

面向多任务的综合模块化航空电子系统重构方法

李国栋^{1,2}, 张翰谷¹, 单鹏³, 王小辉^{1,4}, 周炳林¹, 赵岸荣¹

(1.西北工业大学, 陕西 西安 710072; 2.航空工业第一飞机设计研究院, 陕西 西安 710089;
3.西安航空计算技术研究所, 陕西 西安 710068; 4.中国运载火箭技术研究院, 北京 100076)

摘要:综合模块化航空电子系统(integrated modular avionics, IMA)因其灵活性、易更改、高容错等特点广泛应用于航空领域。然而, IMA 面临多变的环境和频繁变更的需求, 现有的人工配置重构蓝图和传统算法生成重构蓝图方法存在自动化程度低、质量难以保证等问题, 难以满足 IMA 任务切换时对资源调度的复杂度和难度不断提升的需求。针对 IMA 的多任务重构蓝图生成问题, 提出了基于多步学习和网络噪声的 DDQN-MS-NN 重构算法, 提高了资源调度的自动化水平和质量, 从而提升 IMA 的稳定性和抗风险性。

关键词: IMA; 蓝图重构; 强化学习

中图分类号: V243

文献标志码: A

文章编号: 1000-2758(2025)04-0821-10

综合模块化航空电子系统(integrated modular avionics, IMA)是在战场环境愈加复杂, 信息化作战要求不断提高的情况下提出的。IMA 的提出旨在解决电子系统设计和集成复杂性^[1], 满足不断变化的功能需求和应用场景。IMA 具有系统综合化、结构层次化、网络一体化、调度灵活化、维护中央化等优点^[2], 能提供高性能的计算和控制能力^[3], 以及实现设备之间的高效通信和互联。同时 IMA 允许其根据需求增加或替换系统的组件, 使得 IMA 能依据实际的应用场景切换任务模式以完成复杂多变的任务^[4-5]。

起初对 IMA 的重构一般通过各种方法来辅助建模, 使用模型规划重构资源分配。Wang 等^[6]使用 SPLE(software product line engineering)方法降低开发 IMA 等复杂系统的相关成本, 并在领域建模进行特征选择时考虑 IMA 资源配置约束以保证安全性要求, 使用多目标遗传算法实现一种 IMA 资源配置方法。Ma 等^[7]利用动态故障树分析法(dynamic fault tree analysis, DFTA)和 EDSPN 模型对 IMA 系统多分区软件的运行状态进行分解, 得出分区软件间故障传播率对 IMA 系统功能的影响, 为 IMA 系统

多分区软件的可靠性分析提供了良好的模型基础。Wei 等^[8]提出了一种基于安全的 IMA 系统架构及软件重新配置方法。该配置方法将错误事件和危险触发器集成到重配置过程中, 架构和软件重配置方法结合为一个扩展的 AADL 模型, 制定了 AADL 模型到确定性随机 Petri 网的映射规则, 并用 IMA 系统验证了基于安全的重配置方法的适用性和有效性。

使用算法解决重构问题的实例: Hollow 等^[9]将模拟退火算法引入重构领域, 该方法以适配度作为退火算法评估指标, 寻找等效的重构方案。虽然具有一定实践价值, 但是模拟退火算法不能解决耗时问题。Zhang 等^[10]将序贯博弈多智能体强化学习应用于综合模块化电子系统重构领域。该方法为了解决传统算法收敛难度高的问题, 使用基于有偏估计的策略梯度蒙特卡洛搜索树算法加速算法收敛, 但是未考虑智能体竞争、合作阶段探索因子自适应问题。

对 IMA 的研究从静态重构开始, 不少静态重构方法考虑通过设置一个能用以管理监控全局的表结构, 节省重构过程中的资源消耗。陈晓磊^[11]提出了一种静态重构技术, 为 IMA 设置一个可替换的调度表, 通过管理调度表改变分区调度频率和调度顺序, 为静态重构方法添加了新方向, 但是调度表本身又

会增加额外的资源开销。动态重构是一种有效的综合模块化航空电子系统故障容错方法,张涛等^[12]提出了一种基于序贯博弈多智能体强化学习的 IMA 系统重构方法,解决如何在复杂的多级关联故障模式下快速自动生成有效的重配置蓝图问题。杨威等^[13]则是预先建立故障情况和重构蓝图的映射关系,当故障发生时切换对应重构蓝图,该方法在 ARINC653 系统上表现良好且建立映射关系本身消耗资源小,但是该方法需要对所有可能的故障预先设置映射,这在系统复杂的情况下是很难实现的。罗庆等^[14]将 Q-learning 和模拟退火算法相结合,使用智能体模拟退火算法并动态调整探索因子。与单纯的模拟退火算法相比,该方法具有收敛速度更快、蓝图质量更高的优点。

分析关于 IMA 的研究情况,可以发现现有关于 IMA 的研究大多集中在静态重构技术、安全分析以及重构系统建模领域。重构技术研究多集中于故障重构领域,而多任务重构研究则较少。

本文对面向多任务的综合资源调度进行研究,IMA 作为典型的多任务系统,在 DDQN 基础上引入多步学习 (multi-step learning, MS) 和噪声网络 (noisy network, NN) 机制,针对 IMA 重构问题提出了 DDQN-MS-NN 算法以提高重构质量。

1 IMA 重构蓝图建模

IMA 以分区思想为核心思想,旨在通过模块化设计将不同部分和区域之间进行隔离^[15-16]。本节将 IMA 不同部分根据负责功能的不同建立对应模型,并在模型之上为多任务重构蓝图建立优化指标,用于衡量重构蓝图质量。

1.1 IMA 建模

IMA 提供了一种可靠且符合实时要求的软件开发框架,支持多个应用程序在同一硬件平台上并行运行,同时确保应用程序间的隔离性和安全性。

IMA 的核心思想是分区隔离概念。ARINC653 将系统划分为多个独立的分区,每个分区都有自己的应用程序和资源,同时定义了一套严格的时间和空间分配机制^[17-18],每个分区都被分配了特定的 CPU 时间片和内存空间,以确保应用程序在规定时间内执行并且互不干扰。ASAAC 则从整体架构上将系统划分为 3 层,分别是应用程序层、操作系统层、模块支持层。相比于 ARINC653,2 种结构构建

思路相同,只是 ASAAC 将系统进一步细分,更符合设计思想中的系统性和整体性。本文以上述 2 种系统为基础,同样以分区思想为核心,将系统划分为 CPU、分区、应用 3 层,逐层进行 IMA 建模。

1.1.1 硬件资源模型

IMA 中,通过通信总线进行数据传输和信息控制。通信总线是连接系统中组件和设备的物理媒介,系统中的各个中央处理器 (CPU) 和分区通过总线连结在一起,其具体的结构图如图 1 所示。

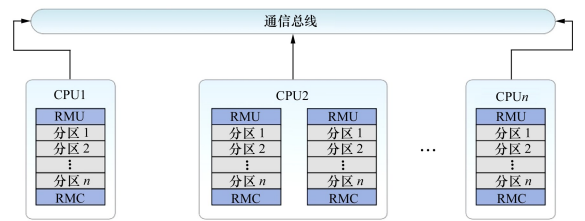


图 1 IMA 硬件结构

硬件资源对系统的性能、可靠性和功能扮演着重要的角色。本文结合 IMA 的分区隔离概念,将 CPU 从时间、空间、健康状态方面抽象出特征进行建模。实际建模中,将 CPU 抽象为有单个或多个分区的数据处理和控制在模块,该模块内部属性包括处理器时间片、系统内存大小、系统内部分区情况以及是否存在需要迁移应用。为全面地描述该模块,得到如(1)式所示的四元组。

$$C = \langle T_{cpu}, M_{cpu}, V_{list}, S \rangle \quad (1)$$

式中: C 表示 CPU 处理器; T_{cpu} 表示处理器从系统处分配得到的时间片; M_{cpu} 表示处理器从系统处分配得到的内存空间; V_{list} 表示处理器上内部分区列表; S 表示处理器内是否存在需要迁移应用,取值 0 代表否,1 代表是。

除上述四元组,为保证每个分区的资源不过量分配,配置的分区还有限制:即 CPU 上所有分区时间片总和不能超过 CPU 主框架时间,所有分区分配到的内存总和不能超过 CPU 内存大小。结合限制得到(2)式。

$$\begin{cases} V_i \in \{V_1, V_2, \dots, V_n\}, 1 \leq i \leq n \\ T(V_s) = \sum_{i=1}^n T(V_i), 0 < T(V_s) \leq T_{cpu} \\ M(V_s) = \sum_{i=1}^n M(V_i), 0 < M(V_s) \leq M_{cpu} \end{cases} \quad (2)$$

式中, V_i 表示编号第 i 个分区; $T(V_s)$ 表示处理器所有分区的时间片总和; $M(V_s)$ 表示处理器所有分区

的内存空间总和。

1.1.2 分区资源模型

IMA 为保证同时支持多个应用程序在同一硬件平台上并行运行,将每个处理器内部划分为多个分区,其中每个分区内部都是一个独立的执行环境,有自己的相关资源;分区与分区之间彼此隔离,分区之间平行运行互不干扰。分区主要保证了系统在时间和空间上的隔离,因此本文建模也从时间和空间两方面进行构建。

1) 分区时间资源

IMA 内部的时间资源分配,以 CPU 为基础单位。CPU 层面与分区层面采用的时间资源分配策略存在差别。

CPU 层面: CPU 层面的时间调度,为提高资源利用率,不能让某个分区独占时间资源,而各个分区之间又不存在优先级关系, CPU 将时间划分为多个时间片,划分时间片不能出现重叠的情况,分区按顺序获取时间片执行任务。所有分区执行任务时间不能超出 CPU 主框架时间,只能在规定时间片内运行。

分区层面: 在每个被 CPU 调度到的时间片内,分区自身再对其内部的多个任务进行调度,通常采用动态的优先级调度策略。分区内部任务共享该分区被分配到的时间资源,因此需要在限定时间内完成任务执行,以实现分区内部任务的实时响应与高效协同。

2) 分区空间资源

IMA 使用虚拟内存对分区空间资源进行管理。分配资源时,每个分区设置独立的虚拟内存,映射到实际物理内存时,保证分区内部的虚拟内存存在物理内存上是连续的,分区之间的映射关系则不要求连续,保证彼此之间相互隔离。

根据 IMA 分区时间调度和内存分配特征,本文针对分区的时间和空间关键属性构建分区资源模型,如(3)式所示。

$$V = \langle t_{\text{start}}, t_{\text{duration}}, V_{\text{memstart}}, V_{\text{memsize}}, S_{\text{app}} \rangle \quad (3)$$

式中: V 表示分区模型; t_{start} 表示分区的开始执行时间; t_{duration} 表示分区的持续执行时间; V_{memstart} 表示分区虚拟内存存在处理器中实际起始物理地址; V_{memsize} 表示处理器划分给分区的虚拟内存大小; S_{app} 表示挂载在分区内部的应用软件集合。

1.1.3 应用软件模型

IMA 内部部署的软件由管理软件和应用软件

组成^[19]。管理软件不易更改且不能根据实际需求变更,属于偏向底层的内容,因此本文构建软件模型时仅考虑应用软件。构建应用软件模型时从时间和空间两方面考虑。

在空间方面,应用软件部署于分区之中。应用软件运行需要足够的内存空间来保存运行时数据,如堆栈数据和代码段等。应用程序保证性能和稳定性的前提是有足够的内存空间。

在时间方面,因为航空航天领域广泛使用 IMA,对应用软件的实时性有着较高要求,应用软件需要在各种时间限制下完成任务。常见的应用软件时间限制包括执行周期、一个周期内的最坏情况执行时间(worst case execute time, WCET)、应用软件的截止运行时间(deadline)。

考虑到应用软件的基本要素,如应用名称、应用状态,并结合上述空间、时间约束,共同构建出应用软件模型。

$$A_{\text{pp}} = \langle p_{\text{id}}, p_{\text{cycle}}, l_{\text{priority}}, T_{\text{WCET}}, T_{\text{deadline}}, S_{\text{mem}}, S_{\text{state}} \rangle \quad (4)$$

式中: A_{pp} 表示应用软件模型; p_{id} 表示应用软件的进程 ID; p_{cycle} 表示应用软件的执行周期; l_{priority} 表示应用软件的任务优先级,分为 1 ~ 5 级,取值 1 代表优先级最低,5 代表优先级最高。 T_{WCET} 表示进程最坏情况执行时间; T_{deadline} 表示应用软件截止运行时间; S_{mem} 表示应用软件所需的内存资源; S_{state} 表示应用软件的迁移状态,取值 0 代表不需要迁移,1 代表迁移。

1.1.4 任务模式模型

任务模式是指 IMA 应对不同任务时,系统内部依据不同任务的需求挂载相对应的应用软件。不同应用软件负责各自的功能和逻辑,使系统能够适应不同的应用场景。但 IMA 内部资源有限,无法在一个任务模式中加载所有可能的应用场景,只能挂载该任务所必须的应用软件。因此本文定义的任务模式是完成该任务需要的应用软件资源,如(5)式所示。

$$T_{\text{B}} = \langle A_{\text{list}} \rangle \quad (5)$$

式中: T_{B} 表示任务模式; A_{list} 表示任务模式中的应用软件列表。

1.1.5 配置蓝图模型

为提高资源利用率,IMA 使用重构配置蓝图(reconfigure blue prints)调度资源,IMA 在遭遇外部任务变更或内部故障时,依据重构配置蓝图对系统

内部资源和应用进行调整,以应对故障或完成新的任务。重构配置蓝图分为故障重构蓝图和任务重构蓝图,分别在系统遭遇故障和任务变更时使用。本文讨论的重构蓝图属于任务重构蓝图。

分析重构配置蓝图的构成,发现蓝图配置也符合分区思想,因此蓝图模型也从处理器到应用软件分层构建,如(6)~(10)式所示。

$$R_{BP} = \langle C_{list}, V_{list}, A_{list}, M_{ap} \rangle \quad (6)$$

$$C_{list} = \{C_1, C_2, \dots, C_n\} \quad (7)$$

$$V_{list} = \{V_1, V_2, \dots, V_m\} \quad (8)$$

$$A_{list} = \{A_1, A_2, \dots, A_t\} \quad (9)$$

$$M_{ap} = \{A_i \xrightarrow{C_i} V_j\}, i \in [1, |A_{list}|], j \in [1, |V_{list}|] \quad (10)$$

式中: C_{list} 为系统中处理器的集合; n 表示系统中处理器一共有 n 个; V_{list} 为系统中分区的集合; m 表示系统中分区一共有 m 个,每个分区有自己独立的物理内存资源和系统时间片; A_{list} 为系统中应用软件的集合; t 表示系统中应用软件一共有 t 个; M_{ap} 用于表示应用软件在系统中的挂载位置,例如: $M_{ap} = \{V_2 \xrightarrow{C_1} V_3\}$ 表示 V_2 挂载在 V_3 ,而 V_3 又在处理器 C_1 中。

系统初始 IMA 蓝图配置为: C_1 的分区 1 挂载 $A_1 \sim A_2$,分区 2 挂载 A_3 ; C_2 的分区 1 挂载 $A_4 \sim A_5$,分区 2 挂载 $A_6 \sim A_8$,分区 3 未挂载应用; C_3 的分区 1 挂载 A_9 ,分区 2 未挂载应用; C_4 的分区 1 挂载 A_{10} ,分区 2 未挂载应用。

1.1.6 重构迁移模型

IMA 在遇到内部故障或者外部任务变更时触发重构,依据触发条件的不同分为故障重构和任务重构。触发重构时系统的配置会依据重构蓝图进行迁移。重构迁移时需要考虑多方面因素,具体约束条件在 1.2 节中给出。

对重构迁移蓝图的模型构建要能体现出整个系统的资源分配,如(11)式所示。

$$R_{BPmove} = \{R_{BPcur} \xrightarrow{activelist} R_{BPdes}\} \quad (11)$$

式中: R_{BPmove} 表示重构迁移蓝图模型; R_{BPcur} 表示系统内部资源、分区、应用软件当前时刻状态; R_{BPdes} 表示系统内部资源、分区、应用软件的目標状态; $activelist$ 表示该次重构系统应用软件迁移、移除、挂载行为总和。

在任务变更后,系统的蓝图进行了重构。重构迁移蓝图的内容涉及整个系统的处理器、分区资源、

所有应用软件。重构蓝图与初始蓝图的应用进行比较发现, A_6 被移除,新增了 A_{11} ,并将 A_{11} 迁移至资源充足的 C_3 的分区 2 中。

1.2 系统重构约束及优化指标

IMA 在生成重构蓝图时,需要考虑多方面的约束,本节对蓝图所需考虑约束进行讨论。IMA 的核心思想是分区隔离思想,在这一思想的主导下,对 IMA 各部分进行模块化设计。各个模块之间相互隔离又共享同一硬件资源,在考虑约束时可以从某一模块去考虑,也可以从多个模块之间共用的硬件资源限制去考虑。本节通过分析 IMA 的硬件资源条件在实际重构蓝图中的影响,从实时性约束、空间约束、唯一性约束定义资源约束。

1.2.1 唯一性约束

IMA 使用虚拟化技术和分区隔离将系统的各部分隔离开来,提高了系统的可靠性和容错性。系统的各部分彼此独立且有各自需要的各类资源。系统中各分区中的应用软件只能部署在一个分区内,无法跨分区部署,这会造成不可控的影响。根据配置蓝图模型中的应用软件列表 A_{list} 和分区列表 V_{list} ,提出(12)式建立部署关系矩阵(deployment relationship matrix, DRM)。

$$D_{RMij} = \begin{cases} 1, & A_i \text{ 部署在分区 } V_j \\ 0, & A_i \text{ 未部署在分区 } V_j \end{cases} \quad (12)$$

$$1 \leq i \leq |A_{list}|, 1 \leq j \leq |V_{list}|$$

DRM 矩阵的行代表系统内应用软件的总数,列代表系统内分区的总数。因此,唯一性约束的定义(13)式所示。

$$\sum_{j=1}^{|V_{list}|} D_{RMij} = 1, i \in [1, |A_{list}|] \quad (13)$$

式中, D_{RMij} 表示应用软件 A_i 在分区 V_j 上的部署情况。对矩阵 D_{RM} 的每一列求和,结果等于 1 或 0,代表 A_i 在单一分区内部署或尚未部署,不会出现某一应用在多个分区部署的情况,由此可以保证部署关系的唯一性。

1.2.2 实时性约束

IMA 对实时性要求严格,所有任务都需要在规定时间内完成,其为每个 CPU 分配了主框架时间, CPU 内部的各种应用执行均需要在主框架时间内完成,结合应用软件模型,实时性约束有如下限制:

1) 分区内应用软件的周期最坏执行时间(WCET)应在分区的时间窗口之内,即所有的任务周期执行时间必须比分区时间片小。

2) 在处理器分配的时间片结束之前,分区内所有应用软件应当结束,即分区内应用软件执行的完成时间必须在分区时间片之内。

实时性约束如(14)式所示。

$$\begin{cases} \sum_{i=1}^n W_{CETA_i} \leq V_{durtime}, A_i \in V \\ V_{starttime} \leq A_{timedeadline} \leq V_{starttime} + V_{durtime} \end{cases} \quad (14)$$

式中: W_{CETA_i} 表示应用软件 A_i 的最坏情况执行时间; $V_{durtime}$ 表示分区 V 的时间窗口; $V_{starttime}$ 表示分区 V 的开始执行时间; $A_{timedeadline}$ 表示软件的运行截止时间。

1.2.3 空间约束

IMA 同一处理器中的各分区采用虚拟内存分配内存空间。虚拟内存并不能凭空增加空间资源,因此各分区内应用软件所需内存总和不能超过处理器分配给分区的内存空间大小。由此,存在空间约束如(15)式所示。

$$\sum_{i=1}^n A_{memsize_i} \leq V_{memsize}, A_i \in V \quad (15)$$

式中: $A_{memsize_i}$ 表示部署在分区内的应用软件 A_i 所需空间资源大小; $V_{memsize}$ 表示分区 V 的内存空间总大小。

1.2.4 负载均衡

IMA 是多核处理器系统,在多核处理器系统中,通过负载均衡实现系统资源的均衡利用。当系统中存在负载不均衡时,某些处理资源可能被过度利用,而其他资源可能处于闲置状态,导致系统整体性能下降,无法充分利用所有资源,任务完成所需时间延长,影响系统正常运作。因此,确保系统各个处理器负载均衡是至关重要的。影响负载的因素有很多,其中 CPU 使用率和内存使用率影响最大也最易获取。因此主要从这两方面入手。

IMA 内部资源分配以分区作为基本单位,因此从分区的 CPU 使用率和内存使用率开始,将同一 CPU 下的各个分区负载情况转换成向量形式,通过分析向量的离散情况评估 CPU 的负载均衡状态。分区负载向量(partition load, PL)如(16)式所示。

$$\begin{cases} L_{V_i} = \mu_1 C_{use} + \mu_2 M_{use} \\ \mu_1 + \mu_2 = 1 \end{cases} \quad (16)$$

式中, C_{use} 表示分区 V_i 的 CPU 利用率; M_{use} 表示分区 V_i 的内存利用率; μ_1 表示 CPU 利用率权重系数; μ_2 表示内存利用率权重系数。

针对分区 V_i , CPU 使用率和内存使用率的计算如(17)式所示。

$$\begin{cases} C_{use} = \frac{\sum_{i=1}^{|A_{list}|} W_{CETA_i}}{V_{durtime}} \\ M_{use} = \frac{\sum_{i=1}^{|A_{list}|} M_{A_i}}{M_{V_p}} \end{cases} \quad (17)$$

分区负载向量可以表示各个分区的负载情况,但无法表示负载均衡程度。负载均衡可以通过离散程度来衡量。而标准差可以用于衡量数据的离散程度,各分区负载均衡时标准差较小,当标准差较大时表明各分区负载出现不均衡的现象,需要进行调整。因为负载均衡无法直观查看,本文通过标准差得出分区内向量离散情况,等效估计负载均衡状态。

$$S_{LB} = 1 - \sqrt{\frac{\sum_{i=1}^{|V_{list}|} (L_{V_i} - \bar{L})^2}{|V_{list}|}} \quad (18)$$

式中: S_{LB} 表示系统负载均衡指标; $|V_{list}|$ 表示系统中分区总数; L_{V_i} 表示分区的负载向量; \bar{L} 表示理想情况,此时系统负载完全平衡。

1.2.5 重构时间

IMA 触发任务切换时,除了保证重构蓝图的可用性,还需要考虑执行任务切换过程中,系统内部资源配置的时间消耗。IMA 对实时性有着严格的标准,任务必须在规定时间内完成。IMA 内部的应用软件迁移分为 3 步:第一步保存应用软件的上下文信息,对应用软件中的数据进行打包;第二步通过数据总线将打包后数据传输到目的分区;第三步解包,并根据上下文消息恢复现场。重构时间包括从记录上下文信息到恢复现场的整个过程。

本文将系统重构时间定义为两部分:①记录上下文信息、打包应用软件数据、解包并恢复现场,这一部分所消耗的时间不好统计且相差不大,因此作为一个常量值;②数据包从初始分区传输到目的分区所消耗的时间,这一部分时间与应用软件数据大小、目的分区都有关系。将两部分结果结合得到重构时间,如(19)式所示。

$$T_{re} = \sum_{i=1}^{|A_{list}|} (T_{A_i,const} + T_{A_i,trans}) \quad (19)$$

式中: T_{re} 表示系统重构时间消耗; $|A_{list}|$ 表示需移出分区的应用软件集合; $T_{A_i,const}$ 表示应用软件 A_i 的固定时间消耗; $T_{A_i,trans}$ 表示应用软件 A_i 的传输时间消耗。

在算法训练完成后,针对任务模式 $S_1 \sim S_8$,系统根据不同任务需求生成了对应的重构蓝图。 S_1 任务模式中,重构蓝图主要保留了初始蓝图中的应用 A_1, A_4, A_6 等应用,同时新增部署了 $A_{12} \sim A_{15}$ 等关键应用。 S_2 至 S_4 中,任务变化导致 A_2, A_3, A_5, A_7 等应用频繁迁移,部分模式还出现 A_6 或 A_{10} 的替换或重部署。 S_5 与 S_6 为多应用变更任务,其中 A_{11}, A_{14}, A_{15} 等为新增或迁移重点, A_1 和 A_4 则根据负载重新分配。 S_7 与 S_8 在处理器和分区之间重新优化了 $A_3, A_{10}, A_{13}, A_{14}$ 的部署位置。

现在对蓝图质量和算法效率进行分析和验证。

3.1 蓝图质量对比

本次实验中,模拟了 IMA 从初始任务模式切换到设计的 8 种任务模式的过程。该过程的实现分别由 DDQN-MS-NN、Q-Learning 和 DQN 算法完成,重构蓝图质量由三部分构成:①负载均衡,负载均衡指系统内部硬件资源富裕的分区挂载更多应用,使得整个系统资源得到合理应用;②重构影响,重构影响表示蓝图重构前后差距,差距越大表面需要进行操作的应用越多,得分越低;③迁移时间,迁移时间越长的蓝图,在系统实际根据蓝图切换任务模式时,迁移过程中出现错误的可能性越高。因此可以认为蓝图质量得分高的蓝图更优秀。图 3~4 是不同算法生成最终蓝图的质量奖励值,奖励值为 10 次结果取平均,将单个应用差异和多个应用差异分开对比。

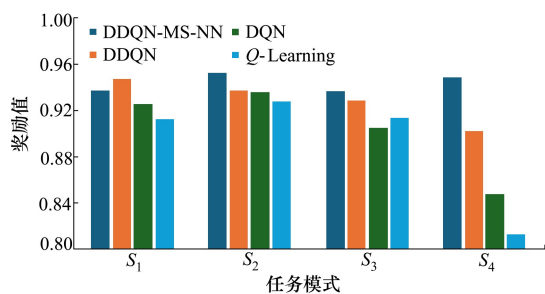


图3 $S_1 \sim S_4$ 重构蓝图质量图

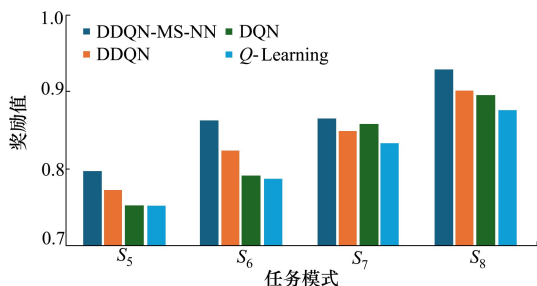


图4 $S_5 \sim S_8$ 重构蓝图质量图

分析图 3~4 可以看出,由 DDQN-MS-NN 算法得出的重构蓝图相比于 Q-Learning、DQN 和 DDQN,无论是在单个应用差异或者多个应用差异的条件下,均有更高的蓝图质量奖励值。

3.2 重构算法性能分析

传统的 DQN 会高估一些情况下的行为价值,DDQN-MS-NN 算法引入了双价值网络机制用于解决预估值比实际值要高的问题。DQN 内部有 1 个预测(predict)网络和 1 个目标(target)网络。DQN 是根据上一轮的目标网络来更新最优动作,这会导致 Q 值高于实际值。而 DDQN-MS-NN 则采用当前的预测网络来选择动作,目标网络用于保证更新的稳定,这样能在一定程度上减小误差,使 Q 值更加接近真实值。因此理论上 DDQN-MS-NN 算法能更快收敛。

对算法的性能进行分析,分别以单个应用不同的重构蓝图 S_1 和多个应用不同的 S_6 为例查看算法随轮次的奖励值增加曲线。图 5 展示蓝图 S_1 下算法奖励曲线。由图 5 可以看出 Q-Learning 和 DDQN-MS-NN 在 5 000 轮次时曲线开始平稳,但 Q-Learning 在后续训练中出现了震荡,而 DQN 曲线相比 Q-Learning 则要稳定许多。DDQN 算法同 DQN 相比,训练过程更平稳且收敛更快。总体而言 DDQN-MS-NN 收敛最快最平稳,且最终奖励值更高。证明在系统进行简单的任务模式切换时,DDQN-MS-NN 比基础的 DDQN、DQN 以及 Q-learning 都更具优势。

$S_5 \sim S_6$ 对应的任务模式切换涉及的应用程序更多,系统内部资源调度变化更大、问题更复杂。以 S_6 对应的重构蓝图为例进行分析,图 6 为 DDQN-MS-NN、Q-Learning、DQN 算法在 20 000 次训练中的奖励值。

对比图 6 中曲线可以发现多个应用差异的情况下,DDQN-MS-NN 算法收敛速度更快,在 6 000 轮次左右就基本收敛,且奖励回报值趋于稳定,未作改进的 DDQN 在 8 000 轮次左右稳定,而 Q-Learning 和 DQN 在 10 000 轮次还未完全收敛,且在 12 000 轮次基本收敛后还出现较大波动。说明 DDQN-MS-NN 算法在复杂条件下相较于 Q-Learning、DQN 和 DDQN,在重构问题上算法收敛性明显更加优秀。

综合结果分析,可以看出面对相同的任务模式切换时,DDQN-MS-NN 算法得出的重构蓝图拥有更高的质量,并且算法收敛速度更快且收敛后奖励值

曲线更稳定。在更复杂的条件下 DDQN-MS-NN 经过前期训练后,后续曲线也能保持稳定。因此可以

验证本文提出的 DDQN-MS-NN 算法在重构尤其是情况复杂时相比 Q-Learning 和 DQN 更加优秀。

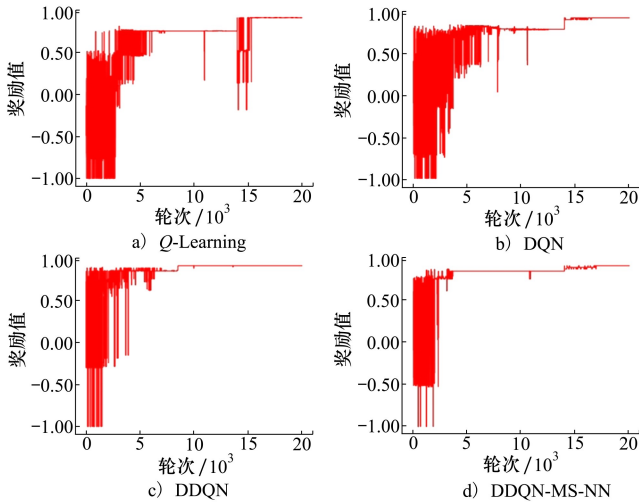


图 5 S₁ 下不同算法奖励值曲线

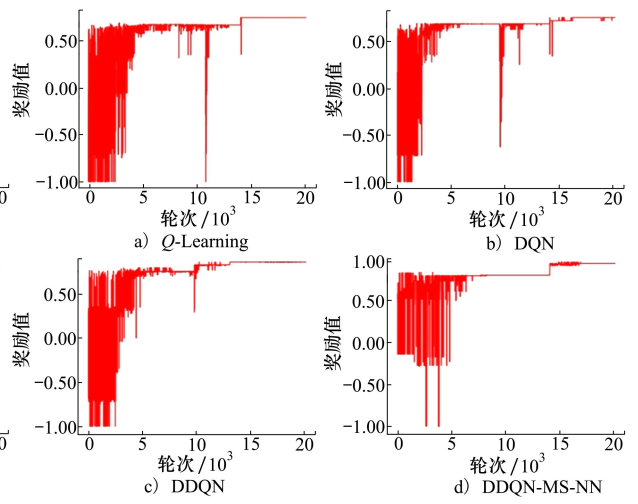


图 6 S₆ 下不同算法奖励值曲线

4 结 论

本文围绕多任务环境下综合模块化航空电子系统的重构方法展开研究,重点探讨了多模式任务调度、资源按需挂载与调度算法改进等关键问题。通过构建任务-应用映射关系与多模式蓝图管理机制,有效提升了系统在资源受限条件下的任务适应

能力。在调度算法方面,引入基于 DDQN 的改进方法 DDQN-MS-NN,解决了传统 DQN 存在的 Q 值高估问题,实现了更精确的决策,同时具有更快的收敛速度。实验结果表明,所提出的方法在任务调度效率、资源利用率以及系统响应能力等方面具有显著优势,为 IMA 系统的智能化重构提供了可行思路与实践基础。

参考文献:

[1] ANNIGHOEFER B, REINHART J, BRUNNER M, et al. Requirements and concept for a self-organizing plug & fly avionics platform[C]//2021 IEEE/AIAA 40th Digital Avionics Systems Conference, San Antonio, 2021: 1-10

[2] REBISCHKE C, ZAESKE W, DURAK U. From the cloud to the clouds: rethinking integrated modular avionics with cloud-native technologies[C]//2023 AIAA Science and Technology Forum and Exposition, 2023

[3] LIU L, ZHAO W, JIANG Z, et al. A modeling method for integrated modular avionics dynamic reconfiguration process based on AADL[J]. Journal of Physics Conference Series, 2020, 1646: 012132

[4] BURGER S, HUMMEL O. Towards automatic reconfiguration of aviation software systems[C]//IEEE Computer Software & Applications Conference Workshops, 2011

[5] LUKIC B, AHLBRECHT A, FRIEDRICH S, et al. State-of-the-art technologies for integrated modular avionics and the way ahead[C]//2023 IEEE/AIAA 42nd Digital Avionics Systems Conference, Barcelona, Spain, 2023: 1-10

[6] WANG L, HONG Y, LI K, et al. A novel IMA resource configuration method based on feature modeling and optimization[C]//2019 3rd International Conference on Electronic Information Technology and Computer Engineering, Xiamen, China, 2019: 1216-1223

[7] MA H, CUI W, YUE M, et al. Modeling and simulation on dynamic reliability of multi-partition software of IMA based on TimeNET4.4[C]//2021 IEEE 6th International Conference on Computer and Communication Systems, Chengdu, China, 2021: 1088-1093

- [8] WEI X, DONG Y, XIAO M. Safety-based software reconfiguration method for integrated modular avionics systems in AADL model[C]//2018 IEEE International Conference on Software Quality, Reliability and Security Companion, Lisbon, Portugal, 2018: 450-455
- [9] HOLLOW P, MCDERMID J, NICHOLSON M. Approaches to certification of reconfigurable IMA systems[J]. INCOSE International Symposium, 2000, 10(1): 372-379
- [10] ZHANG T, CHEN J, LV D, et al. Automatic generation of reconfiguration blueprints for ima systems using reinforcement learning[J]. IEEE Embedded Systems Letters, 2021, 13(4): 182-185
- [11] 陈晓磊. IMA 系统中支持应用软件重构的分区管理技术研究[J]. 电子技术与软件工程, 2014(11): 93-95
CHEN Xiaolei. Research on partition management technology supporting application software reconstruction in IMA system[J]. Electronic Technology and Software Engineering, 2014(11): 93-95 (in Chinese)
- [12] 张涛, 张文涛, 代凌, 等. 基于序贯博弈多智能体强化学习的综合模块化航空电子系统重构方法[J]. 电子学报, 2022, 50(4): 954-966
ZHANG Tao, ZHANG Wentao, DAI Ling, et al. Reconstruction method of integrated modular avionics system based on sequential game multi-agent reinforcement learning[J]. Acta Electronica Sinica, 2022, 50(4): 954-966 (in Chinese)
- [13] 杨威, 李珊, 常磊. 一种基于 ARINC653 的航电应用分区动态重构方法[J]. 信息系统工程, 2020(3): 148-149
YANG Wei, LI Shan, CHANG Lei. A ARINC653-based dynamic reconstruction method for avionics application partitions[J]. Information Systems Engineering, 2020(3): 148-149 (in Chinese)
- [14] 罗庆, 张涛, 单鹏, 等. 基于改进 Q 学习的 IMA 系统重构蓝图生成方法[J]. 航空学报, 2021, 42(8): 327-336
LUO Qing, ZHANG Tao, SHAN Peng, et al. Blueprint generation method for IMA system reconstruction based on improved Q learning[J]. Acta Aeronautica et Astronautica Sinica, 2021, 42(8): 327-336 (in Chinese)
- [15] LIU Z, ZHAO Z. Modeling and schedulability verification of IMA partitioning based on AADL[C]//2017 10th International Symposium on Computational Intelligence and Design, Hangzhou, China, 2017: 417-420
- [16] ZHOU S, WANG S, LIU B. A static load balancing evaluation method for IMA system[C]//2017 2nd International Conference on Reliability Systems Engineering, Beijing, China, 2017: 1-5
- [17] XIANG W, HE F. End-to-end delay analysis considering partition scheduling on a DIMA platform[C]//TENCON 2018-2018 IEEE Region 10 Conference, 2018: 1548-1553
- [18] MELANI A, MANCUSO R, CACCAMO M, et al. A scheduling framework for handling integrated modular avionic systems on multicore platforms[C]//2017 IEEE 23rd International Conference on Embedded and Real-Time Computing Systems and Applications, 2017
- [19] 王满超, 陈海峰, 刘雷, 等. 软件定义导弹技术发展研究[J]. 战术导弹技术, 2020(2): 93-98
WANG Manchao, CHEN Haifeng, LIU Lei, et al. Research on the development of software-defined missile technology[J]. Tactical Missile Technology, 2020(2): 93-98 (in Chinese)

Research on method of comprehensive electronic system reconstruction for multi-tasking

LI Guodong^{1,2}, ZHANG Hangu¹, SHAN Peng³, WANG Xiaohui^{1,4},
ZHOU Binglin¹, ZHAO Anrong¹

1.Northwestern Polytechnical University, Xi'an 710072, China;
2.The First Aircraft Institute, Xi'an 710089, China;
3.Aeronautics Computing Technology Research Institute, Xi'an 710068, China;
4.China Academy of Launch Vehicle Technology, Beijing 100076, China

Abstract: The integrated modular avionics system is widely used in the aviation field due to its flexibility, ease of modification, and high fault tolerance. However, the system faces changing environments and evolving multi-tasking requirements. Existing manual configuration methods and traditional algorithms for generating reconstruction blueprints have limitations in terms of automation and quality assurance. They struggle to meet the increasing complexity and difficulty of resource scheduling during task switching in the comprehensive electronic system. In this paper, the DDQN-MS-NN reconstruction algorithm is proposed to address these challenges. The algorithm focuses on generating multi-tasking reconstruction blueprints for the comprehensive electronic system, improving the automation level and quality of resource scheduling, by introducing multi-step learning and noise network mechanism. Experimental results show that, the DDQN-MS-NN reconstruction algorithm can enhance system stability and resilience compared with traditional algorithms.

Keywords: integrated modular avionics; task reconstruction; reinforcement learning

引用格式: 李国栋, 张翰谷, 单鹏, 等. 面向多任务的综合模块化航空电子系统重构方法[J]. 西北工业大学学报, 2025, 43(4): 821-830

LI Guodong, ZHANG Hangu, SHAN Peng, et al. Research on method of comprehensive electronic system reconstruction for multi-tasking[J]. Journal of Northwestern Polytechnical University, 2025, 43(4): 821-830 (in Chinese)