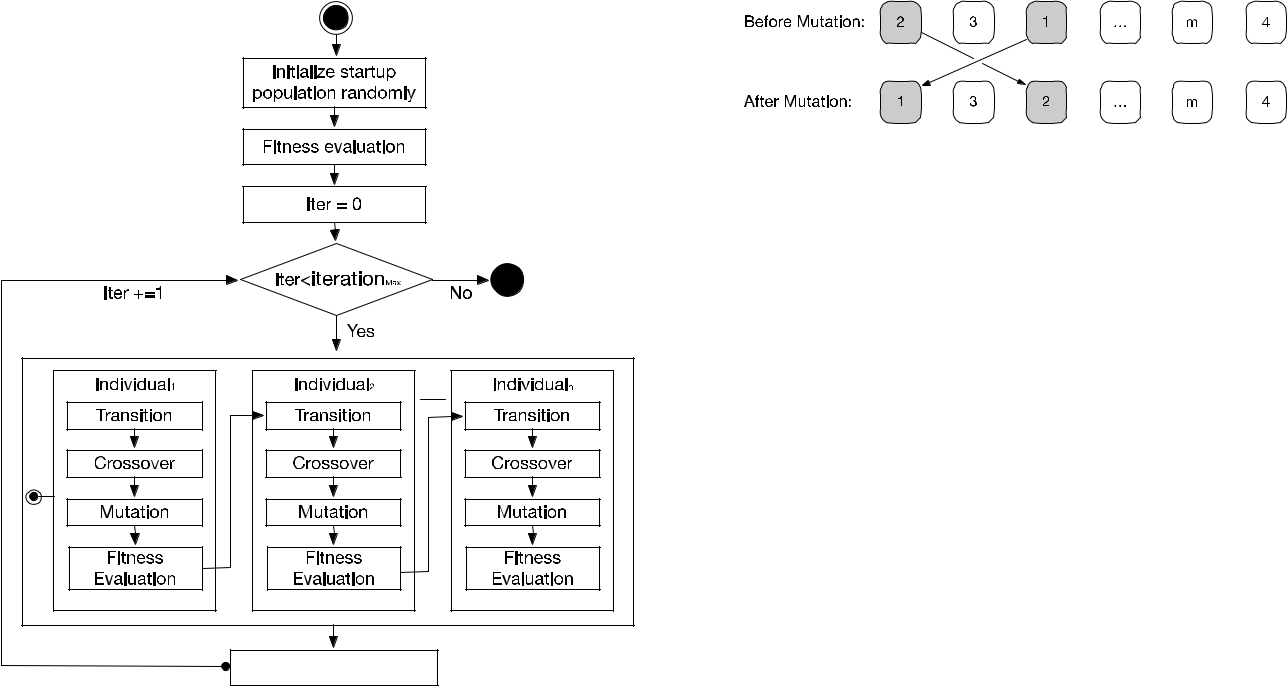
Speci cally, the solution was rst encoded in the following format, as shown in Fig. [4.](#page7) A set of *n* tasks is represented as a string of length *n*, while the partition is labeled in numbers ranging from 1 to *m*. The task here is de ned as the gene of the chromosome, and each gene of the chromosome will be assigned to a partition number ranging from 1 to *m*, representing the partition to which the task is allocated.

A single-point crossover is then performed for TAGA species. The crossover point is randomly generated within a range from 0 to *n*, which de nes the chromosome for exchange, and the offspring takes a chromosome from both parents, separated by the crossover point, as illustrated in Fig. [6.](#page7) The rst parent is chosen from the population in sequence, and the other parent is determined by Roulette

1. Wheel Selection. The crossover procedure exchanges the task

VOLUME 8, 2020

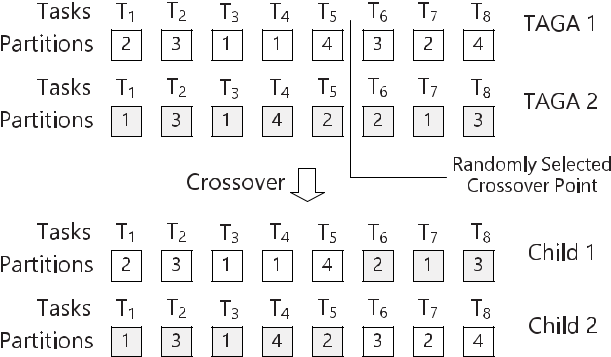
K. Zhang *et al.*: Independent Temporal Integration of ARINC653 Conformed Architecture Search Based Solution



**FIGURE 4.** The flowchart of TAGA.



**FIGURE 5.** Encoding tasks into partitions.

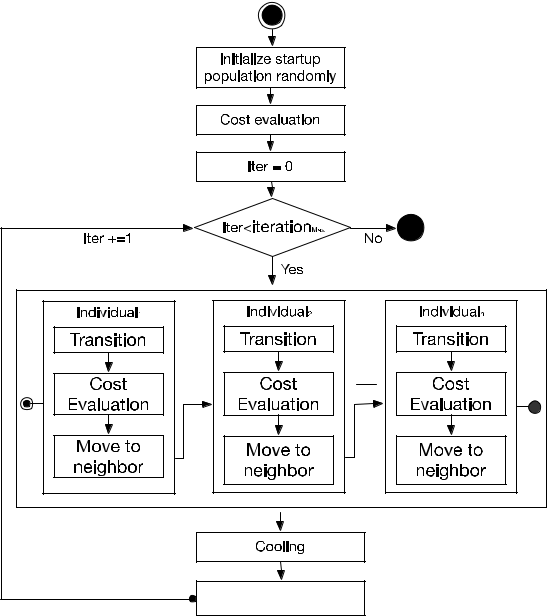


**FIGURE 6.** Crossover for TAGA: single point crossover.

allocations of two different individuals. The child1 individual contains the allocation of rst half tasks from Parent1, and the rest from Parent2, while the child2 individual contains the allocation of rst half tasks form child2, and the rest from child1. The entire procedure aims to choose the most excellent genes to create new individuals who might have higher tness value.

After this, the mutation operation will generate new indi-viduals to avoid falling into local optima. The task is then set to swap the allocations of two tasks chosen randomly from the individual, as shown in Fig. [7.](#page7) The tness evaluation of this

**FIGURE 7.** Mutation of TAGA solution.



**FIGURE 8.** Flowchart of TASA.

species is de ned as the *Qavg* of current partition allocation, for which a higher global CPM utilization leads to a higher tness value.

1. **TASK ALLOCATION WITH SIMULATED ANNEALING (TASA)**

The simulated annealing algorithm is a classic method to solve allocation optimization, which simulates the annealing process to obtain the best optimization result. As a compari-son algorithm, the task of TASA includes mapping between tasks and partitions, which is also the second row of the

solution matrix *M*. The solutions are optimized by simulated

! *th*

annealing as follows. Letting *X* (*i*) be the population of the *i* iteration, where S, C and M are operators, the algorithm can be written asV

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ! *X* (*i* | C | 1) | D | *T* |  | *S*!*X* (*i*); *i* | D | 0; 1 |  | *n* |  |
|  |  |  |  |  |  |  |

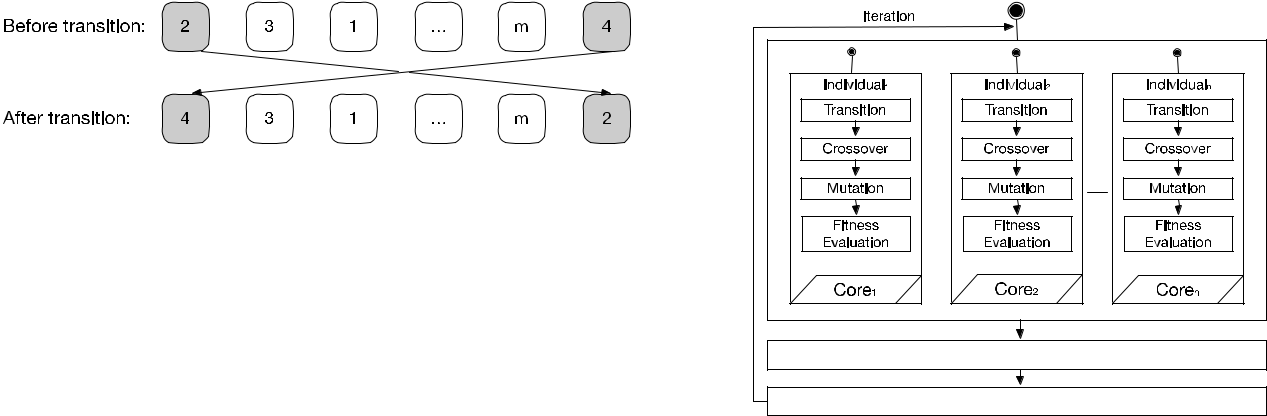
where

1. Select each individual from population
2. Transition of selected individual

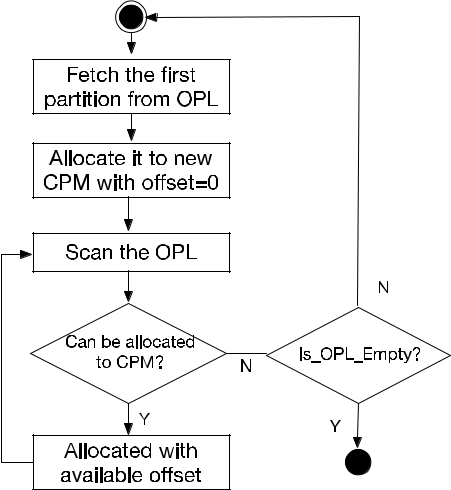
As is illustrated in Fig. [8:](#page7) transition, cost evaluation and moving to the neighbor are the three main processes to pro-duce an optimized individual. The neighbor point is chosen in order to attempt different individuals and evaluated by the tness function. Similarly, similar to GA, these three processes consist of the evaluation of the single individual,

|  |  |
| --- | --- |
| VOLUME 8, 2020 | 38339 |

K. Zhang *et al.*: Independent Temporal Integration of ARINC653 Conformed Architecture Search Based Solution



**FIGURE 9.** Transition of TASA solution.



**FIGURE 10.** Brief flowchart of the simulator.

where the sequential processing of all individuals constitutes an epoch. After each epoch, the algorithm will record the best individual and continue until the iteration is nished.

Speci cally, the solution is encoded in the following for-mat, as shown in Fig. [10.](#page8) The allocation of a set of *n* tasks to a set of *m* partitions is represented as a string of length *n*. Each position in the string corresponds to a partition number, ranging from 0 to *m*, representing the partition to which the task is allocated.

A transition is generated by changing the allocation of two randomly chosen tasks, as is shown in Fig. [10.](#page8) The transition to neighbor procedure exchanges some task allocations of each individual in attempting to nd a better solution by searching within its neighbors. The cost of each individual

*Ei* in the population is considered as the distance betweenthe ideal goal (*Qavg* D 1:0) and the current *Qavg* calculated by the simulator, which can be calculated by the formula *Ei* D1 *Qavg*. The acceptance rate of the neighborhood canbe measured by Equation [(16):](#page8)

(

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *P* D | 1 | *temp* | | 0<1*E* | (16) |  |
| *e* | 0>1*E* |  |
|  |  |  | 1*E* |  |  |  |

which means that the algorithm always accepts better solu-tions but may also accept an inferior solution with the probability *P*. That probability is approximately half at the beginning, but increasingly approaches 0 as iterations proceed. Cooling is performed via the geometric method *Ti*C1D *Ti*, where *Ti* is the current temperature, *Ti*C1rep-resents the next temperature and stands for the temperature decreasing gradient.

38340

**FIGURE 11.** A brief flowchart of paralleling TAGA.

**D. SIMULATOR**

The simulator is designed to allocate partitions into CPMs with a greedy algorithm and is integrated as the tness evaluator of GA and SA. In addition to the tness value, the simulator will provide the CPM to which each partition is allocated, along with its offset, which completes the third and fourth rows of the solution matrix. The owchart of the simulator can be seen in Fig. [10:](#page8)

As shown, the simulator takes an ordered partition list (OPL) as its input, in which the tasks with long computing time but short period will be allocated rst. The simulator will rst fetch the rst partition from the ordered partition list and allocate it to a new CPM. It will then sequentially scan the ordered input list to nd an available partition that can be added to that CPM. This step will be repeated until no more partitions can be added to that CPM. After this, one epoch will nish, and used partitions will be removed from the partition list; the procedure above will be repeated to ll a second partition until the input list is empty. The simulator then evaluates *Qavg* by averaging the utilization rate of each core. By simulating the input, the simulator can determine the allocation of partitions into *M* CPMs that uses the least CPMs.

To serve as a tness evaluation, the output of the simulator is deterministic, which means that it will provide the same output if the input for the greedy algorithm scans is the same and will process the input list in the same order.

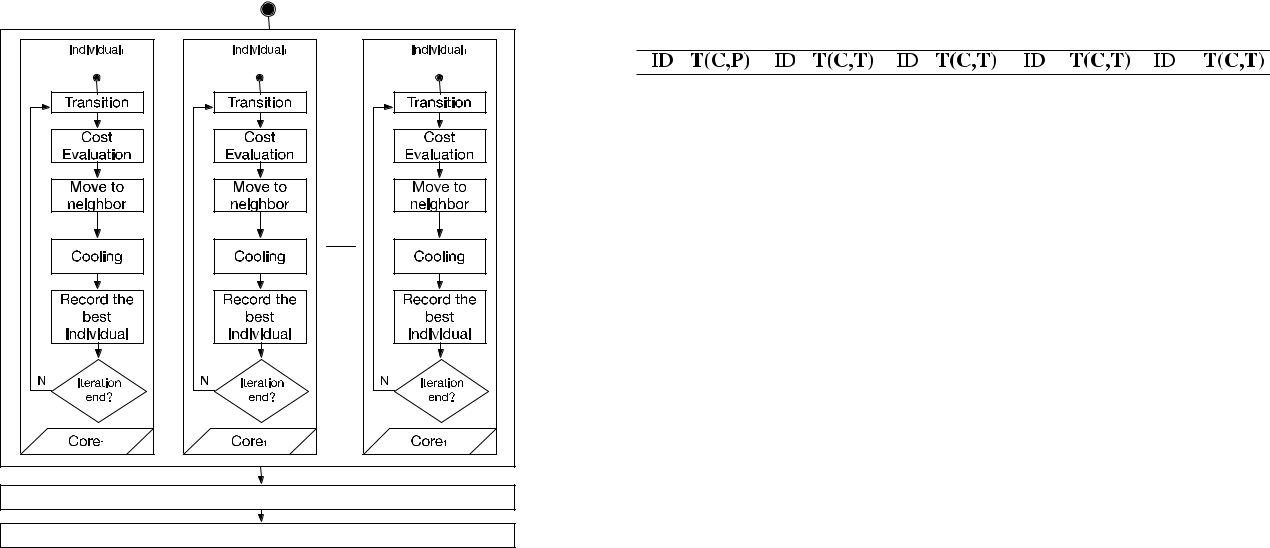
**E. PARALLELING GA AND SA**

The paralleling of the algorithm needs to decompose the algo-rithm into the paralleling part to achieve speed increase under the multicore platform. Speci cally, in both algorithms, the evaluation of each individual is the most time-consuming part of the entire process, which needs to be decomposed into independent units. The decomposition of the GA and SA procedures can be seen in Fig. [11](#page8) and Fig. [12:](#page9) As shown, the crossover and mutation procedure in TAGA can be designed as a parallelization, in which the individuals can crossover and mutate on different cores at the same time. The procedure for selection of the best individuals must

VOLUME 8, 2020

K. Zhang *et al.*: Independent Temporal Integration of ARINC653 Conformed Architecture Search Based Solution

**TABLE 1.** Task list.



**FIGURE 12.** A brief flowchart of paralleling TASA.

evaluate the tness value of each individual, and so all indi-vidual evaluations must be nished before entering the next iteration. However, the parallelization design of TASA is different from that of TAGA, which can be observed in the following gure.

As shown, TASA can parallel all of its procedures, includ-ing the selection of the current best individual. This indi-vidual can be separately self-optimized without waiting for other individuals. Each individual will achieve the best tness value, which will then be compared in order to select the best individual after the entire process is complete.

**V. EXPERIMENT AND RESULT ANALYSIS**

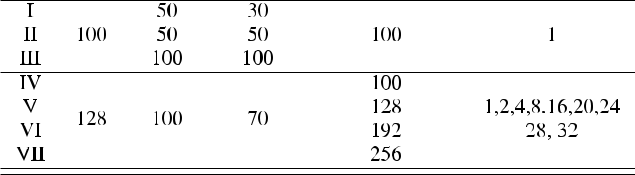
In this section, the synthetic data and con gurations are introduced, and the experimental design with result analysis is also presented. Finally, all four research questions will be answered. The purpose of the experiment is to verify the validity of the proposed method in chapter IV. Furthermore, the purpose includes answering the research questions raised in chapter III. A server with 32-core CPUs (4 \* E7-4830), 128 GB DDR3 memory, 500 GB HDD, and a Windows Server 2008 R2 system was used to execute the experiments. Approximately 300 hours were required to nish all of the experiments.

1. **SYNTHETIC DATA AND EXPERIMENTAL CONFIGURATIONS**

A collection of tasks that need to be allocated into parti-tions and CPMs was used for the experiment. The temporal attributes of the tasks were distinct from those of our pre-vious case study in [25]; however, more simulated tasks in the same pattern were generated, as shown in Table [1.](#page9) For the most common scenario, in this research study, only the periodic independent task is considered, which is periodically executed in a given period; the relation between tasks is also independent. Thus, each task includes two attributes,

VOLUME 8, 2020

**TABLE 2.** Configurations of the experiment.



the execution time and the period of executing, and the unit of time is normally measured in milliseconds. The period value is generated randomly from the set f20; 30; 40; 60; 100; 120g, while the execution time is a random integer between 1 and 0:2 *period*.

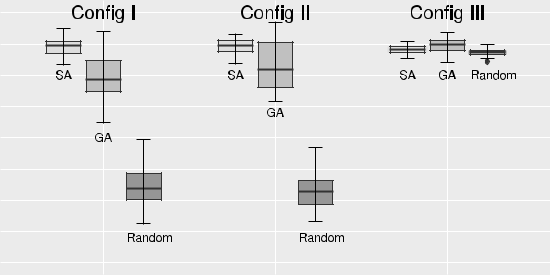
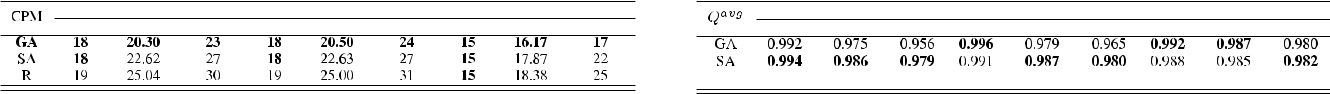
The comparison of the effectiveness and ef ciency of the algorithms are tested with respect to different con gurations. As shown in Table [2,](#page9) the con guration is divided into two main groups: one is for the single-core platform, and the other for the multicore platform. In the single-core experiment, three con gurations were implemented, while four con g-urations were executed under the multicore platform. Each con guration is applied to three algorithms, GA, SA, and Random, in which the random search algorithm is considered as a control group.

To determine the acceleration ratio of parallelization, a group of another 36 experiments was carried out. Each experiment includes the same 128 tasks to be allocated, as shown in Table [2.](#page9) As shown, each experiment has a maximum partition number of 100 and a maximum CPM number of 70. However, each experiment owns different pop-ulation numbers and simulated cores. All con gurations will

38341

K. Zhang *et al.*: Independent Temporal Integration of ARINC653 Conformed Architecture Search Based Solution

**TABLE 3.** Statistics of the CPM used for task allocation. **TABLE 4.** Statistics of utilization rate.



be executed under both Paralleling GA and Paralleling SA. The acceleration ratio here is simply de ned according to the following equationV

|  |  |
| --- | --- |
| *Timemulticore* |  |
| *Acc*\_*Ratio* D *Timesinglecore* | (17) |

As illustrated in Equation [(17),](#page10) the time consumption for producing an optimized result under the single-core condition is taken as the benchmark, while the ratio is the comparison between time consumption under the multicore condition and the benchmark. Experiments under 1, 2, 4, 8, 16, 20, 24, 28 and 32 cores on the 32-core processor were carried out to determine the trend of acceleration ratio. Moreover, the population numbers of 100, 128, 196 and 256 were used to measure whether the population number in uences the acceleration ratio.

Con gurations of the experiment I, II, III are repeated 30 times while con gurations IV, V, VI, VII are repeated 5 times to reduce the statistical error.

**B. ANALYSIS RESULTS**

The objective of this research study is to provide a heuristic solution for task allocation with an optimization objective under the IMA platform. In this section, the results of the comparison experiment are presented, which proved that the heuristic solution is much better than random search with respect to either the single-core or the multicore platform, while the performance of the chosen heuristic algorithm exhibits different features with respect to effectiveness and ef ciency. Moreover, at the end of this section, we have encountered all of the research questions that have been presented in the problem de nition section.

1) THE SANITY CHECK

The sanity check requires answering whether the problem is suitable for the heuristic algorithm, and furthermore, by what percentage the algorithm outperformed random search. After the execution of con gurations I, II, III for 30 times each, the static results can be observed in Table [3](#page10) and Table [4.](#page10)

As illustrated in Table [3,](#page10) the least, average and highest numbers of CPM are used to represent the allocation results. The results show that both GA and SA can provide a better solution than random search in all three con gurations (less CPM is used), even in the con guration III, where the least number of CPM is equal. However, GA exhibits the best result with respect to CPM utilization for all three levels, as shown by the numbers labeled in bold. Finally, the utilization of the CPM can be calculated from Equation (**??**)

The utilization rate is shown in Table [4,](#page10) where three static indexes (Best, Average, Worst) are used to represent the

38342

**FIGURE 13.** Utilization rate comparison.



performance of each con guration and the highest rate in each static index is labeled in bold. The results show that both SA and GA can provide a better solution than a random search in all three con gurations. Moreover, SA achieved the six highest utilization rates in three con gurations, while GA produced the two highest utilization rates in the best static column. The results can be observed more directly in Fig. [13:](#page10) As is shown, in each con guration, SA and GA perform much better than random search. This is especially true in con gu-ration I and con guration II, which do not contain much space for the allocation. Even in con guration III, where the best solutions are much more easily found, the static indexes from SA and GA are still superior to random search.

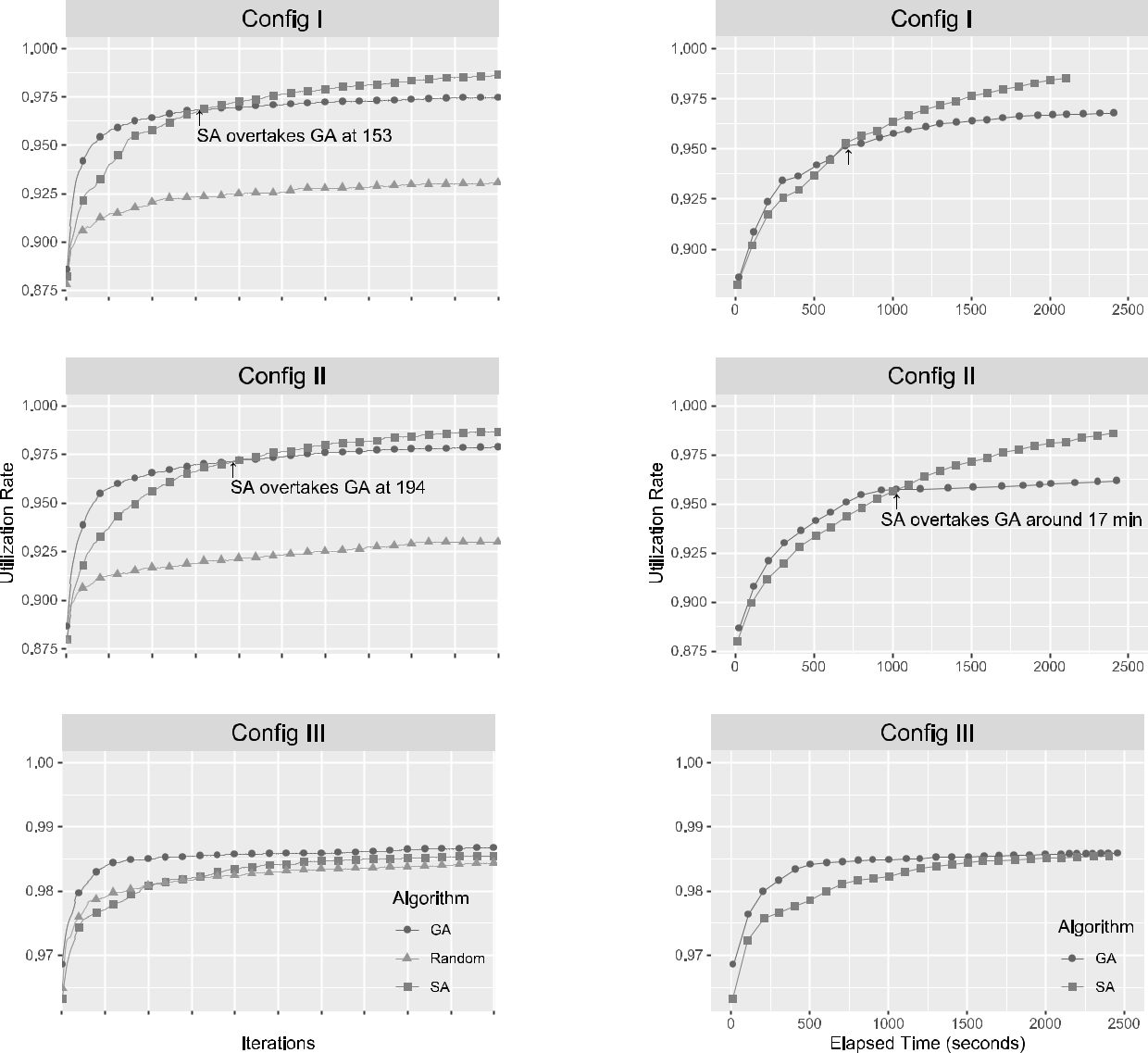
2) EFFECTIVENESS AND EFFICIENCY COMPARISON

During the experiment, the best utilization rate in each iter-ation and its time consumption were recorded for further analysis. The utilization rates of the solutions produced by three algorithms during the optimization process with con g-urations I, II, III are shown in Fig. [14.](#page11) The average value of the best utilization rate found for the same iteration during 30 repeated experiments is used to eliminate the random error. It can be observed that the utilization rates of workable task allocation found by GA and SA rise rapidly during the rst 50 iterations and keep ascending in the following iterations. However, the utilization of random search grows much more slowly and stops improving after approximately 200 iterations. Speci cally, GA provides a better solution in the rst 153 iterations in con guration I and the rst 194 iterations in con guration II, after which the utilization of SA overtakes GA. However, GA is always better than SA in con guration III.

In addition to the effectiveness comparison, the ef ciency will determine the time to produce an optimized result. The results obtained during the experiment can be observed in Fig. [15.](#page11) As illustrated, GA is more ef cient than SA under a given lower optimization target (e.g., 0.950 in con guration I and con guration II). After that point, SA can continuously

VOLUME 8, 2020

K. Zhang *et al.*: Independent Temporal Integration of ARINC653 Conformed Architecture Search Based Solution



**FIGURE 14.** Effectiveness comparison.

optimize the result, while GA increases slowly, which means that SA will be more ef cient for a given higher target.

However, GA is always better in con guration III and can produce an optimized result in a shorter timeframe. There-fore, GA is more suitable when the calculation time is limited, or when the target utilization rate is not especially high,

|  |  |  |  |
| --- | --- | --- | --- |
| especially in the con gurationI and con guration II. | | |  |
|  | To compare the solutions provided by GA and | | SA, |
| a | Wilcoxon signed-rank test [45] | is applied. This | test |
| is | an alternative nonparametric test | that is widely | used |
| in validation. | |  |  |
|  | The results of the test are presented in Table [5.](#page11) The p-values | | |

reported in Table [5](#page11) conform with the box-plots, indicating that SA is superior to GA, while GA is better than Random with respect to con gurations I and II. GA outperforms SA with respect to con guration III. In other words, SA performs better than GA with respect to con gurations I and II, while GA is superior with respect to con guration III.

**FIGURE 15.** Efficiency comparison.

**TABLE 5.** The results of the Wilcoxon signed-rank test.



3) MULTICORE ACCELERATION RATIO COMPARISON

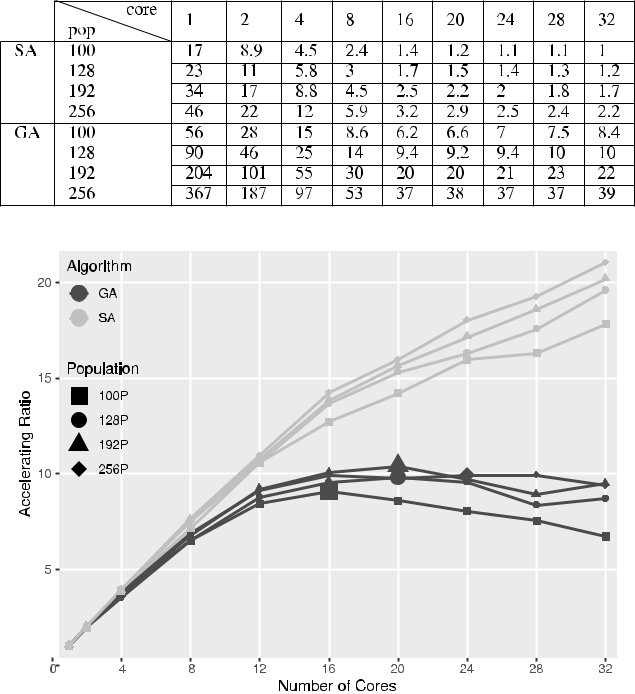
As mentioned in section IV, paralleling versions of TAGA and TASA are designed to reduce the time consumption involved in searching the optimization result. The average running times of SA and GA are illustrated in Table [6](#page12) for the numbers of cores ranging from 1 to 32.

As illustrated, the average acceleration ratios for the two algorithms over different numbers of cores decreases obvi-ously. The lowest time consumption is for the SA version under the 100 population on 32 cores, and the rate of decrease of SA is much higher than that of GA.

|  |  |
| --- | --- |
| VOLUME 8, 2020 | 38343 |

K. Zhang *et al.*: Independent Temporal Integration of ARINC653 Conformed Architecture Search Based Solution

**TABLE 6.** Time consumption (minutes) of paralleling searching.



**FIGURE 16.** Comparison of acceleration ratios.

According to the acceleration ratio de ned in Equa-tion [(17),](#page10) the nal acceleration ratio can be observed in Fig. [16](#page12) As demonstrated by the light gray lines in Fig. [16,](#page12) the acceleration ratio of paralleling SA monotonically increases upon the addition of more computational resources. However, the black lines show that adding more computa-tional resources cannot bene t GA further when the number of cores is approximately 16 to 24. The peak acceleration ratio of GA occurs when the numbers of cores are 16, 20, 20 and 24, while the population numbers are set to 100, 128, 192 and 256, respectively.

More interestingly, it can be observed that the acceleration ratio of SA is much higher than that of GA, especially after GA reaches its peak. After inspecting the solutions generated during the process, the reason preventing GA from bene t-ing from parallelism might be attributed to the reproduction process of GA. The reproduction process of GA generates a large number of invalid solutions that are time-consuming to validate or repair. Moreover, the communication overhead for sorting and comparing individuals in this population becomes the key factor in running time when parallelism within a population increases to a certain level. Therefore, adding cores is effective when paralleling SA, but will be less useful when paralleling GA, especially when there are more than approximately 16 cores.

4) ANSWERS TO RESEARCH QUESTIONS

The objective of this study was accompanied by four research questions. After the formulation of the industry data into the heuristic algorithm and conduction of the

38344

designed experiments, we are now in a state of responding to the answers of research questions.

**RQ1 (Sanity Check): How do GA and SA perform compared with random search?** After formulating the tem-poral allocation problem into the heuristic search optimiza-tion, both effectiveness and ef ciency of the solution can be compared with random search. As shown in Fig. [11,](#page8) and as observed for the experimental data, both GA and SA can provide better solutions than random search. Moreover, both GA and SA perform better than random search, especially in situations where good solutions are more dif cult to nd, as in con gurations I and II.

**RQ2 (Effectiveness Comparison): How does GA per-form with respect to effectivity compared with SA?** Effec-

tiveness mainly focuses on the result of the optimization target, which is the average utilization of the CPM in our research study. Both algorithms perform well, but SA is more effective in the CPM limited situation, while GA performs much better when time is limited, as is shown in Fig. [12](#page9) and Table [3.](#page10) Moreover, with the Wilcoxon signed-rank test, the solutions found by SA are higher than those of GA.

**RQ3 (Ef ciency Comparison): How does GA perform with respect to ef ciency compared with SA?** Ef ciencypertains to the time required to obtain an optimized solution. In our experiment, GA performs more ef ciently when there is a strict time limit or a relatively minimally optimized target. However, SA is more ef cient in other situations, such as when there is no time limit or when the solution needs to be as good as possible. These results show that both algorithms have applicable areas, which may depend on the speci c scenario.

**RQ4 (Multicore Acceleration Ratio Comparison): How does GA perform under the multicore scenario compared**

**with SA?** As shown in the experiment, both SA and GAcan be accelerated to a relatively large ratio under the mul-ticore scenario. However, due to the algorithm structure of the TAGA, adding more computational resources can only bene t GA to a certain degree, after which the defects of GA overtake the bene ts gained from parallelism. Moreover, the acceleration ratio of SA is monotonically increasing in our experiments as more computational resources are added for the independent structure of the paralleling decomposition of individual evaluation.

**VI. CONCLUSION**

Ample work has been conducted on the modeling of temporal relations under ARINC653 [13], [36], [46] and attempting to nd possible temporal allocation con gurations for a given task and platform [15], [47]. Furthermore, the solution of the temporal allocation must consider the WCET of the task, health monitoring strategy, the dependency relation, communication delay, safety assurance level and even the requirements of the certi cation agency. Therefore, for the deterministic requirement of the avionics system design, little attention has been paid to the solving of the task alloca-tion under optimization target with a probabilistic algorithm.

VOLUME 8, 2020

K. Zhang *et al.*: Independent Temporal Integration of ARINC653 Conformed Architecture Search Based Solution

However, with the development of the ARINC653-based platform, increasing applications are decomposed into tasks, so that the allocation and the optimization will eventually become important tasks during the virtual integration of the IMA system, and even within the online recon guration mod-ule. In this scenario, a workable allocation con guration or the con guration with an optimized target, or even under con-siderably small delay, form the basis of the decision support or even the recon guration module.

In this study, the mapping of the independent task allo-cation under ARINC653 conformed architecture is solved by the heuristic algorithm. Two classic heuristic allocation algorithms, GA and SA, are chosen for the experiment, and the parallel version is also designed for the low delay requirement of possible avionic applications. By combining industry data and simulated data, experiments are carried out to discover the effectiveness and ef ciencies of the chosen heuristic algorithm.

The results show that both GA and SA are much better than random search: SA is more effective in the CPM limited situation, while GA performs much better when the time is limited. Moreover, GA performs more ef ciently when there is a strict time limit or a relatively low optimization target. In the scenario of acceleration by multicore, SA performed much better than GA, which may indicate a potential choice for further application.

The algorithms presented in this paper have answered the problem but also raise some new questions that need to be discussed in the future. There are two primary aspects that may represent the focus of future work. First, though the algorithms provide solutions within reasonable time, they are still too slow to meet the needs of online recon guration. Thus, parallel computing methods, such as GPU accelera-tion, multithreading or FPGA, can be applied to the current algorithm. However, parallel computing methods may lead to some potential safety issues such as exchanging information, which also must be taken into consideration. Second, the tasks in industry are more complex and include more attributes such as network latency, multicore processors, limited max-imum CPM usage and associated tasks, as mentioned previ-ously. This indicates that the algorithms must be modi ed by inclusion of additional elements and constraints in order to satisfy more critical requirements, which may represent the direction of further research.

**VII. ACKNOWLEDGMENT**

We dedicate this work to Prof. C. Liu, who unfortunately passed away just before the paper was submitted for publi-cation. Prof. Liu served in an essential role in the research described herein, and he is greatly missed by all of us.

**REFERENCES**

1. S. Asif and P. Webb, ‘‘Software system integration-middleware An overview,’’ *Int. J. Comput. Appl.*, vol. 121, no. 5, pp. 27 29, 2015.
2. S. Hamdan and S. Alramouni, ‘‘A quality framework for software contin-uous integration,’’ *Procedia Manuf.*, vol. 3, pp. 2019 2025, 2015.

VOLUME 8, 2020

1. M. Dias, V. Uquillas-Gomez, D. Cassou, and S. Ducasse, ‘‘Software integration questions: A quantitative survey,’’ Tech. Rep. 01093496, Lab-oratoire d’Informatique Fondamentale de Lille, Lille, France, 2014.
2. D. A. M. Montana, D. F. C. Rodriguez, D. I. C. Rey, and G. Ramos, ‘‘Hardware and software integration as a realist SCADA environment to test protective relaying control,’’ *IEEE Trans. Ind. Appl.*, vol. 54, no. 2,
   1. 1208 1217, Mar. 2018.
3. S. Mirheidari, A. Fallahi, D. Zhang, and K. Kuppam, ‘‘AUTOSAR model-based software component integration of supplier software,’’ *SAE Int. J.* *Commercial Vehicles*, vol. 8, no. 2, pp. 544 548, Sep. 2015.
4. H. Kopetz, *Real-Time Systems: Design Principles for Distributed Embed-ded Applications*. Springer, 2011.
5. GulfStream. *Gulfstream 280 avionics application*. Accessed: Jan. 26, 2018.

[Online]. Available: https://www.rockwellcollins.com

1. A. Tsourdos, ‘‘Flight control systems: Practical issues in design and implementation,’’ *Proc. Inst. Mech. Eng.*, vol. 214, no. 5, p. 323, 2000.
2. C. B. Watkins and R. Walter, ‘‘Transitioning from federated avionics architectures to integrated modular avionics,’’ in *Proc. IEEE/AIAA 26th* *Digit. Avionics Syst. Conf.*, Oct. 2007, p. 2.
3. R. Garside and F. Pighetti, ‘‘Integrating modular avionics: A new role emerges,’’ *IEEE Aerosp. Electron. Syst. Mag.*, vol. 24, no. 3, pp. 31 34, Mar. 2009.
4. C. Spitzer, U. Ferrell, and T. Ferrell, *Digital Avionics Handbook*. Boca Raton, FL, USA: CRC Press, 2014.
5. H. Butz, ‘‘Open integrated modular avionic (IMA): State of the art and future development road map at Airbus deutschland,’’ *Signal*, vol. 10,
   1. 1000, 2010.
6. N. Badache, K. Jaffres-Runser, J.-L. Scharbarg, and C. Fraboul, ‘‘End-to-end delay analysis in an integrated modular avionics architecture,’’ in *Proc. IEEE 18th Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2013,
   1. 1 4.
7. M. Lauer, J. Ermont, F. Boniol, and C. Pagetti, ‘‘Latency and freshness analysis on IMA systems,’’ in *Proc. ETFA*, Sep. 2011, pp. 1 8.
8. E. Deroche, J.-L. Scharbarg, and C. Fraboul, ‘‘Mapping real-time commu-nicating tasks on a distributed IMA architecture,’’ in *Proc. IEEE 21st Int.* *Conf. Emerging technol. Factory Autom. (ETFA)*, Sep. 2016, pp. 1 8.
9. J. Souyris, E. L. Pavec, G. Himbert, G. Borios, V. JØgu, and R. Heck-mann, *Computing the Worst Case Execution Time of an Avionics Program* *by Abstract Interpretation* (OpenAccess Series in Informatics (OASIcs)),vol. 1. Dagstuhl, Germany: Schloss Dagstuhl-Leibniz-Zentrum für Infor-matik, 2007.
10. P. Lokuciejewski and P. Marwedel, *Worst-Case Execution Time Aware* *Compilation Techniques for Real-Time Systems*. Berlin, Germany:Springer, 2010.
11. C. Ekelin and J. Jonsson, ‘‘A lower-bound algorithm for minimizing net-work communication in real-time systems,’’ in *Proc. Int. Conf. Parallel* *Process.*, Jun. 2003, pp. 343 351.
12. H. Charara, J.-L. Scharbarg, J. Ermont, and C. Fraboul, ‘‘Methods for bounding end-to-end delays on an AFDX network,’’ in *Proc. 18th Euromi-cro Conf. Real-Time Syst. (ECRTS)*, Aug. 2006, p. 10.
13. A. A. Sheikh, O. Brun, and P.-E. Hladik, ‘‘Partition scheduling on an ima platform with strict periodicity and communication delays,’’ in *Proc. 18th* *Int. Conf. real-time Netw. Syst.*, 2010, pp. 179 188.
14. L. Sha, T. Abdelzaher, K.-E. ¯rzØn, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. K. Mok, ‘‘Real time scheduling theory: A historical perspective,’’ *Real-Time Syst.*, vol. 28, nos. 2 3, pp. 101 155, Nov. 2004.
15. Y.-H. Lee, D. Kim, M. Younis, and J. Zhou, ‘‘Scheduling tool and algorithm for integrated modular avionics systems,’’ in *Proc. 19th DASC. 19th Digit.* *Avionics Syst. Conf.*, vol. 1, Nov. 2002, p. 1.
16. J.-E. Kim, M.-K. Yoon, S. Im, R. Bradford, and L. Sha, ‘‘Multi-IMA par-tition scheduling with synchronized solo-partitions for multi-core avionics systems,’’ Dept. Comput. Sci., IDEAR, Chicago, IL, USA, Tech. Rep. han-dle/2142/35280, 2012.
17. C. Fraboul and F. Martin, ‘‘Modeling advanced modular avionics archi-tectures for early real-time performance analysis,’’ in *Proc. 7th Euromicro* *Workshop Parallel Distrib. Process. (PDP)*, 1999, p. 181.
18. K. Zhang, J. Wu, C. Liu, S. S. Ali, and J. Ren, ‘‘Behavior modeling on ARINC653 to support the temporal veri cation of conformed application design,’’ *IEEE Access*, vol. 7, pp. 23852 23863, 2019.
19. A. Al Sheikh, O. Brun, P. E. Hladik, and B. J. Prabhu, ‘‘A best-response algorithm for multiprocessor periodic scheduling,’’ in *Proc. 23rd Euromi-cro Conf. Real-Time Syst.*, Jul. 2011, pp. 228 237.

38345

K. Zhang *et al.*: Independent Temporal Integration of ARINC653 Conformed Architecture Search Based Solution

1. T. Gaska, C. Watkin, and Y. Chen, ‘‘Integrated modular avionics-past, present, and future,’’ *IEEE Aerosp. Electron. Syst. Mag.*, vol. 30, no. 9,
   1. 12 23, Sep. 2015.
2. B. Sutter eld, J. A. Hoschette, and P. Anton, ‘‘Future integrated modular avionics for jet ghter mission computers,’’ in *Proc. IEEE/AIAA 27th* *Digit. Avionics Syst. Conf.*, Oct. 2008, p. 1
3. A. Khoroshilov, D. Albitskiy, I. Koverninskiy, M. Olshanskiy, A. Petrenko, and A. Ugnenko, ‘‘AADL-based toolset for IMA system design and integration,’’ *SAE Int. J. Aerosp.*, vol. 5, no. 2, pp. 294 299, Oct. 2012.
4. A. Wilson and T. Preyssler, ‘‘Incremental certi cation and integrated modular avionics,’’ *IEEE Aerosp. Electron. Syst. Mag.*, vol. 24, no. 11,
   1. 10 15, Nov. 2009.
5. G. Bartley and B. Lingberg, ‘‘Certi cation concerns of integrated modular avionics (IMA) systems,’’ in *Proc. IEEE/AIAA 27th Digit. Avionics Syst.* *Conf.*, Oct. 2008, p. 1.
6. R. L. Eveleens, ‘‘Integrated modular avionics development guidance and certi cation considerations,’’ *Mission Syst. Eng.*, vol. 2, pp. 1120 1132, 2006.
7. P. H. Feiler, J. Hansson, D. D. Niz, and L. Wrage, ‘‘System architecture vir-tual integration: An industrial case study,’’ Software Eng. Inst., Pittsburgh, PA, USA, Tech. Rep. CMU/SEI-2006-TN-011, 2009.
8. J. Delange, L. Pautet, A. Plantec, M. Kerboeuf, F. Singhoff, and F. Kordon, ‘‘Validate, simulate, and implement arinc653 systems using the aadl,’’ *ACM SIGAda Ada Lett.*, vol. 29, no. 3, pp. 31 44, 2009.
9. T. Robati, A. Gherbi, A. El Kouhen, and J. Mullins, ‘‘Design and sim-ulation of distributed IMa architectures using TTEthernet: A model-driven approach,’’ *J. Ambient Intell. Humanized Comput.*, vol. 8, no. 3,
   1. 345 355, Jun. 2017.
10. N. Badache, K. Jaffres-Runser, J.-L. Scharbarg, and C. Fraboul, ‘‘Man-aging temporal allocation in integrated modular avionics,’’ in *Proc. IEEE* *Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2014, pp. 1 8.
11. A. Al-Sheikh, O. Brun, and P.-E. Hladik, ‘‘Decision support for task mapping on IMA architecture,’’ in *Proc. 3rd Junior Researcher Workshop* *Real-Time Comput. (JRWRTC)*, 2009, pp. 31 34.
12. Q. Zhang, S. Wang, and B. Liu, ‘‘Approach for integrated modular avionics recon guration modelling and reliability analysis based on AADL,’’ *IET* *Softw.*, vol. 10, no. 1, pp. 18 25, Feb. 2016.
13. M. Harman and B. F. Jones, ‘‘Search-based software engineering,’’ *Inf.* *Softw. Tech.*, vol. 43, no. 14, pp. 833 839, 2001.
14. J. Korst, E. Aarts, and J. K. Lenstra, ‘‘Scheduling periodic tasks,’’ *INFORMS J. Comput.*, vol. 8, no. 4, pp. 428 435, 1996.
15. J. H. Holland, *Adaptation in Natural and Arti cial Systems: An Intro-ductory Analysis With Applications to Biology, Control, and Arti cial Intelligence*. Cambridge, MA, USA: MIT Press, 1992.
16. N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, ‘‘Equation of state calculations by fast computing machines,’’ *J. Chem. Phys.*, vol. 21, no. 6, pp. 1087 1092, Jun. 1953.
17. A. Aleti, B. Buhnova, L. Grunske, A. Koziolek, and I. Meedeniya, ‘‘Soft-ware architecture optimization methods: A systematic literature review,’’ *IEEE Trans. Softw. Eng.*, vol. 39, no. 5, pp. 658 683, May 2013.
18. M. Harman, S. A. Mansouri, and Y. Zhang, ‘‘Search-based software engineering: Trends, techniques and applications,’’ *ACM Computing Surv.* *(CSUR)*, vol. 45, no. 1, p. 11, 2012.
19. A. Shukla, H. M. Pandey, and D. Mehrotra, ‘‘Comparative review of selection techniques in genetic algorithm,’’ in *Proc. Int. Conf. Futur-istic Trends Comput. Anal. Knowl. Manage. (ABLAZE)*, Feb. 2015,
    1. 515 519.
20. 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,
    1. 359 386, Jul. 2012.
21. D. Tamas-Selicean and P. Pop, ‘‘Task mapping and partition allocation for mixed-criticality real-time systems,’’ in *Proc. IEEE 17th Paci c Rim Int.* *Symp. Dependable Comput.*, Dec. 2011, pp. 282 283.

38346

KUI ZHANG received the master’s degree in information management and information sys-tems from the Beijing Institute of Technol-ogy, Beijing, China, in 2010. He is currently pursuing the Ph.D. degree with the Software Engineering Institute (SEI), Beihang University. His research interests include model-driven engi-neering, model-based real-time analysis, airwor-thiness certi cation, model-based safety analysis, and general model-based software engineering.



JUNCHEN LIU is currently pursuing the bache-lor’s degree with the School of Computer Science and Engineering, Beihang University, Beijing, China. His research interests are in the areas of search-based software engineering, big data analy-sis, software engineering, and heuristic algorithms.



JIAN REN received the M.Sc. degree from the Queen Mary University of London, the M.Sc. degree from King’s College London, and the Ph.D. degree in computer science from Univer-sity College London. He is currently an Assistant Professor with the School of Computer Science, Beihang University, Beijing. His research interests include search-based software engineering, soft-ware project planning and management, require-ments engineering, and evolutionary computation.



JINGHUI HU received the bachelor’s degrees in computer science and technology, mathematics and applied mathematics and the master’s degree in computer technology from Beihang Univer-sity. He is currently an Assistant Engineer with the AVIC Manufacturing Technology Institute. His research interests include intelligent manu-facturing technology, search-based software engi-neering, and machine learning.



CHAO LIU received the M.S. and Ph.D. degrees in computer software and theory from Beihang University. He was a Professor with the School of Computer Science and Engineering, Bei-hang University. His research interests include software quality engineering, software testing, model-driven software development, and soft-ware process improvement. During the preceding decade, he primarily focused on the modeling and veri cation of safety-critical software and sys-



tems, including safety requirement modeling and analysis, evidence-based software safety analysis and evaluation, software safety and reliability anal-ysis based on the software development process, and model-driven soft-ware testing.



VOLUME 8, 2020