DOI:10.20079/j.issn.1001-893x.230103001

# 基于 DDPG 的综合化航电系统多分区任务分配优化方法*

赵长啸[1,2],李道俊[1],汪鹏辉[1],田 毅[1,2]

(1. 中国民航大学 安全科学与工程学院,天津 300300;2. 民航航空器适航审定技术重点实验室,天津 300300)

**摘 要**:综合化航电系统(Integrated Modular Avionics,IMA)通过时空分区机制实现共享资源平台下的多航电功能集成,分区间的任务分配方法的优劣决定着航电系统的整体效能。针对航电任务集合在多分区内的分配调度问题,提出了一种基于深度强化学习的优化方法。构建了航电系统模型与任务模型,以系统资源限制与任务实时性需求为约束,以提高系统资源利用率为优化目标,将任务分配过程描述为序贯决策问题。引入马尔科夫决策模型,建立基于深度确定性策略梯度(Deep Deterministic Policy Gradient,DDPG)法的 IMA 任务分配模型并提出通用分配架构;引入状态归一化、行为噪声等策略训练技巧,提高 DDPG 算法的学习性能和训练能力。仿真结果表明,提出的优化算法迭代次数达到 500 次时开始收敛,分析 800 次之后多分区内驻留任务方案在能满足约束要求的同时,最低处理效率提升 20.55%。相较于传统分配方案和 AC(Actor-Critic)算法,提出的 DDPG 算法在收敛能力、优化性能以及稳定性上均有显著优势。

**关键词**:综合模块化航空电子系统(IMA);任务分配及调度;深度强化学习;DDPG 算法

**开放科学(资源服务)标识码(OSID)**:



**中图分类号**:V243 **文献标志码**:A **文章编号**:1001-893X(2024)01-0058-09

# A DDPG-based Optimization Method for Multi-partition Task Assignment of IMA

ZHAO Changxiao[1,2], LI Daojun[1], WANG Penghui[1], TIAN Yi[1,2]

(1. College of Safety Science and Engineering, Civil Aviation University of China, Tianjin 300300, China;
2. Key Laboratory of Civil Aircraft Airworthiness Technology, Tianjin 300300, China)

**Abstract**: The integrated modular avionics (IMA) system implements the integration of multiple avionics functions under a shared resource platform through a spatio-temporal partitioning mechanism. The merit of the task distribution method between partitions determines the overall effectiveness of the IMA system. An optimization method based on deep reinforcement learning (DRL) is proposed for the distribution and scheduling of avionics task sets within multiple partitions is proposed. The IMA system model and task model are constructed, and the constraints of system resource and task real-time requirements are used to improve the system resource utilization as the optimization objective. The task distribution process is described as a sequential decision problem. A Markov decision model is introduced to develop a deep deterministic policy gradient (DDPG) algorithm-based IMA task distribution model and a generic distribution architecture is proposed. Policy training techniques such as state normalization and behavioral noise are introduced to improve the learning performance and training capability of the DDPG algorithm. Simulation results show that the proposed optimization algorithm starts to converge after 500 iterations, and the efficiency of distribution scheme is improved by 20.55% while satisfying the constraint requirements after 800 iterations. Compared with the traditional assignment scheme and the Actor-Critic (AC) algorithm, the proposed DDPG algorithm has significant advantages in terms of convergence ability.

**Key words**: integrated modular avionics (IMA); task allocation and scheduling; deep reinforcement learning; DDPG algorithm

* **收稿日期**:2023-01-03;**修回日期**:2023-03-07
**基金项目**:国家重点研发计划(2021YFB1600601);天津市自然科学基金(21JCQN JC00900)
**通信作者**:田毅 Emai:ytian@cauc.edu.cn

## 0 引　言

随着信息技术、高性能计算技术和航空电子技术的发展,综合模块化航空电子(Integrated Modular Avionics,IMA)系统已经成为当前民用航空领域的主流架构。相较于传统的分立式、联合式航电系统,IMA 系统将多个独立的子系统综合起来,具备多个应用同时共享系统内软硬件资源的能力[1],在提高资源利用率、增强系统灵活性和适应性等方面具有很多优势[2]。航电系统作为飞机的"大脑"与"神经中枢",其任务的执行不仅有较高的资源保证需求,对实时性也有严苛的要求,对于超出规定时限的任务,即使完成也会被判定任务失败,并直接影响飞机的飞行安全。因此,有效、合理的分区任务分配方法成为保证航电系统任务有效执行的基础,该问题也受到了学者和航电系统研发单位的重视。

航电系统任务调度问题主要涉及任务分配、任务优先级分配、可调度性分析[3]等问题,其中首先需要解决的是在满足可调度性[4]前提下,如何在多个分区内进行任务分配。文献[5]通过模拟退火算法解决实时系统的任务分配问题,文献[6]通过基于模拟退火的改进 Q 学习算法来对软件和硬件资源进行重构配置,重新分配任务方案。文献[7]将基于遗传算法的多目标优化方法应用于实时任务的调度问题。文献[8]在确保任务模块间执行顺序的前提下,采用分支定界的启发式算法,保证任务的实时调度。文献[9]提出一种基于序贯博弈多智能体强化学习的 IMA 系统重构方法,根据应用软件优先级确定博弈顺序,模拟应用软件重新分配过程,优化任务分配策略。随着计算技术的发展,以深度学习和强化学习为代表的人工智能技术主要运用于对数据的分析处理,深度强化学习结合了深度学习较强的感知能力和强化学习优秀的决策能力,已经成功应用在自动驾驶、资源分配、机器控制和优化调度等领域中,为复杂系统的决策问题提供了新的思路。航电系统任务在多分区内的分配优化,属于决策制定的优化问题。基于此,本文选择深度强化学习的算法作为优化问题的求解方法。

综合模块化航空电子系统相较于传统航电系统在配置灵活性、综合性以及硬件集成度方面均有改进,为了充分利用系统软硬件资源,提高系统性能,本文提出一种基于深度确定性策略梯度(Deep Deterministic Policy Gradient,DDPG)的任务调度优化算法,对任务集合在不同分区之间的分配调度进行优化。

## 1 系统模型

IMA 系统是高度集成、资源共享的开放式平台[10],通过分区技术进行航电功能的驻留,一个处理节点可以包含多个分区。本节对航电系统分区及任务模型进行建模。

### 1.1 综合化航电系统模型

IMA 系统通常采用分布式的处理框架[11],系统内含有多个通过 AFDX 网络相连接的嵌入式处理节点,每个处理节点内可设置多个分区,每个分区内可分配一个或多个预先定义的实时任务,如图 1 所示。

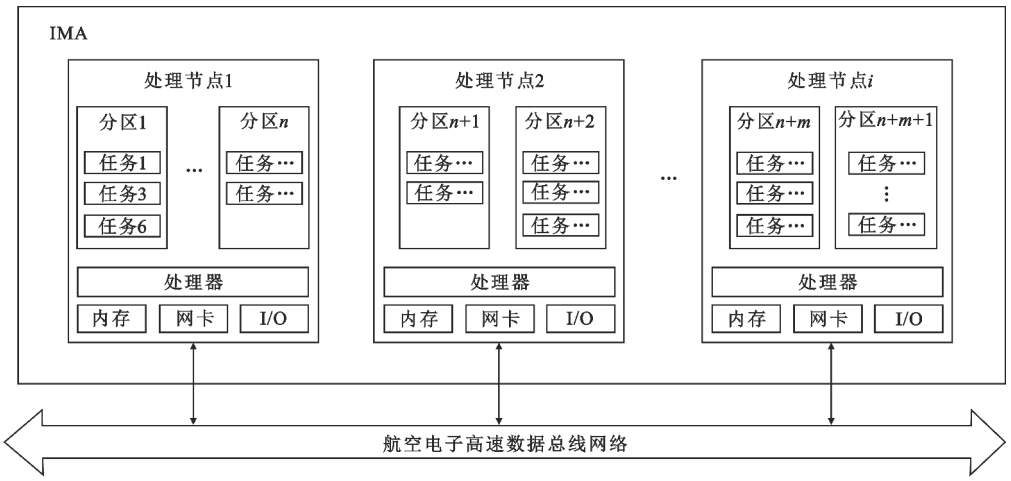


图 1　IMA 系统框架

Fig. 1 IMA system framework

### 1.2 处理节点集

IMA 系统内包含一组同构的处理节点,处理节点通过端系统连入 AFDX 网络,并可以通过多级交换机实现信息的交换。

处理节点集可以表示为

$$N=\{N_i \mid 1\leqslant i\leqslant n^N\} \tag{1}$$

式中:$n^N$ 表示系统内处理节点的个数。每个处理节点都有内存、执行时间和带宽等资源上限,令 $M$ 表示各处理节点的内存资源上限,$T$ 表示各处理节点的处理能力,$B$ 表示各处理节点的带宽。综上,处理节点可表述为一个三元数组 $N_i(M_i,T_i,B_i)$。

### 1.3 分区集

在 ARINC 653 标准中,分区的定义指的是对航空电子功能按照某种策略进行分组,对运行在核心模块上的多个应用程序按功能可划分为多个分区。分区内的所有任务共享分区所占有的系统资源,操作系统对分区所占用的处理时间、内存和其他资源拥有控制权,从而使得处理节点中各分区相互独立。分区集可以表示为

$$P=\{P_i \mid 1\leqslant i\leqslant n^P\} \tag{2}$$

式中:$n^P$ 表示系统内所有处理节点内分区个数之和。每个分区有以下属性:所属的处理节点序号 $C$;内存资源占用系数 $\alpha^M$,表示所占用的处理节点内存资源比率;处理资源占用系数 $\alpha^T$,表示所占用的处理节点处理能力比率。

综上,分区可表述为一个三元数组 $P_i=(C_i,\alpha_i^M,\alpha_i^T)$。

### 1.4 任务集

航电系统通过在系统平台驻留应用实现系统功能,不同的航电系统通过任务进行具体功能的实现。一个航电功能内的按顺序执行的所有任务称为一个任务集,任务可以按照一定的策略在多个分区内进行执行,以满足任务资源需求与实时性需求。

任务集表示为

$$G=\{G_i \mid 1\leqslant i\leqslant n^G\} \tag{3}$$

式中:$n^G$ 表示系统内应用程序的个数,即任务集的个数。每个任务集由以下属性组成:任务集负载 GL,任务集内所有任务负载之和;任务集所需内存资源 GM,任务集内所有任务所需内存资源之和;任务集所在的分区序号 GC;任务集在分区内的优先级 GP。

综上,任务集可以表述为一个四元数组 $G_i$,$G_i$ 中包含的任务记为 $\{\tau_{i,1},\tau_{i,2},\cdots,\tau_{i,n}\}$,任务 $\tau_{i,j}$ 可以描述为一个三元数组 $\tau_{i,j}=(\mathrm{GL}_{i,j},\mathrm{GM}_{i,j},\mathrm{GD}_{i,j})$,$\mathrm{GL}_{i,j}$ 表示任务 $\tau_{i,j}$ 的负载,$\mathrm{GM}_{i,j}$ 表示任务 $\tau_{i,j}$ 所需内存资源,$\mathrm{GD}_{i,j}$ 表示任务 $\tau_{i,j}$ 的截止时间。

同一个任务集内的任务可以不限定在同一个分区内执行,可以在相同处理节点的不同分区以及不同处理节点内的分区中执行。处理节点之间通过 AFDX 交换机进行互连,所以在不同处理节点之间进行任务调度需要考虑经过交换机的时延 $Z_{i,j}$,计算公式定义如下:

$$Z_{i,j}=\sum_{m=1}^{n^{\mathrm{switch}}}\frac{H_{i,j,m}\cdot\sum\limits_{\tau_{k,n}\in\mathrm{dip}(k,j)}\mathrm{GM}_{k,n}}{B_m^{\mathrm{switch}}} \tag{4}$$

式中:$n^{\mathrm{switch}}$ 为 AFDX 网络中交换机的个数;$\mathrm{dip}(k,j)$ 为任务集 $k$ 中被调度到分区 $j$ 的任务集合;$B_m^{\mathrm{switch}}$ 为 AFDX 网络内交换机 $m$ 的带宽;$H_{i,j,m}$ 取值为 1 或者 0,表示从分区 $i$ 所属处理节点 $C_i$ 到分区 $j$ 所属处理节点 $C_j$ 之间是否经过交换机 $m$。

综上,任务集 $K$ 的调度任务集合由分区 $i$ 调度到分区 $j$ 进行处理的数据传输延迟可以用以下公式表示:

$$\mathrm{DTT}_k=\frac{\sum\limits_{\tau_{k,n}\in\mathrm{dip}(k,j)}\mathrm{GM}_{k,n}}{B_{C_i}}+Z_{i,j} \tag{5}$$

## 2 航电系统多分区任务分配问题建模

航电系统多分区任务分配问题,可以描述为任务集合 $G$ 在分区集合 $P$ 上的一个映射问题,即

$$G\rightarrow P \tag{6}$$

其在分配过程中需要考虑资源、实时性等约束,本文将其建模为多约束优化问题。

### 2.1 IMA 系统任务分配约束

不同的任务有不同的功能资源需求,需要利用有限的资源,为运行在 IMA 系统上不同优先级的任务提供资源保证。因此,任务分配具有很多的约束条件。

#### 2.1.1 实时性约束

任务集中每个任务都有实时性要求,即要求所有任务的完成时间不超过该任务截止时间。该约束可以描述为

$$\mathrm{CT}_{i,j}\leqslant\mathrm{GD}_{i,j} \tag{7}$$

$$\mathrm{CT}_{i,j}=\begin{cases}\sum\limits_{m=1}^{j-1}T_{i,m}^{\mathrm{compute}}+T_{i,j}^{\mathrm{compute}} & \tau_{i,j}\notin\mathrm{dip}(i,n)\\ \max\left(\sum\limits_{m=1}^{j-1}T_{i,m}^{\mathrm{compute}},\sum\limits_{p}^{n^G}A_{p,j}\cdot\mathrm{GCT}_p\right)+T_{i,j}^{\mathrm{compute}}+\mathrm{DDT}_i & \tau_{i,j}\in\mathrm{dip}(i,n)\end{cases} \tag{8}$$

$$T_{i,j}^{\text{compute}}=\begin{cases}\dfrac{\text{GL}_{i,j}}{\alpha_i\cdot T_{C_i}} & \tau_{i,j}\notin \text{dip}(i,n)\\ \dfrac{\text{GL}_{i,j}}{\alpha_n\cdot T_{C_n}} & \tau_{i,j}\in \text{dip}(i,n)\end{cases} \tag{9}$$

式中:$\text{CT}_{i,j}$ 为任务 $\tau_{i,j}$ 的完成时间;$T_{i,j}^{\text{compute}}$ 为任务 $\tau_{i,j}$ 的处理时间;$\text{GCT}_p$ 为任务集 $p$ 的完成时间;$A_{p,j}$ 的取值为 1 或者 0,表示任务集 $p$ 是否被分配到分区 $j$。

### 2.1.2　内存约束

由于各个分区的内存有限,需要保证分配到同一个分区的任务内存之和不超过分区的内存资源:

$$\sum_{i=1}^{n^G}\sum_{k=1}^{n^{G_i}}\Psi_{i,k,j}\cdot \text{GM}_{i,k}\leqslant\alpha_j^M\cdot M_{C_j} \tag{10}$$

式中:$n^{G_i}$ 表示任务集 $i$ 中任务的个数;$\Psi_{i,k,j}$ 取值为 1 或者 0,表示任务 $\tau_{i,k}$ 是否分配到分区 $j$。

### 2.1.3　唯一性约束

在同一时刻,只能按照任务集中的执行顺序调度一个或者多个任务,保证在数据传输过程中类似数据冲突等影响安全性的事件不会发生。因此,唯一性约束可描述为

$$\begin{gathered}\eta_i(t)\in\{0,1\},\forall t\in\{1,2,\cdots,I\},i\in\{1,2,\cdots,n^G\}\\ \sum_{i=1}^{n^G}\eta_i(t)=1,\forall t\end{gathered} \tag{11}$$

## 2.2　优化目标

为了确保有效利用有限的计算资源、内存资源,以进一步优化系统任务的分配调度,这里定义了平均完成时间、均衡分区负载两个优化目标。

### 2.2.1　平均完成时间

通过联合优化任务分配,使系统内任务集平均完成时间最小化。具体而言,平均完成时间(Average Completion Time, ACT)的公式可以描述如下:

$$\text{ACT}=\sum_{i=1}^{n^G}\text{GCT}_i/n^G \tag{12}$$

$$\text{GCT}_i=\max(t_{\text{loc},i},t_{\text{nloc},i}) \tag{13}$$

$$t_{\text{loc},i}=\sum_{k\in\text{hp}(i)}\text{GCT}_K+\frac{\text{GL}_i-\sum\limits_{\tau_{k,n}\in\text{dip}(k,j)}\text{GL}_{k,n}}{\alpha_i^T\cdot T_{C_i}} \tag{14}$$

$$t_{\text{nloc},i}=\text{DDT}_i+\sum_{m=1}^{n^G}A_{m,j}\cdot\text{GCT}_m+\frac{\sum\limits_{\tau_{k,n}\in\text{dip}(k,j)}\text{GL}_{k,n}}{\alpha_j^T\cdot T_{C_j}} \tag{15}$$

式中:$\text{hp}(i)$ 为分区内优先级高于任务集 $i$ 的任务集集合;$t_{\text{loc},i}$ 为任务集 $i$ 的本地任务完成时间;$t_{\text{nloc},i}$ 为任务集 $i$ 调度到分区 $j$ 内的任务集合完成时间。

### 2.2.2　均衡分区负载

均衡各个处理节点内分区负载有助于提高任务的执行速度,使用标准差来衡量每个处理节点内分区荷载的离散程度。因此,HB 计算公式为

$$\text{HB}=\frac{1}{n^P}\sum_{i=1}^{n^P}(\text{load}_i-\overline{\text{load}})^2 \tag{16}$$

$$\text{load}_i=\mu_1\times C_i^{\text{use}}+\mu_2\times M_i^{\text{use}},\mu_1+\mu_2=1 \tag{17}$$

$$\overline{\text{load}}=\frac{1}{n^P}\sum_{i=1}^{n^P}\text{load}_i \tag{18}$$

$$C_i^{\text{use}}=\frac{\sum\limits_{n=1}^{n^G}\sum\limits_{k=1}^{n^{G_n}}\Psi_{n,k,i}\cdot T_{n,k}^{\text{compute}}}{\max(\text{GCT}_1,\text{GCT}_2,\cdots,\text{GCT}_{n^G})} \tag{19}$$

$$M_i^{\text{use}}=\frac{\sum\limits_{n=1}^{n^G}\sum\limits_{k=1}^{n^{G_n}}\Psi_{n,k,i}\cdot\text{GM}_{n,k}}{\alpha_i^M\cdot M_{C_i}} \tag{20}$$

式中:$\overline{\text{load}}$ 表示分区平均负载;$\text{load}_i$ 表示分区 $i$ 的负载;$C_i^{\text{use}}$ 表示分区 $i$ 计算资源利用率;$M_i^{\text{use}}$ 表示分区 $i$ 的内存资源利用率;$\mu_1$ 和 $\mu_2$ 表示计算资源和内存资源的权重;$\Psi_{n,k,i}$ 取值为 1 或 0,表示任务 $\tau_{n,k}$ 是否被分配到分区 $i$ 执行。

综上,该问题的优化目标函数为

$$\min(\lambda_1\cdot\text{ACT}+\lambda_2\cdot\text{HB}),\lambda_1+\lambda_2=1 \tag{21}$$

# 3　基于 DDPG 算法的 IMA 任务分配

引入基于 IMA 系统状态信息输入的、具有连续动作输出的深度强化学习来优化系统任务分配问题,提高系统执行效率。DDPG 算法采用了深度 Q 网络(Deep Q-Network, DQN)的思想和基于 Actor-Critic(AC)的算法结构,弥补了 DQN 算法不能处理连续动作空间问题的缺点[12],具有较好的收敛能力、较强的训练能力以及优秀的稳定性[13]。DDPG 是深度强化学习的方法之一,其基于深度学习的部分,通过神经网络拟合泛化网络参数;其基于强化学习的部分,通过策略梯度算法使得智能体根据学习经验选择最优或较优的策略,并确定性地选择动作。

## 3.1　DDPG 算法原理

DDPG 算法分为 Actor-Critic 网络、IMA 系统模型、奖励函数等部分[14]。本文中将 IMA 系统的观察信息,即 IMA 系统内处理节点、分区、任务集状态共同组合成该时刻的状态信息 $s_t$ 输入至神经网络。

Actor-Critic 网络根据 IMA 系统此时状态向 IMA 系统提供相应的动作 $a_t$,IMA 系统执行该动作与环境交互,根据任务分配完成得到的下一个状态 $s_{i+1}$ 获得奖励。IMA 系统的目标是通过学习最优策略 $\pi:S\rightarrow A$,来最大化累计回报奖励 $R_t$。同时为了提高对于系统环境的探索能力,通过添加行为噪声 $N_i$ 引导算法学习新的策略,一般设定行为噪声服从高斯分布[15]。综上,DDPG 算法框架如图 2 所示。

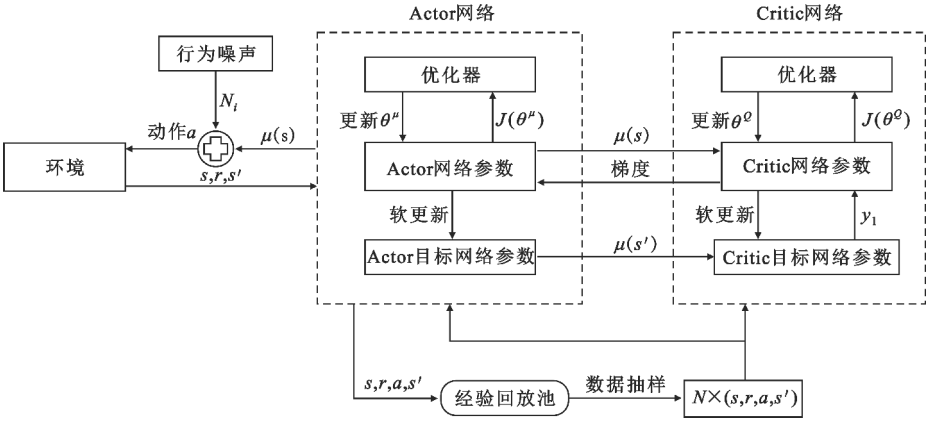


**图 2 DDPG 算法框架**

Fig. 2 DDPG algorithm framework

### 3.2 Actor-Critic 网络

Actor-Critic 网络具有值函数、策略函数更新方式的优点[16]。其中,Critic 网络根据值函数单步更新,评价 IMA 系统当前分配策略好坏,网络参数为 $\theta^Q$;Actor 网络基于当前系统状态确定性输出任务分配策略,网络参数为 $\theta^{\mu}$。

同时 Actor-Critic 网络各自细分为评估神经网络与目标神经网络两部分。为了减少更新目标扰动带来的神经网络更新困难,评估神经网络与目标神经网络之间通过软更新(soft update)方式传参[16],公式如下所示:

$$\theta^{\mu'}\leftarrow\tau\theta^{\mu}+(1-\tau)\theta^{\mu'} \tag{22}$$

$$\theta^{Q'}\leftarrow\tau\theta^{Q}+(1-\tau)\theta^{\mu'} \tag{23}$$

式中:$\tau$ 为滑动平均系数,一般取 0.01 保证更新平缓;$\theta^{\mu}$ 和 $\theta^{Q}$ 为 Actor-Critic 评估神经网络参数;$\theta^{\mu'}$ 和 $\theta^{Q'}$ 为 Actor-Critic 目标神经网络参数。

Critic 网络使用全连接神经网络拟合每一个状态动作值[17],评估神经网络通过输入系统状态与采取的动作,输出状态-动作对的 Q 值,表示对该动作策略的评价;目标神经网络的输入来自 Actor 目标神经网络生成的下一时刻动作,输出下一步状态-动作对的 Q 值。Critic 评估神经网络 $Q(s,a|\theta^Q)$ 可以更新如下:

$$L(\theta^Q)=\frac{1}{N}\sum_i(r_i+\gamma Q'(s_{i+1},\mu'(s_{i+1})|\theta^Q)-Q(s_i,a_i|\theta^Q))^2 \tag{24}$$

Actor 网络确定性地将状态 $s_t$ 映射到动作 $a_t$,即基于当前系统状态给出任务分配的确定性动作策略[18]。Actor 评估神经网络根据 Critic 网络给出的 Q 值,通过梯度上升指导网络参数向更高评价方向更新,如下式所示:

$$\nabla_{\theta^{\mu}}J(\theta^{\mu})\approx\frac{1}{N}\sum_i(\nabla_a Q(s_i,a_i|\theta^Q)\cdot\nabla_{\theta^{\mu}}\mu(s_i|\theta^{\mu})) \tag{25}$$

### 3.3 状态空间

状态空间 $S$ 中包含了当前环境可以观察到的全部或部分信息,并根据观察到的信息来选择动作。为了实现对任务分配的优化,需要获取的状态信息为系统内任务集的完成时间、系统内分区待处理任务负载之和以及系统内分区剩余内存资源。因此,状态空间 $S$ 表示为

$$S=\{E_{t,1},E_{t,2},\cdots,E_{t,n^G},E_{l,1},E_{l,2},\cdots,E_{l,n^P},E_{rm,1},E_{rm,2},\cdots,E_{rm,n^P}\} \tag{26}$$

### 3.4 动作空间

智能体根据系统状态和观察的环境信息,选择待分配的任务集、分配的任务个数以及分配到处理节点中分区的序号,动作空间 $a$ 表示为

$$a=\{D_{g\_\text{task}},D_{\text{number}},D_{\text{part}}\} \tag{27}$$

### 3.5 奖励函数

设置合适的奖励函数可以引导算法学习到更好的策略,提高收敛的速度[19]。本文的研究目的是优

化任务分配方案,使得系统运行更加高效,减少系统运行时间以及均衡负载,同时在分配过程中满足系统内约束条件。因此,奖励函数定义为

$$R(s,a)=\begin{cases}-E_{\mathrm{rm},i},\exists E_{\mathrm{rm},i}<0,i=\{1,2,\cdots,n^P\}\\ \mathrm{GD}_{i,j}-\mathrm{CT}_{i,j},\exists \mathrm{CT}_{i,j}>\mathrm{GD}_{i,j},i=\{1,2,\cdots,n^G\},j=\{1,2,\cdots,n^{G_i}\}\\ -(\lambda_1\cdot \mathrm{ACT}+\lambda_2\cdot \mathrm{HB}),\text{满足约束}\end{cases} \tag{28}$$

式中分别表示了三种状态:第一,存在分区不满足内存资源约束,则对这次分配给予超出内存资源的负奖励;第二,存在任务超出完成时限,则对这次分配给予超出时限的负奖励;第三,在满足约束的情况下,最小化平均完成时间及负载均衡,基于优化指标权重对 ACT 和 HB 之和取负值作为奖励。

### 3.6　状态归一化

在神经网络的训练过程中,每一层的输入都会随着前一层参数的变化而变化,这就对参数初始化以及学习率的选择有更高的要求。在 IMA 系统环境中,$E_{t,1},\cdots,E_{t,n^G},E_{l,1},\cdots,E_{l,n^P},E_{rm,1},\cdots,E_{rm,n^P}$ 属于不同的序列,可能导致训练中出现问题。所以,使用一种状态归一化算法对状态观测信息做预处理,防止问题出现[20]。在状态归一化算法中,使用 3 个尺度因子,分别为 $\gamma_t$(缩放系统任务集完成时间)、$\gamma_l$(缩放分区待处理指令条数)和 $\gamma_{rm}$(缩放分区剩余内存资源[21])。算法伪代码如下:

Input:

All state variables that require to be normalized: $E_{t,1},\cdots,E_{t,n^G},E_{l,1},\cdots,E_{l,n^P},E_{rm,1}\cdots,E_{rm,n^P}$

Scale Factors: $\gamma_t,\gamma_l,\gamma_{rm}$

Output:

Normalized variables: $E'_{t,1},\cdots,E'_{t,n^G},E'_{l,1},\cdots,E'_{l,n^P},E'_{rm,1}\cdots,E'_{rm,n^P}$

1 function StateNormalization ( $E_{t,1},\cdots,E_{t,n^G},E_{l,1},\cdots,E_{l,n^P},E_{rm,1}\cdots,E_{rm,n^P}$)

2　　$E'_{t,k}=\gamma_t E_{t,k},k=1,\cdots,n^G$

3　　$E'_{l,k}=\gamma_t E_{l,k},k=1,\cdots,n^P$

4　　$E'_{rm,k}=\gamma_{rm}E_{rm,k},k=1,\cdots,n^P$

5　　return $E'_{t,1},\cdots,E'_{t,n^G},E'_{l,1},\cdots,E'_{l,n^P},E'_{rm,1}\cdots,E'_{rm,n^P}$

6 end function

7 $S'$ = StateNormalization ( $E_{t,1},\cdots,E_{t,n^G},E_{l,1},\cdots,E_{l,n^P},E_{rm,1}\cdots,E_{rm,n^P}$)

8 return $S'$

## 4　仿真实验

系统硬件环境为 CPUi12 - 12600KF,内存 32 GB,显卡 AMD 5700XT;软件环境为 Windows10。实验旨在分析 DDPG 算法对 IMA 系统任务分配能力的优化,测试不同指标权重和不同算法参数对仿真效果的影响。同时引入不同环境和方案进行对比,分析 DDPG 算法的收敛性能和算法效率。算法部分伪代码如下:

Input:

Training episode length E, Training sample length $L$;

Actor learning rate $\alpha_{\mathrm{Actor}}$, Critic learning rate $\alpha_{\mathrm{Critic}}$;

Discount factor $\gamma$, soft update factor $\tau$

Experience replay buffer $E_{\mathrm{pool}}$, least batch size $\mathrm{batch}_{\min}$

Gaussian noise $N_i$

Output: The actor network $\mu(\hat{s}|\theta^\mu)$

1 Randomly initialize the weights of actor network $\theta^\mu$ and critic network $\theta^Q$

2 Initialize the target network with weights $\theta^\mu\leftarrow\theta^{\mu'}$ and $\theta^Q\leftarrow\theta^{Q'}$

3 Empty the experience replay buffer $E_{\mathrm{pool}}$

4 for each episode $e=1,2,\cdots,E$ do

5　Reset simulation parameters of the IMA system and obtain initial observation $s_i$

6　for $i=1,2,\cdots,L$ do

7　　Normalize state $s_i$ to $\hat{s}_i'$

8　　Get the action $a_i$ with actor network $\theta^\mu$ and the gaussian noise $N_i$:

$$a_i=\min(\max(\mu(\hat{s}_i|\theta^\mu)+N_i,-1),1)$$

9　　Get the reward $r_i$ and observe next state $s_{i+1}$ after performing action $a_i$

10　　Normalize next state $s_{i+1}$ to $\hat{s}'_{i+1}$

11　　if $E_{\mathrm{pool}}$ is not full then

12　　Store transition $(\hat{s}_i,a_i,r_i,\hat{s}_{i+1})$ in $E_{\mathrm{pool}}$

13　　else

14　　Randomly replace a transition in replay buffer $E_{\mathrm{pool}}$ with $(\hat{s}_j,a_j,r_j,\hat{s}_{j+1})$,

$\forall j=1,2,\cdots,L$ from $E_{\mathrm{pool}}$

15　　Set $y_j=r_j+\gamma Q'(\hat{s}_{j+1},\mu(\hat{s}_{j+1}|\theta^{\mu'},\theta^{Q'}))$

16　　Update the $\theta^Q$ in critic network by minimizing the loss:

$$L(\theta^Q)=\frac{1}{N}\sum_{j=1}^{N}((y_j-Q(\hat{s}_j,a_j|\theta^Q))^2)$$

17　　Update actor network $\theta^\mu$ by the sampled policy gradient

18　　Soft update the critic target network with $\tau$: $\theta^{\mu'}\leftarrow\tau\theta^\mu+(1-\tau)\theta^{\mu'}$

19　　Soft update the actor target network with $\tau$: $\theta^{Q'}\leftarrow\tau\theta^Q+(1-\tau)\theta^{Q'}$

20 end if

21 end for

22 end for

23 return the actor network $\mu(\hat{s}|\theta^{\mu})$

### 4.1 仿真实验设置

实验设定的 IMA 系统基本配置信息如表 1 和表 2 所示,表中 $\tau_{i,j}$ 为系统模型章节中任务集 $G_i$ 包含的任务,$j$ 表示在任务集中的任务执行优先级。同时所有处理节点以及网络中交换机的带宽都设置为 100 Mb/s,AFDX 网络中交换机个数设置为 1(即所有处理节点通过一个交换机互联),优化目标权重系数 $\lambda_1$ 和 $\lambda_2$ 都设置为 0.5。算法初步设定训练参数如表 3 所示。

**表 1 系统任务属性数据**

Tab. 1 System task attribute data

| 任务序号 | 截止时间/ms | 指令条数/$10^6$ | 内存/kb | 执行顺序($\tau_{i,j}$) |
|---|---|---|---|---|
| $Task_1$ | 60 | 6.4 | 90 | $\tau_{2,1}$ |
| $Task_2$ | 60 | 6.4 | 50 | $\tau_{3,2}$ |
| $Task_3$ | 60 | 6.4 | 30 | $\tau_{6,3}$ |
| $Task_4$ | 60 | 3.2 | 40 | $\tau_{6,1}$ |
| $Task_5$ | 60 | 4.0 | 70 | $\tau_{6,2}$ |
| $Task_6$ | 30 | 8.0 | 90 | $\tau_{6,4}$ |
| $Task_7$ | 60 | 8.0 | 80 | $\tau_{1,1}$ |
| $Task_8$ | 30 | 8.0 | 60 | $\tau_{4,2}$ |
| $Task_9$ | 60 | 8.0 | 50 | $\tau_{5,2}$ |
| $Task_{10}$ | 60 | 9.6 | 95 | $\tau_{5,1}$ |
| $Task_{11}$ | 60 | 6.0 | 60 | $\tau_{5,3}$ |
| $Task_{12}$ | 60 | 8.0 | 75 | $\tau_{1,2}$ |
| $Task_{13}$ | 60 | 4.8 | 30 | $\tau_{3,1}$ |
| $Task_{14}$ | 30 | 6.4 | 50 | $\tau_{4,1}$ |
| $Task_{15}$ | 60 | 6.4 | 60 | $\tau_{2,2}$ |

**表 2 IMA 系统任务初始分配方案**

Tab. 2 Initial allocation scheme of IMA system tasks

| 处理节点 | 分区 | 内存/kb | 主频/GHz | 驻留任务 |
|---|---|---|---|---|
| $C_1$ | $P_1$ | 128 | 1.6 | $Task_{2,13}$ |
| | $P_2$ | 128 | 2.0 | $Task_{1,15}$ |
| $C_2$ | $P_1$ | 512 | 3.2 | $Task_{3,4,5,6}$ |
| $C_3$ | $P_1$ | 256 | 2.0 | $Task_{7,12}$ |
| | $P_2$ | 256 | 2.0 | $Task_{8,14}$ |
| $C_4$ | $P_1$ | 256 | 2.0 | $Task_{9,10,11}$ |

**表 3 DDPG 算法初步设定参数**

Tab. 3 Preliminary setting parameters of DDPG algorithm

| 参数 | 设置 |
|---|---|
| 单回合训练步长 | 20 |
| Critic 学习率 | 0.01 |
| Actor 学习率 | 0.02 |
| 软更新参数 $\tau$ | 0.01 |
| 最大训练回合 | 1 000 |
| 经验缓冲区容量 | 10 000 |
| Mini 批大小 | 64 |
| 奖励折扣因子 | 0.9 |

### 4.2 学习率与奖励衰减分析

首先测试不同的学习率 $\alpha_{Actor}$ 和 $\alpha_{Critic}$ 以及奖励衰减因子 $\gamma$ 对优化训练过程的影响,确定参数的最优值用于算法的比较。在初步设定奖励衰减因子 $\gamma=0.9$ 的情况下,取不同的学习率进行训练,结果如图 3 所示。

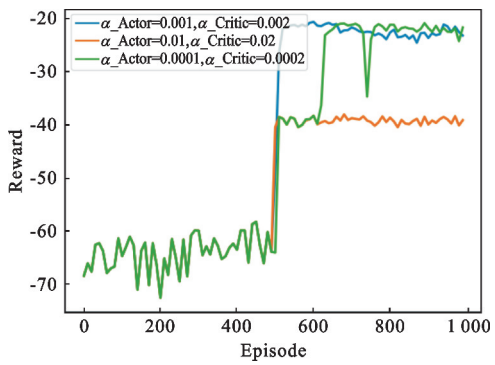


**图 3 不同学习率下奖励回报曲线**

Fig. 3 Reward return curve under different learning rates

从图 3 可以清楚地看到,前两种情况下本文提出的算法都可以收敛,但是当 $\alpha_{Actor}=0.01$,$\alpha_{Critic}=0.02$ 时,算法收敛到局部最优解。可能的原因是当 Actor 网络和 Critic 网络拥有较大学习率时,更新步骤会更长。其次,可以发现当学习率很小时,即 $\alpha_{Actor}=0.000\ 1$,$\alpha_{Critic}=0.000\ 2$,算法不能稳定收敛。因此,Actor 网络和 Critic 网络学习率最优值分别为 $\alpha_{Actor}=0.001$,$\alpha_{Critic}=0.002$。

在确定了学习率的情况下,比较了不同奖励衰减因子 $\gamma=0.6$ 对算法收敛性能的影响,结果如图 4 所示。虽然图中曲线上升幅度相差不大,但是当奖励衰减因子 $\gamma=0.6$ 时,训练后的任务分配策略性能最佳。$\gamma$ 越大,表示将整个时间段收集的数据视为长期数据,对于后续奖励的占比越大。因此,选择适当的 $\gamma$ 值有助于提高训练后策略的性能,在接下来的实验中将奖励衰减因子设置为 0.6。

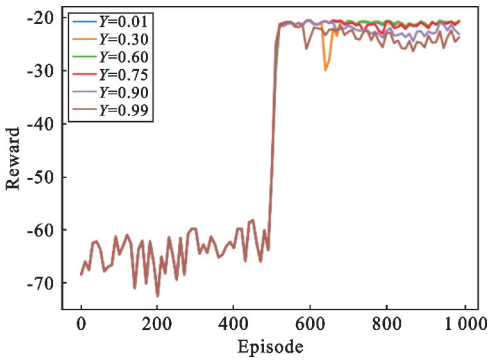


**图 4　不同奖励衰减因子下奖励回报曲线**

Fig. 4 Reward return curve under different reward attenuation factors

### 4.3　算法性能分析

由上述分析确定 DDPG 算法最终参数选择如表 4 所示,本小节基于确定的参数将进行 AC、DDPG、不带有状态归一化的 DDPG、不带有行为噪声的 DDPG 等 4 种算法的比较。

**表 4　DDPG 算法最终设定参数**

Tab. 4 Final setting parameters of DDPG algorithm

| 参数 | 设置 |
|---|---|
| 单回合训练步长 | 20 |
| Critic 学习率 | 0.001 |
| Actor 学习率 | 0.002 |
| 软更新参数 $\tau$ | 0.01 |
| 最大训练回合 | 1 000 |
| 经验缓冲区容量 | 10 000 |
| Mini 批大小 | 64 |
| 奖励折扣因子 | 0.6 |

图 5 中显示了在不使用状态归一化和行为噪声的情况下,DDPG 算法性能的差异。首先,可以明显发现,如果在训练策略中不使用状态归一化,训练算法将会因为状态空间内数值过大而失效。其次,如果在训练策略中没有添加行为噪声,将会导致算法收敛速度变慢以及收敛时奖励回报值更小。

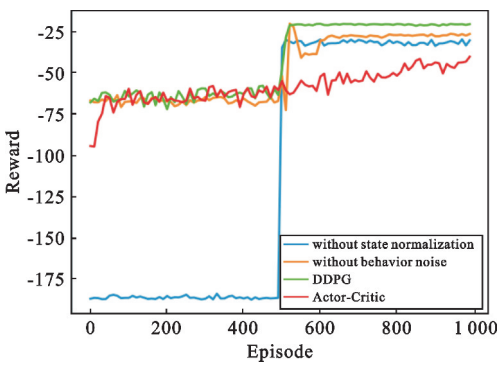


**图 5　不同算法下奖励回报曲线**

Fig. 5 Task hosted in partition of processing node

随着迭代次数的增加,不使用状态归一化和行为噪声的 DDPG 算法以及带有两者的 DDPG 算法都可以收敛,但 AC 算法却难以收敛。这是因为 AC 算法的 Actor 网络和 Critic 网络同时更新,而 Critic 网络本身难以收敛。相比之下,DDPG 受益于评价网络和目标网络的双网络结构,可以减少数据之间的相关性,从而找到最优策略。根据图 5,对于相同的 IMA 系统环境,本文提出的 DDPG 算法在跳出局部最优的同时亦有最好的收敛性能。

### 4.4　实验结果

使用本文提出的 DDPG 算法进行初始环境下的 IMA 系统任务分配,分别提取奖励最优值和收敛值情况下的分配方案与初始方案进行对比,表 5 记录了从迭代第 800 次开始处理节点分区内驻留的任务情况。

**表 5　处理节点分区内任务驻留情况**

Tab. 5 Task hosted in partition of processing node

| 方案 | $C_1$ | | $C_2$ | $C_3$ | | $C_4$ |
|---|---|---|---|---|---|---|
| | $P_1$ | $P_2$ | $P_1$ | $P_1$ | $P_2$ | $P_1$ |
| $S_0$ | $\text{Task}_{2,13}$ | $\text{Task}_{1,15}$ | $\text{Task}_{3,4,5,6}$ | $\text{Task}_{7,12}$ | $\text{Task}_{8,14}$ | $\text{Task}_{9,10,11}$ |
| $S_1$ | $\text{Task}_{2,13}$ | $\text{Task}_{1}$ | $\text{Task}_{3,4,5,6,15}$ | $\text{Task}_{7,12}$ | $\text{Task}_{8,14}$ | $\text{Task}_{9,10,11}$ |
| $S_2$ | $\text{Task}_{2,13}$ | $\text{Task}_{1}$ | $\text{Task}_{3,4,5,6,11,15}$ | $\text{Task}_{7,12}$ | $\text{Task}_{8,14}$ | $\text{Task}_{9,10}$ |
| $S_3$ | $\text{Task}_{2,13}$ | $\text{Task}_{3,4}$ | $\text{Task}_{1,5,6,15}$ | $\text{Task}_{7,12}$ | $\text{Task}_{8,14}$ | $\text{Task}_{9,10,11}$ |
| $S_4$ | $\text{Task}_{2,13}$ | $\text{Task}_{-1}$ | $\text{Task}_{3,4,5,6,15}$ | $\text{Task}_{1,7,12}$ | $\text{Task}_{8,14}$ | $\text{Task}_{9,10,11}$ |
| $S_5$ | $\text{Task}_{2,13}$ | $\text{Task}_{3,4,11}$ | $\text{Task}_{1,5,6,15}$ | $\text{Task}_{7,12}$ | $\text{Task}_{8,14}$ | $\text{Task}_{9,10}$ |
| $S_6$ | $\text{Task}_{2,13}$ | $\text{Task}_{1,3}$ | $\text{Task}_{4,5,6,9,11}$ | $\text{Task}_{7,12}$ | $\text{Task}_{8,14,15}$ | $\text{Task}_{10}$ |
| $S_7$ | $\text{Task}_{2,13}$ | $\text{Task}_{-1}$ | $\text{Task}_{3,4,5,6}$ | $\text{Task}_{1,7,12}$ | $\text{Task}_{8,14,15}$ | $\text{Task}_{9,10,11}$ |
| $S_8$ | $\text{Task}_{2,13}$ | $\text{Task}_{1,3}$ | $\text{Task}_{9,10,11,15}$ | $\text{Task}_{7,12}$ | $\text{Task}_{8,14}$ | $\text{Task}_{4,5,6}$ |
| $S_9$ | $\text{Task}_{2,13}$ | $\text{Task}_{1,3}$ | $\text{Task}_{9,10,11}$ | $\text{Task}_{7,12}$ | $\text{Task}_{8,14}$ | $\text{Task}_{4,5,6,15}$ |

表中 $S_1 \sim S_9$ 任务驻留情况都满足分配约束,同时因为任务分配的唯一性约束,在训练开始时有过渡任务分配情况 $S_{1,3,4,7}$。以上 4 种情况虽然对整体系统的任务处理完成时间没有优化,但是对于故障分区 $C_1 \sim P_2$ 的内存资源超出情况做出修正,优化了部分系统分区的负载性能,避免影响系统运行安全的情况发生,同时也是回合奖励最优值和收敛值分配方案的中间阶段。

奖励最优值下的任务分配方案,系统运行完成时间为 9.125 ms,与未使用算法优化的初始分配方案 $S_0$ 相比,效率提高了 22.67%。奖励收敛值下的任务分配方案,系统运行完成时间为 9.375 ms,与 $S_0$ 相比,效率提高了 20.55%。

## 5 结束语

本文以优化 IMA 系统任务分配为研究对象,建立 IMA 系统框架,并通过 DDPG 算法对框架进行训练求解。在 DDPG 算法的基础上引入了行为噪声、状态标准化等策略训练技巧,有助于增加对于动作空间的探索以及防止出现状态空间数据过大无法计算的情况。完成了仿真实验分析,通过 DDPG 算法对 IMA 系统任务分配进行优化,得出的任务分配方案系统运行效率比初始分配方案提高了 20. 55%。同时与传统算法相比,收敛性能更好,效率更高。

DDPG 算法涉及 Actor-Critic 网络,网络中参数的更新前后存在相关性,导致神经网络只能片面地看待问题。后续可以深入考虑网络参数更新机制,获得更高效的任务分配方案。

## 参考文献:


[1] 褚文奎,张凤鸣,樊晓光. 综合模块化航空电子系统软件体系结构综述[J]. 航空学报,2009,30(10):1912-1917.

[2] 赵长啸,刘嘉琛,王鹏. 综合化航电系统驻留功能组合失效分析方法[J]. 电讯技术,2020,60(10):1175-1180.

[3] 丁全心. 综合模块化航空电子系统标准述评[J]. 电光与控制,2013,20(6):1-3.

[4] ANNIGHOEFER B,NIL C,SEBALD J,et al. Structured and symmetric IMA architecture optimization:use case Ariane launcher[C]//Proceedings of 2015 IEEE/AIAA 34th Digital Avionics Systems Conference. Prague:IEEE,2015:1-14.

[5] AMINIFAR A, ELES P, PENG Z. A scheduling algorithm to stabilize control applications [C]// Proceedings of the 21st IEEE Real-Time and Embedded Technology and Applications Symposium. Seattle:IEEE, 2015:63-72.

[6] 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.

[7] KURDI M. An effective new island model genetic algorithm for job shop scheduling problem [J]. Computers and Operations Research,2016,67:132-142.

[8] HOU C J,SHIN K G. Allocation of periodic task modules with precedence and deadline constraints in distributed real-time systems[J]. IEEE Transactions on Computers, 1997,46(12):1338-1356.

[9] 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.

[10] 高泽海,马存宝,牛浩田. 综合模块化航电系统性能退化 Lévy 模型[J]. 系统工程与电子技术,2021,43(4):1144-1152.

[11] 徐显亮,张凤鸣,褚文奎. 一种以安全性为中心的 IMA 软件体系结构设计方法[J]. 计算机科学,2012(3):128-130.

[12] YAO J, GE Z. Path-tracking control strategy of unmanned vehicle based on DDPG algorithm [J]. Sensors,2022,22(20):7881-7892.

[13] 梁燕,惠莹. 基于竞争双深度 Q 网络的动态频谱接入[J]. 电讯技术,2022,62(12):1715-1721.

[14] SILVER D, LEVER G, HEESS N, et al. Deterministic policy gradient algorithms [C]//Proceedings of 2014 IEEE International Conference on Machine Learning. Beijing:IEEE,2014:387-395.

[15] ZHANG Z, QIU C, ZHANG D, et al. A coordinated control method for hybrid energy storage system in microgrid based on deep reinforcement learning [J]. Power System Technology,2019,43(6):1914-1921.

[16] QIAO J, WANF X, ZHANG Q. Optimal dispatch of integrated electricity-gas system with soft actor-critic deep reinforcement learning [J]. Proceedings of the CSEE,2021,41(3):819-833.

[17] YANG J,PENG W,SUN C. A learning control method of automated vehicle platoon at straight path with DDPG-based PID[J]. Electronics,2021,10(21):2580-2591.

[18] ZARROUKI B, KLÖS V, HEPPNER N, et al. Weights-varying MPC for autonomous vehicle guidance: a deep reinforcement learning approach[C]//Proceedings of 2021 European Control Conference. Delft:IEEE,2021:119-125.

[19] WOO J,YU C,KIM N. Deep reinforcement learning-based controller for path following of an unmanned surface vehicle [J]. Ocean Engineering,2019,183:155-166.

[20] HE K,DONG C,YAN A,et al. Composite deep learning control for autonomous bicycles by using deep deterministic policy gradient [C]//Proceedings of the 46th Annual Conference of the IEEE Industrial Electronics Society. Singapore:IEEE,2020:2766-2773.

[21] WANG Y, FANG W, DING Y, et al. Computation offloading optimization for UAV-assisted mobile edge computing:a deep deterministic policy gradient approach [J]. Wireless Networks,2021,27(4):2991-3006.


## 作者简介:


**赵长啸** 男,1989 年生于山东临清,2013 年于北京航空航天大学获博士学位,现为副教授,主要从事民机航电系统设计与评估、航空电子综合技术研究。

**李道俊** 男,2000 年生于福建南平,硕士研究生,主要研究方向为民机航电系统多分区任务分配。

**汪鹏辉** 男,1998 年生于甘肃嘉峪关,硕士研究生,主要研究方向为航空器适航审定工程。

**田 毅** 男,1983 年生于陕西汉中,2009 年获硕士学位,现为副研究员,主要从事机载电子硬件适航审定、航空专用集成电路设计、计算机体系结构研究工作。