基于深度强化学习的软件定义安全中台 QoS 实时优化算法

李元诚，秦永泰
（华北电力大学控制与计算机工程学院，北京 102206）

摘 要：针对软件定义安全场景中的服务质量（QoS）实时优化方案因安全防护手段与业务场景不匹配而导致的适用困难和性能下降的问题，提出了基于深度强化学习的软件定义安全中台 QoS 实时优化算法。首先，将碎片化的安全需求与安全基础设施统一到软件定义安全中台云模型中；然后，通过深度强化学习结合云计算技术提高安全中台的实时匹配和动态适应能力；最后，生成满足 QoS 目标的安全中台资源实时调度策略。实验结果表明，与现有实时算法相比，所提算法不但保证负载均衡，还提高了 18.7%的作业调度成功率以提高服务质量，降低了 34.2%的平均响应时间，具有很好的稳健性，更适用于实时环境。

关键词：软件定义安全；深度强化学习；安全中台；服务质量
中图分类号：TP393
文献标志码：A
DOI: 10.11959/j.issn.1000−436x.2023090

Deep reinforcement learning based algorithm for real-time QoS optimization of software-defined security middle platform

LI Yuancheng, QIN Yongtai
School of Control and Computer Engineering, North China Electric Power University, Beijing 102206, China

Abstract: To overcome the problem that the real-time optimization of the quality of service (QoS) in software-defined security scenarios was hindered by the mismatch between security protection measures and business scenarios, which led to difficulties in application and performance degradation., a novel algorithm based on deep reinforcement learning for optimizing QoS in software defined security middle platforms (SDSmp) in real-time was proposed. Firstly, the fragmented security requirements and infrastructure were integrated into the SDSmp cloud model. Then by leveraging the power of deep reinforcement learning and cloud computing technology, the real-time matching and dynamic adaptation capabilities of the security middle platform were enhanced. Finally, a real-time scheduling strategy for security middle platform resources that meet QoS goals was generated. Experimental results demonstrate that compared to existing real-time methods, the proposed algorithm not only ensures load balancing but also improves job success rate by 18.7% for high QoS and reduces the average response time by 34.2%, and it is highly robust and better suited for real-time environments than existing methods.

Keywords: software defined security, deep reinforcement learning, security middle platform, quality of service

0 引言

近年来，信息接入终端设备种类不断丰富。物联网、边缘计算、机器学习技术飞速发展，互联网与人类生活日益密切，数据面临的风险更加复杂多元，安全业务的碎片化越来越严重。碎片化的安全需求与安全场景是网络安全所面临的巨大挑战之一[1]，碎片化难题也使安全防护手段与业务场景不匹配的矛盾日益凸显[2]，安全产品的服务质量（QoS, quality of service）和实时响应能力越

收稿日期：2022−11−01；**修回日期：**2023−02−04
基金项目：国网江西信息通信公司基金资助项目（No.52183520007V）
Foundation Item: The State Grid Jiangxi Information & Telecommunication Company Project (No.52183520007V)

来越受到重视。

《关键信息基础设施安全保护条例》指出了关键信息基础设施所面临的安全挑战及重点防护要求，传统城墙式防守不足以应对安全挑战，需要构建以安全中台为核心的，积极、主动、弹性、快速响应的安全防御体系，实现从安全监测、全局态势、能力调度到编排响应的防护理念。

受到软件定义安全、安全中台的启发，文献[3]构建了面向全场景的软件定义安全中台（SDSmp, software defined security middle platform）架构，如图 1 所示，目的是解决安全资源利用率低、复用难，安全需求与安全场景高度碎片化等问题。SDSmp 为解决安全防护手段与业务场景不匹配问题提供了有效着力点。

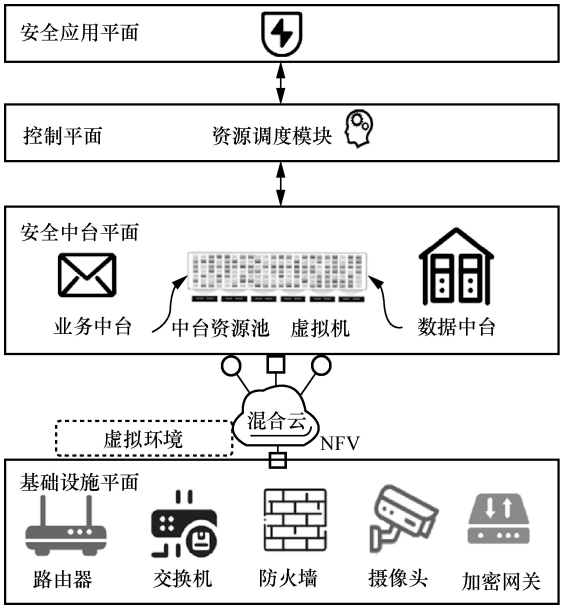

图 1 软件定义安全中台架构

软件定义安全中台架构通过网络功能虚拟化（NFV, network functions virtualization）技术和云计算技术实现基础设施平面的虚拟化[3]。安全中台[1-2]平面兼具数据中台和业务中台的优点，能够消灭数据孤岛、提高资源复用率、降低开发难度和成本等。解决安全防护手段与业务场景不匹配问题，关键在于控制平面的资源调度模块。

控制平面资源调度模块根据安全应用平面传来的安全业务的特点，分析所需的计算能力，由南向应用程序接口（API, application programming interface）通过中台资源池将其分配给可用的安全中台资源执行，物理上最终在基础设施平面执行。

安全中台平面为了平衡资源复用率、负载均衡和用户的 QoS，需要采用恰当的调度策略，高效地利用中台资源池来提高 QoS。目前，安全中台资源调度的研究很少，特别是对于实时情况，考虑到人们对于 QoS 的要求越来越高，提高 QoS 对于拥有大量服务器、每天收到大量用户请求的产品来说至关重要[4]。为了解决上述问题，本文提出了一种基于深度强化学习（DRL, deep reinforcement learning）的软件定义安全中台 QoS 实时优化算法，提供了算法的详细设计和实现过程，并对不同类型作业负载场景的大量模拟实验进行了广泛的性能评估。本文主要贡献如下。

1) 架构层面，提出面向碎片化安全需求和安全场景的 SDSmp 自动控制框架，实现对安全中台资源的在线实时调度和自动化控制。

2) 建模层面，通过建立软件定义安全中台 QoS 优化模型，结合云计算技术和深度强化学习算法，使控制平面的调度器能够根据经验在线学习如何合理地选择安全中台资源，从而提高服务质量，缩短响应时间，实现负载均衡。

3) 实现层面，搭建了软件定义安全中台实验环境，将所提基于深度强化学习的软件定义安全中台 QoS 实时优化算法，在不同的工作负载模式下与现有实时作业调度算法进行比较。实验结果表明，所提算法在平均响应时间和作业调度成功率方面普遍优于现有实时算法。

1 相关工作

软件定义领域的资源调度主流研究集中在软件定义安全（SDSec, software defined security）和软件定义网络（SDN, software defined network）。文献[5]提出了一种 SDSec 架构开放安全设备的方法，提出使用前向传播（BP）神经网络来预测安全任务的执行时间的安全资源调度算法。文献[6]提出了一种基于软件定义安全的资源调度机制，设计了南向 API，提出了安全资源抽象和负载均衡调度算法。文献[7]研究了云计算中软件定义网络的资源分配机制。文献[8]提出一种动态调度算法，以最大限度地提高每次切换过程中的安全性，同时考虑切换成本和时延。文献[9]提出了在云中基于软件定义安全架构的安全解决方案，设计实现了虚拟安全设备管理器来管理资源池中多种虚拟安全设备。文献[10]在 SDN 场景中针对参数与场景不匹配的问题，提出了基于

DRL 的 QoS 优化算法。文献[11]针对 SDN 中转发验证机制，优化通信与计算开销的问题。上述文献都对 SDSec 和 SDN 进行了不同层面的资源调度算法研究，但是并没有针对碎片化的安全需求与安全场景，解决安全防护手段与业务场景不匹配问题。

深度强化学习具有较高的准确性，已经解决了很多困难的决策问题[12-14]，例如，用于云计算中的价格优化[15]。DRL 具有深度神经网络（DNN）的优势，适用于具有高维状态空间和低维行动空间的复杂控制问题[16]。该技术已经证明了它在决策方面的强大能力，仅需要提前很短时间训练模型，就可以解决各种优化问题[17]。文献[4]提出对于变化的负载和复杂的决策情况，基于 DRL 的算法能在云端作业调度中表现出良好的性能，此外，DRL 已被部分用于解决云计算的资源调度问题。

上述研究都取得了良好的效果，但它们并不是专为软件定义安全设计的。此外，安全虚拟化技术仍处于初级阶段，该领域目前主流的资源调度算法是在保持最后期限约束的情况下，对批量作业进行调度，而所提算法融合 DRL、SDSmp、云计算、安全中台等技术，在解决安全防护手段与业务场景不匹配问题的基础上，实现软件定义安全场景中的实时 QoS 优化。

2　软件定义安全中台 QoS 优化架构

为了解决安全防护手段与业务场景不匹配问题，本文从图 1 所示的 SDSmp 出发，设计了基于深度强化学习的软件定义安全中台 QoS 优化架构。如图 2 所示，优化架构由用户、安全应用平面、控制平面、安全中台平面、基础设施平面组成。控制平面的北向为安全应用平面，南向为安全中台平面。控制平面中对于调度起关键作用的是资源调度模块的 DRL 调度器，其他关键部分如应用管理模块、信息收集器包括资源监视器和作业监视器，用于收集中台资源池中的安全中台资源和前台作业信息。

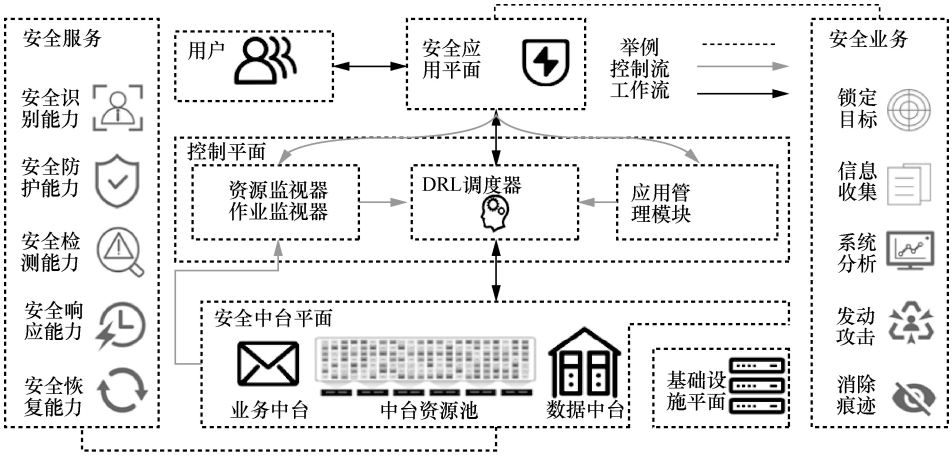

图 2　基于深度强化学习的软件定义安全中台 QoS 优化架构

软件定义控制平面与基础设施平面分离的关键在于控制平面对虚拟化的中台资源池中的资源统一调度，以及安全中台、大中台小前台的宗旨[18]。碎片化的安全需求与安全场景都对资源调度提出了更高的要求，不同的调度算法对软件定义安全中台性能影响巨大。

软件定义安全场景中，安全中台的主旨是资源可复用、能力服务化。以一次网络安全防御为例，安全中台将能力抽象为逐条服务，服务由相应能力的安全中台资源提供，安全应用的请求先转化为对应的逐类安全业务；然后，不同的安全业务用作业请求的形式向中台资源池请求所有需要的服务。日常使用中，用户在安全应用平面通过使用终端安全产品，以连续作业请求的方式提交用户需求。控制平面通过对用户的请求解析，分析作业请求，如资源利用率、计算能力、内存、所需的响应时间、QoS 等。安全中台资源按中台结构封装成虚拟机（VM, virtual machine），是逻辑上的执行单位，物理上的实际执行单位是具体的基础设施层安全设备，基础设施平面通过 NFV 技术和云计算技术，按功能映射到不同的虚拟机集群，实现逻辑设备无关。

当一个作业到达时，控制平面的 DRL 调度器会在中台资源池中寻找合适的封装好的安全中台资源虚拟机来分配作业，并执行请求的作业。因此，

作业调度器是核心模块，它在特定的时间间隔内根据用户提交的 QoS 要求做出决策。对应 DRL 的运作机制中，作业调度器采取将前台作业分配给特定的安全中台资源池中某一具体虚拟机的行动，根据这一行动，由环境提供奖励并更新状态，迭代实现调度器的智能学习。在这个过程中，资源和作业监视器负责管理作业队列的工作量和性能，以及作业的执行和分配。

为了对优化问题进行建模，本文给出了负载和安全中台资源定义，以及作业调度机制，参数含义如表 1 所示。

表 1 参数含义

参数	含义
J^{id}	安全业务前台作业的 ID
J^{at}	安全业务前台作业到达时间
J^{t}	安全业务前台作业类型（计算或 I/O 密集型）
J^{l}	安全业务前台作业长度（所需的指令、服务）
J^{q}	安全业务前台作业 QoS 要求
J^{rt}	安全业务前台作业响应时间
J^{et}	安全业务前台作业执行时间
J^{wt}	安全业务前台作业等待时间
V^{id}	安全中台资源（VM）的 ID
V^{t}	安全中台资源（VM）类型（计算或 I/O 密集型）
V^{p}	处理速度（每秒处理的指令、服务）
$V^{\mathrm{p}}_{\mathrm{com}}$	安全中台资源（VM）的计算处理速度
$V^{\mathrm{p}}_{\mathrm{io}}$	安全中台资源（VM）的读写处理速度
V^{it}	安全中台资源（VM）的空闲时间
R	奖励（体现 QoS、作业调度成功率、响应时间等）
Suc	作业调度成功率（作业是否调度成功满足 QoS）

2.1 负载定义

不同的安全应用请求首先在安全应用平面完成了并行分类和细化，转化为安全业务，安全业务提交的请求是高度解耦、低相关性、细粒度的简单作业，在调度过程中被分配给同样细粒度的安全中台资源，它们以提供服务的形式完成每个作业的执行，最后统一组装，提高了并行性，很大程度上避免了因为传统作业间逻辑依赖、前驱后继关系和资源抢占带来的问题。

假设在实时场景中的作业是独立的，在执行过程中没有其他作业相互干扰。为了解决该场景中状态空间维度过大导致的 DRL 调度器待选择动作太多的问题，引入一个面向事件的决策机制，在前台作业到达控制平面后立即对作业进行实时分析。这些作业信息被用来训练作业调度机制。对于提出的模型，本文考虑了 2 种典型的作业类型，即计算密集型作业和 I/O 密集型作业。前台安全业务传来的作业 i 建模为

$$\boldsymbol{J}_i=\{J_i^{\mathrm{id}},J_i^{\mathrm{at}},J_i^{\mathrm{t}},J_i^{\mathrm{l}},J_i^{\mathrm{q}}\} \tag{1}$$

2.2 安全中台资源定义

在作业的调度运行中，由于前台用户提交的作业可能属于不同的类型，它们在不同类型的安全中台虚拟机上有不同的响应时间。与作业负载类似，考虑 2 种安全中台资源，即 I/O 密集型虚拟机 VM_{t1} 连接基础设施层的最终执行资源（如监控器），计算密集型虚拟机 VM_{t2} 连接基础设施层的最终执行资源（如数据加密解密模块）。每个安全中台资源定义为

$$\boldsymbol{V}_j=\{V_j^{\mathrm{id}},V_j^{\mathrm{t}},V_{\mathrm{com}_j}^{\mathrm{p}},V_{\mathrm{io}_j}^{\mathrm{p}}\} \tag{2}$$

2.3 作业调度机制

调度决策后，当一个作业被分配给一个特定的安全中台 VM 实例时，该作业首先进入一个等待队列 L_j^i。在不失一般性的前提下，假设每个虚拟机实例在任何时候都只能独占式执行其等待队列中的一个作业。作业调度器是核心组件，负责根据最终用户的要求将作业分配给合适的中台资源池中的资源。如果等待队列为空，被分配的作业会顺利通过队列到达虚拟机，并被立即执行；否则先进入等待状态。根据上述假设，作业的响应时间将由作业执行时间 J_i^{et} 和作业等待时间 J_i^{wt} 两部分组成，响应时间可以表示为

$$J_i^{\mathrm{rt}}=J_i^{\mathrm{et}}+J_i^{\mathrm{wt}} \tag{3}$$

作业执行时间会因为调度到不同的安全中台资源而不同，对于某个固定类型的前台作业，由于每个安全中台资源在实际运行中作业的各部分都是并行的，影响作业在中台资源上执行时间的主要因素是该作业类型对应的长度，其他类型的长度相较之下很短，并在运行过程中不会产生实际的影响。因此，作业执行时间定义为

$$J_i^{\mathrm{et}}=\max\left(\frac{J_{\mathrm{com}_i}^{\mathrm{l}}}{V_{\mathrm{com}_j}^{\mathrm{p}}},\frac{J_{\mathrm{io}_i}^{\mathrm{l}}}{V_{\mathrm{io}_j}^{\mathrm{p}}}\right) \tag{4}$$

其中，$J_{\mathrm{com}_i}^{\mathrm{l}}$ 是作业所需计算长度，$J_{\mathrm{io}_i}^{\mathrm{l}}$ 是作业所需

I/O 长度，$V_{\mathrm{com}_j}^{\mathrm{p}}$ 是安全中台资源计算处理速度，$V_{\mathrm{io}_j}^{\mathrm{p}}$ 是安全中台资源读写处理速度。可以看到，对应长度的作业类型是主要影响因素，但是作业有可能会被调度到合适或者不同类型的中台资源，类似于木桶效应。如果作业类型与资源类型匹配，安全中台资源对应类型的性能好，则作业执行时间短；如果不匹配，由于中台资源对应类型性能差，作业执行时间会长得多。另外，作业等待时间会影响资源调度，等待时间定义如下

$$J_i^{\mathrm{wt}}=\begin{cases}0, L_j^i=0\\ \sum_{n=0}^{i} J_n^{\mathrm{et}}, \text{其他}\end{cases} \tag{5}$$

如果等待队列为空，作业立即执行，否则需要先等待，等待时间是所有已到达的作业执行时间的累加。当前台作业 $\boldsymbol{J}_i$ 被调度到资源 $\boldsymbol{V}_j$，并完成处理后，安全中台资源的空闲时间更新如下

$$V_j^{\mathrm{it}}=J_i^{\mathrm{wt}}+J_i^{\mathrm{at}}+J_i^{\mathrm{et}} \tag{6}$$

其中，J_i^{at} 是作业到达时间

2.4　QoS 感知的调度成功条件

安全中台资源以服务的形式为软件定义安全中台赋能，满足 QoS 要求意味着安全中台资源成功给请求服务的安全业务提供安全防护能力。软件定义安全中台允许终端用户在提交前台作业请求时指定 QoS 的需求，安全业务往往有着严格的最晚响应时间的要求，实时环境中更是如此。

实时响应要求高的云计算领域广泛采用作业调度成功率指标来衡量实时环境中的 QoS，文献[4,19]明确给出式(7)所示的成功率。

$$\mathrm{Suc}_{ij}=\begin{cases}1, J_i^{\mathrm{rt}}\leqslant J_i^{\mathrm{q}}\\ 0, J_i^{\mathrm{rt}}>J_i^{\mathrm{q}}\end{cases} \tag{7}$$

文献[15,20-22]也指出响应时间低于 QoS 要求即认为此次调度成功。与其他研究保持一致，前台作业的 QoS 要求定义为 J_i^{q}，代表前台作业可接受的最大响应时间，超过此期限可能会导致安全业务失效。

安全业务传来的每个前台作业都有一个执行期限（预期）。如果安全中台资源的执行结果能够在最后期限内返回，则本次调度执行成功，QoS 要求得到满足；否则，本次调度执行失败。

3　算法设计

为了解决目前主流的防护方案因为安全防护手段与业务场景的碎片化和不匹配导致适用困难和性能下降的问题，如传统的控制论调度算法和基于启发式的调度算法均难以适用，本文提出基于深度强化学习的软件定义安全中台 QoS 实时优化算法。以自动化实时 QoS 感知的方式提高安全业务作业调度成功率，生成满足 QoS 要求的安全中台资源实时调度策略。安全中台平面还能使基础设施平面具有更高的负载均衡和更低的成本。此外，模型的训练阶段离线进行，运行决策阶段在线进行，既不占用安全中台资源，也能更好地适应多变的安全场景。

深度 Q 学习（DQN, deep q-learning）是一种无模型的强化学习（RL）[12]算法，代理几乎不需要人为输入先验的知识。强化学习模型包括环境、代理、行动、状态、奖励，奖励函数为 $Q:S\times A\Rightarrow R$，目的是预测最大化奖励的行动，奖励函数是回报函数 Return 的基础。代理通过试错互动做出决定，每执行一个行动后，环境会移动到下一个新的状态 $\mathbb{S}_{t+1}$。同时，代理将获得奖励 $\mathbb{R}_t$，实验重放机制是连续的[4]。回报函数表示为

$$\mathrm{Return}=\sum_{t=0}^{n} R_t\gamma^t \tag{8}$$

其中，γ 是一个加权未来奖励的系数，用来指导模型更侧重于当下还是未来可能的奖励；Return 是从开始到结束所有 R 的加权累加。训练最常用的损失是均方误差（MSE）损失，可表示为

$$\min\sum_{i=1}^{|B|}(R_{t_i}+\gamma\max_{A'}Q'_{\theta^{Q'}}(S_{t+1},A')-Q_{\theta^{Q}}(S_t,A_{t_i}))^2 \tag{9}$$

其中，$|B|$ 是迷你经验池规模，评估网络参数 $\theta^{Q'}$ 在计算 MSE 损失时是固定的；R_{t_i} 是在状态 S_t 下采取行动获得的奖励；γ 是折扣系数，$\gamma\in(0,1]$。代理利用 DNN 产生的奖励回馈环境，在具体的状态上做决策，所有状态–行动对相关。

如图 3 所示，软件定义安全中台控制平面中，代理根据前台到达的安全业务需求的类型和奖励函数的权值 $W_1\sim W_4$，采取智能的调度决策将不同类型的作业分配给中台资源池中类型最合适的封装的中台虚拟化资源，代理获得复合奖励，DNN 更新参数，中台资源池更新状态。

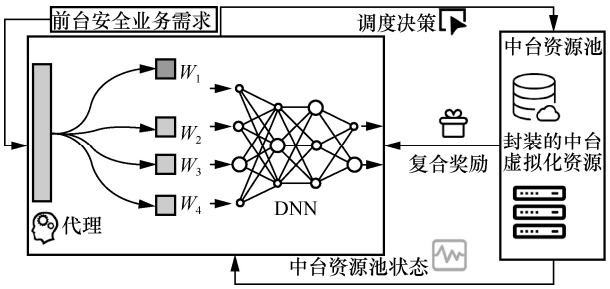

图 3 基于深度强化学习的软件定义安全中台 QoS 实时优化算法架构

在安全中台环境中，传入作业负载的属性和类型是无法预测的。对于这样的场景，基于 RL 的模型表现出色，因为它们仅需要极少人为输入的先验经验，如状态转移和其他系统信息。在每次决策迭代中，RL 代理观察当前的环境状态，然后使用 DNN 来估计所有中台资源池中可用的中台资源的 Q 值，从而产生自我训练的数据，提高未来的决策能力。

根据智能体生成的策略 $\pi(A|S)$，代理将选择一个中台资源池中的实例来执行作业并获得奖励。由于状态空间很大，训练 DNN 的时间也可能很长。为了避免这种情况，本文算法使用面向事件的决策机制。当新的作业到达时，代理将做出在线决策。所有作业都遵循先来先服务的规则。作业到达后，所需的行动是在软件定义安全中台资源池中分配该作业。得益于在线决策机制，同样减少了可选行动的数量。所提算法分为决策阶段和训练阶段。

决策阶段。DRL 算法采用 DQN 技术，将作业分配给适当的安全中台资源 VM 实例。根据特定的要求做决策，代理获得相应的奖励。代理检查更新当前的环境状态做出下一个可能的决定。以下是强化学习模型的重要组成部分。

动作空间 A 是代理在特定环境中可以采取的所有行动的集合[23]。动作空间可以表示为所有中台资源池中的安全中台资源虚拟机实例总数的集合，包含的行动为将前台安全业务分配给安全中台资源池，长度可以表示为所有可用安全中台资源的数量[4]。每个虚拟机都有自己的队列来容纳传入的作业请求。对传入的作业请求没有长度限制。

$$A=\left\{A_{t_i}\middle|A_{t_i}\in \mathrm{VM}_{t1}\cup \mathrm{VM}_{t2}\right\} \tag{10}$$

其中，VM_{t1} 和 VM_{t2} 指不同类型的安全中台资源，给不同类型的前台作业提供安全服务，例如，设置 VM_{t1} 为计算密集型安全中台资源，VM_{t2} 为 I/O 密集型安全中台资源。动作 A_{t_i} 为作业 i 选择合适类型的某一个具体的安全中台资源，因为选择有限所以维度通常不高。

状态空间 S 是一个由代理可以根据行动更新的所有状态组成的集合，这些行动会产生有限的状态空间[23]。对于软件定义安全中台，一个新的前台安全业务提交的作业 i 在时间 t 到达，此刻的状态空间可以用安全资源的总状态和作业的当前状态描述。

$$S=\left\{S_t\middle|S_t\in S_{\mathrm{job}}\cup S_{\mathrm{VM}}\right\} \tag{11}$$

其中，S_{VM} 是作业 i 在时间 t 到达时的所有安全中台资源的状态，S_{job} 是当前需要被调度作业的状态，S_t 是当前所有待调度作业的状态与此时安全中台资源状态的集合。然而，由于在不同时间有无数种选择的可能，状态空间维度通常很高，本文采用面向事件的决策机制解决此问题。

行动选择和状态转移。本文模型考虑当前状态和 DNN 网络 Q 值中预测的未来状态采取行动。训练初，模型在安全中台资源虚拟机上用概率 ϵ 随机分配作业；随着算法的学习，ϵ 不断变化。代理随机分配作业，用贪婪的策略探索几种可能性。这里将选择最高的预测 Q 值。随着作业的分配，状态将从 S_t 转移到 S_{t+1}。

奖励函数 R_{t_i}。在当前状态 S_t 下采取行动后，系统更新到状态 S_{t+1}，并从环境中获得奖励 R_{t_i}。每次迭代中，环境都会给予奖励。奖励正负取决于行动的情况，代理能通过行动获得不同的奖励，奖励函数引导代理为作业调度框架的目标做出智能决策。在本文模型中，作业调度的高 QoS 是主要优化目标，使作业调度成功率最大化。此外，对于满足 QoS 要求的每个作业，响应时间越短，服务质量就越好。基于此，定义一个作业的奖励如下

$$R_{t_i}=\begin{cases}\dfrac{J_i^{\mathrm{l}}}{J_i^{\mathrm{rt}}V_j^{\mathrm{p}}}, J_i^{\mathrm{rt}}\leqslant J_i^{\mathrm{q}}\\ 0, J_i^{\mathrm{rt}}>J_i^{\mathrm{q}}\end{cases} \tag{12}$$

其中，J_i^{rt} 是作业响应时间，J_i^{q} 是 QoS 要求，J_i^{l} 是作业长度，V_j^{p} 是中台资源执行速度，当且仅当作业的响应时间低于 QoS 要求时，此次调度满足 QoS 要求，调度成功，否则失败。研究重点是软件定义

安全领域中，QoS 实时安全中台资源调度策略，解决因安全防护手段与业务场景不匹配造成适用困难和性能下降的问题。

训练阶段。为了从经验中学习，DRL 将当前状态、行动、奖励和下一状态的过渡值存储在容量为 N_Δ 的重放存储器 Δ 中。DNN 的参数 θ 将在 Q-learning 更新时使用 S_Δ 进行更新，为避免时间复杂度过大设置决策集 U，且 $U \geqslant 1$。经验重放机制从随机样本中学习，减少数据相关性，减少 θ 的方差。使用目标网络生成 Q 值，采用目标网络和评估网络消除 DNN 的分歧和振荡，目标网络和评价网络结构相同，但参数不同[14]。

训练的复杂性分析和开销与其他领域的深度学习应用一样，所提算法的训练过程是离线进行的，这样可以最大限度地节省成本，避免占用宝贵的安全中台资源。模型被训练出来以后，就可以进行实时调度，后续正常运行中不需要再次离线训练。具体来说，模型的隐藏层使用了 20 个神经元，当模型不大时，开销接近于 0，而调度时间总小于 10 ms，这实际上是可以忽略的。

算法 1　基于 DRL 算法的训练过程

输入　初始值 ϵ，α，γ，学习率 f，开始学习时间 τ，迷你经验池 S_Δ，重放时间 η

输出　最大 Return 最小损失 $L(\theta)$ 的生成调度策略 $\pi(A|S)$

1) 随机初始化行动–价值评价网络 $Q(S,A|\theta^Q)$ 和行动者 $\mu(S|\theta^\mu)$，权值为 θ^Q 和 θ^μ

2) 初始化目标网络 $Q'(S,A'|\theta^{Q'})$ 和 μ'，权值为 $\theta^{Q'}$ 和 $\theta^{\mu'}$

3) 初始化容量为 N_Δ 的记忆重放 Δ

4) for 每份在 t 时间到达的新作业 i do

5)　确定当前状态 S_t

6)　for episode $= 0,1,\cdots,M$ do

7)　以概率 ϵ 随机选择一个行动 A_t；否则 $A_t = \text{argmax}_A Q(S_t,A|\theta^Q)$

8）　根据行动 A_t 调度作业 i，并将前台作业 i 添加到中台资源等待队列 L_j^i 中

9)　计算动作 A_t 获得的奖励函数 R_t

10)　在下一个决策时刻 t_{i+1} 状态更新到 S_{t+1}

11)　把转移参数 $(S_t,A_{i+1},R_{t+1},S_{t+1})$ 存储到 Δ

12)　if $i \geqslant t$ 并且 $i \equiv 0 \bmod f$

13)　if $i \equiv 0 \bmod \eta$

14)　重置 $Q' = Q$

15)　end if

16)　从 N_Δ 随机抽取样本 S_Δ

17)　for S_Δ 中的每次转移 $(S_t,A_{i+1},R_{t+1},S_{t+1})$

18)　计算 $\text{target}_t = R_t + \gamma \max_{A'} Q'(S_{t+1},A'|\theta^{Q'})$

19)　通过最小化损失函数更新评价网络 $L(\theta) = \frac{1}{B}\sum_t \left(\text{target}_t - Q(S_t,A_i|\ \theta^Q)^2\right)$

20)　通过梯度下降更新策略 $\frac{\partial L(\theta)}{\partial \theta} = L(\theta)\frac{\partial Q\left(S_t,A_i|\ \theta^Q\right)}{\partial \theta}$

21)　end for

22)　ϵ 逐渐减少直到下限

23)　end if

24)　end for

25) end for

4　仿真评估

本节通过一系列实验来评估提出的基于深度强化学习的软件定义安全中台 QoS 实时优化算法性能，并与常见的 5 种在线作业调度算法进行比较。首先，对实验进行了合理的设置和必要的简化，保证实验顺利进行并具有说服力。然后，说明了建议的模型和对比算法中的参数，对 5 种对照算法、各个参数说明介绍，设置 3 种不同的工作负载模式来模拟真实情况，进行充分的仿真实验验证本文算法能适应不同类型的环境。为了进一步展示算法性能，本节还进行了更长时间窗口的实验。实验硬件软件配置为 Python3、TensorFlow，使用 2.7 GHz 英特尔酷睿 i5 处理器和 16 GB RAM 的机器。

4.1　实验设置

4.1.1　实验模拟环境

考虑安全中台平面上已经被池化虚拟化的中台资源池，对于控制平面的资源池管理模块，显示为统一调用的不同类型不同性能的 API。为了简化实验，将中台资源池的安全中台资源 VM 设置为计算密集型和 I/O 密集型，应用平面通过应用管理模块传入控制平面的作业是连续的计算密集型和 I/O 密集型。

控制平面将北向的应用平面传来的作业调度到安全中台平面执行。安全业务作业如果被调度到同种类型的安全中台资源，则执行速度快；如果执

行不同种类的作业类型，则执行速度慢。安全中台资源 VM 平均处理能力如表 2 所示。

表 2 安全中台资源 VM 平均处理能力

安全中台资源	计算型作业		I/O 型作业	
	平均值/MIPS	标准差/MIPS	平均值/MIPS	标准差/MIPS
计算密集型资源	1 000	100	500	50
I/O 密集型资源	500	50	1 000	100

实验中，默认情况下，作业长度由平均值 100 MIPS 和标准差 20 MIPS 的正态分布生成，MIPS 表示百万条指令每秒。每个作业的 QoS 要求（即可接受的最大响应时间）在 250 ms 和 350 ms 之间均匀随机生成。新到达作业类型在计算密集型和 I/O 密集型之间均匀地随机选择。作业到达率和作业类型的概率分布每 5 s 为一个周期更新。对于每个模拟工作负载模式，实验都随机生成 20 个安全中台资源虚拟机实例，并追踪每个安全资源从开始运行到结束的整个过程，一共持续 300 s。

4.1.2 模型参数

基于深度强化学习的软件定义安全中台 QoS 实时优化算法使用前馈神经网络构建底层 DNN，该网络的全连接隐藏层具有 20 个神经元，设置记忆重放 N_{Δ}=1 000 的容量，迷你经验池 S_{Δ}=40。采用 AdamOptimizer 算法对评价网络参数进行更新，学习速率为 0.01。每 50 个决策集从评估网络克隆一次参数给目标网络。在记忆重放中累积了足够的过渡样本后，DNN 开始进行训练。设置 τ=500，f=0.1，γ=0.9，每轮学习迭代中 ϵ 从 0.9 降低到 0.002。

4.1.3 对比算法和评价指标

为了评估提出的基于深度强化学习的软件定义安全中台 QoS 实时优化算法（后文简称 DQN）性能，把它与 5 种常见算法进行对比，分别为随机调度算法、循环调度算法、最早调度算法、最佳拟合调度算法[24]和合理的调度算法[25]。

常见的控制论调度算法中，随机调度算法（后文简称 random）是一种非常简单的算法，它为每个作业选择一个随机的 VM 实例。循环调度算法（后文简称 round-robin）主要侧重于如何公平地将作业调度到 VM 实例。因此，VM 实例按循环顺序选择以执行传入作业。最早调度算法（后文简称 earliest）是一种先来先服务的策略，其中新到达的作业调度到最早的空闲 VM 实例。

最佳拟合调度算法[24]（后文简称 suitable）是一种贪心算法，尽量做出对于当下来说最好的选择。与最早调度算法相比，最佳拟合调度算法考虑 2 个因素，即时间因素以及所选 VM 实例的类型是否与新到达作业的类型匹配。该算法总是通过寻找局部最优解而不是整体最优解，将作业分配给类型适合的 VM 实例，来减少执行时间。也就是说，最佳拟合调度算法将新到达的作业分配给类型合适的所有 VM 实例中最先空闲的资源。

合理的调度算法（后文简称 sensibleR）[25]是一种自适应的启发式算法，它使用基于预期 QoS 的随机路由策略，即平均作业响应时间。作业分配到概率较高的 VM 实例，该实例在一段时间内平均响应时间较低。合理的调度算法需要 2 个参数，即持续观察时间 D 和折扣系数 a。本节实验设置 D=5 s，a=0.7。

此外，用 3 个不同的指标来评估每种算法的性能。第一个指标是作业调度成功率，用来衡量有多少作业被成功处理，直观体现 QoS，当且仅当一个作业的响应时间低于预先定义的 QoS 要求时，此次调度满足 QoS 要求，调度成功。第二个指标是平均响应时间，用于衡量处理每个作业的平均响应时间。第三个指标是负载均衡率，用于衡量安全中台资源的利用率。一般来说负载均衡率越低，调度算法性能越好。换句话说，为了处理相同强度作业，高效的调度方法在调度过程中使用更少的资源，最终表现出较低的负载均衡率。

4.1.4 工作负载模式

设置 3 种不同的工作负载模式，工作负载的作业到达率按照规律随机生成，3 种模拟实验环境工作负载模式生成的参数如表 3 所示。作业类型概率分布和作业到达数量始终随着时间而变化。

表 3 负载模式生成的参数

负载模式	到达率	平均值	标准差	对应现实场景
随机	[0,100%]	53.53%	29.51%	随机使用
低频	[20%,40%]	30.07%	6.36%	低频使用
高频	[60%,80%]	70.32%	5.57%	高频使用

4.2 性能评估

本文进行了更长时间窗口的实验，结果如表 4 所示，统计除了前 40 s 外长达 2 小时的实验结果。

去除前 40 s 的原因是排除离线训练阶段对实时调度、正式运行造成的干扰。与现有方法相比，所提算法在短暂的学习适应后，在各种不同的工作负载模式下都能通过 QoS 感知的方式把前台作业合理地调度给安全中台资源，以提高性能，具体来说既保证了服务质量和负载均衡，还提高了 18.7%的作业调度成功率，同时降低了 34.2%的平均响应时间。

表 4　　工作负载模式实验结果

负载模式	算法	平均响应时间/s	作业调度成功率	负载均衡率
随机	random	0.807	51.3%	75.4%
	round-robin	0.426	72.1%	76.1%
	earliest	0.412	74.4%	76.7%
	DQN	0.203	95.7%	65.5%
	suitable	0.255	82.7%	64.1%
	sensibleR	1.108	43.8%	75.7%
低频	random	0.237	99.5%	29.8%
	round-robin	0.163	99.9%	29.7%
	earliest	0.158	99.9%	27.4%
	DQN	0.115	99.9%	26.8%
	suitable	0.057	99.9%	23.8%
	sensibleR	0.254	98.4%	33.7%
高频	random	11.637	11.4%	98.4%
	round-robin	10.362	12.6%	97.7%
	earliest	3.527	13.8%	91.4%
	DQN	0.357	93.7%	76.2%
	suitable	0.658	70.3%	73.8%
	sensibleR	11.246	12.2%	98.1%

4.2.1　随机工作负载模式

本节采用大幅度波动的随机工作负载模式来测试模型的性能，实验结果如图 4～图 6 所示。

如图 5 和图 6 所示，初始化的所有负载队列都为空，最初 5 s 涌入了大量的作业，所有方法表现不好但能正常运作；5～20 s，请求作业到达率较低，为每秒 8%～35%，所有方法均表现良好；25～125 s，由于作业量突增并且保持在极高频状态，等待队列压力过大，出现了堵塞的情况，所有方法均受到影响；125～300 s，作业不会持续高频输入，作业队列不再严重堵塞，中台调度有序进行，suitable 和 DQN 效果最好。

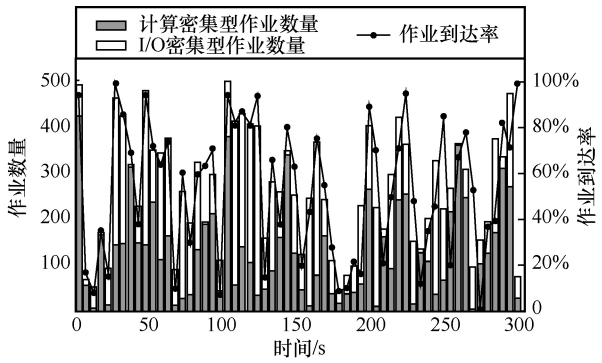

图 4　随机工作负载模式作业到达率

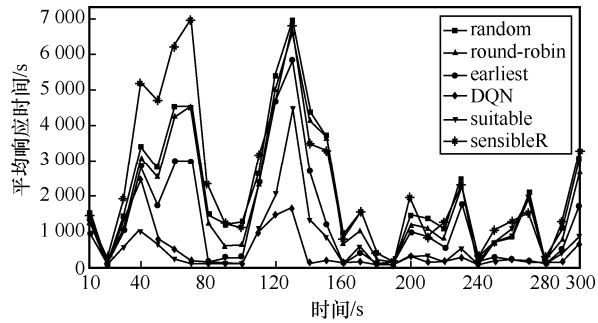

图 5　随机工作负载模式平均响应时间

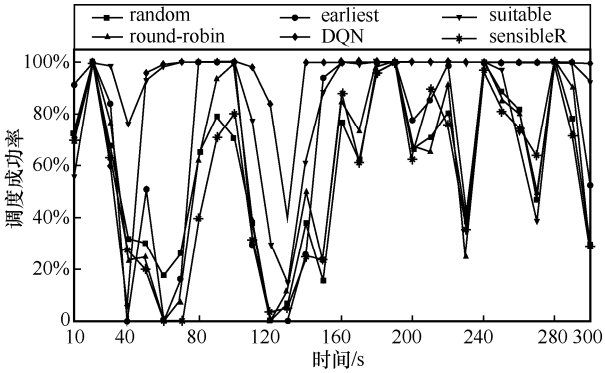

图 6　随机工作负载模式作业调度成功率

总体来看，50 s 之前，所提算法处在积极的训练阶段，与其他算法性能接近；50 s 左右，可以看到 DQN 逐渐完成了训练，适应了该工作负载模式，并和其他算法拉开了差距；之后无论是高频还是低频模式，均效果最佳，优于 suitable。

如表 4 中随机负载模式所示，DQN 取得了最低的平均响应时间、略次于 suitable 的负载均衡率和最高的作业调度成功率。suitable 表现次之，random 和 sensibleR 表现相对较差。

4.2.2　低频工作负载模式

为了测试算法在安全中台大多数日常低频静息使用场景中的性能，本节实验设置了低频工作负载模式，实验结果如图 7～图 9 所示。结合表 4

中低频负载模式，低频状态下几种算法均表现良好，平均响应时间普遍较低，均拥有较高的作业调度成功率和较低的负载均衡率，suitable 算法取得最低的平均响应时间，表现最好。如图 8 所示，最初 40 s 训练阶段后，DQN 平均响应时间逐渐超过除了 suitable 以外的其他算法，接着贴近于表现最优的 suitable 平稳运行。如图 9 所示，在低频模式下仅 sensibleR 的作业调度成功率有些波动，所有算法总体表现良好。

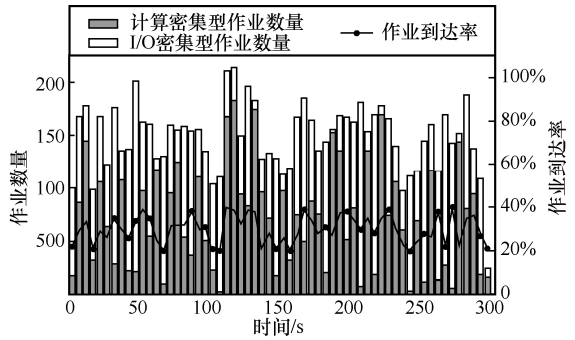

图 7 低频工作负载模式作业到达率

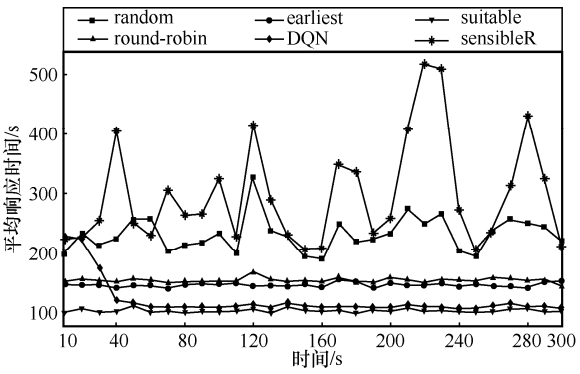

图 8 低频工作负载模式平均响应时间

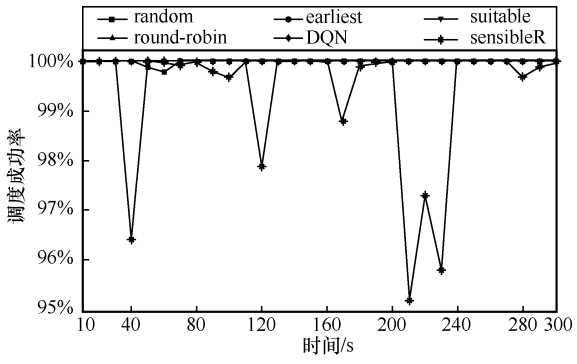

图 9 低频工作负载模式作业调度成功率

4.2.3 高频工作负载模式

为了测试算法在安全中台极端恶劣使用场景（如用户使用量爆发式增长并始终保持在高频）中的性能，设置了高频工作负载模式，实验结果如图 10～图 12 所示。结合表 4 中高频负载模式，极端高频状态使环境接近崩溃，大多数算法均难以适应这种超高强度的模式，除 DQN 和 suitable 以外的 4 种算法均无法正常运行，负载均衡率和作业调度成功率表现极差。DQN 和 suitable 算法仍能正常运行，suitable 算法受到了高频的影响性能下降，DQN 表现出完全不同于其他算法的稳定性，是 6 种算法中唯一保持高性能、高稳定性的算法，拥有最低的平均响应时间、较低的负载均衡率和最高的作业调度成功率。

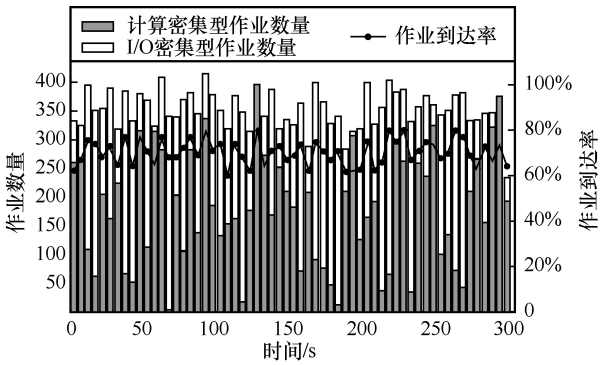

图 10 高频工作负载模式作业到达率

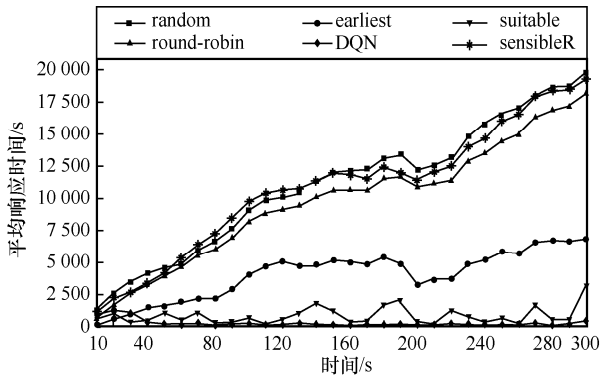

图 11 高频工作负载模式平均响应时间

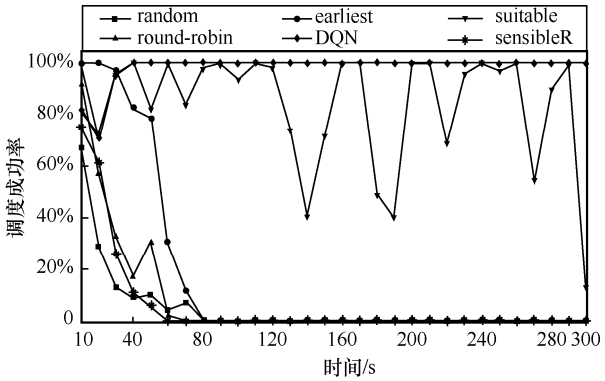

图 12 高频工作负载模式作业调度成功率

如图 11 和图 12 所示，在整个过程中，random、round-robin、earliest 的平均响应时间持续增长，直至系统瘫痪，成功率在 80 s 后全部接近 0；而 suitable 在高频工作负载模式中保持正常运行，在 0～290 s, 平均响应时间低于 2 500 ms；成功率高于 40%，但波动很大。而 DQN 始终保持极高的水准运行，最初 30 s 作业调度成功率在 70%～95%，其余时间均接近 100%，适应新工作负载模式的训练时间缩短到 20 s。

4.2.4　对比分析

在 3 种工作负载模式的实验结果中，通过所提算法与 5 种现有实时算法的对比可以看出，所提 DQN 算法适合各种场景，性能优于其他算法，可以归纳出以下结论。

1) 随着输入作业量的数量或频率增加，基于深度强化学习的软件定义安全中台 QoS 实时优化算法平均响应时间会增加。对比低频和高频工作负载模式可知，所提算法在高频工作负载模式中显现出更大的优势，尤其是当其他算法已经明显不能正常运行时，suitable 和所提算法依然满足可用性，所提算法是 6 种算法中唯一保持高性能、高稳定性的算法，拥有最低的平均响应时间、较低的负载均衡率和最高的作业调度成功率，显著提高 QoS。

2) 对比随机、低频、高频工作负载模式可知，提出的算法基于训练的经验，一但完成训练，便具有良好的稳健性。安全中台平面因为结构的封装，软件定义也使它远离应用平面和基础设施平面，后期平稳运行过程中产生的新数据较少。

3) 图 4～图 12 完整展示了实验过程中 0 s～40 s 的训练阶段和后续的运行阶段，而不是将实时调度与前期的离线训练分开展示，侧重点在安全中台资源的实时调度和 QoS 优化，离线训练并不是关注的重点，因为离线训练是在本地进行的，并不占用云端安全中台资源，对于安全中台资源的消耗几乎为 0，一旦完成训练就可以直接进行调度。除了训练应该离线完成外，如图 7 中低频时间，安全中台业务服务切换过程中提供可选的平滑在线无感部署，在线部署的优势在于重新部署新的安全中台服务时不中断原有服务，不需要切断系统重新训练，只需要上线离线训练后的新服务，在原有服务正常运行的过程中用极小的代价差异训练副本，无感切换即可，具有更好的可拓展性和容错性。因此所提算法在多变的软件定义安全中台环境中表现出良好的稳定性和稳健性，而且自身具有一定抗攻击能力和容灾能力，更加适用。

4) 负载均衡率指标能直观体现资源占用程度，直观地体现不同算法实际运行过程中的开销。如表 4 所示，在 3 种工作负载模式下的长期运行表现中，与现有实时算法相比，本文算法和 suitable 均取得了优势，suitable 负载均衡率表现最好。低频模拟静息环境中，所有算法都表现良好，负载均衡率接近。随机负载模式下，因为本文算法和 suitable 均有学习的特性，持续运行过程中和其他算法拉开了较大差距。高频负载模式下，其他算法进入性能瓶颈而无法正常运行，本文算法和 suitable 依然正常运行，证明其在大规模高负载作业场景下依然可用。

5　结束语

面向关键信息基础设施所面临的安全挑战及重点防护要求，为解决软件定义安全防护过程中遇到的 QoS 优化方案适用困难和性能下降问题，本文针对安全防护手段与业务场景不匹配问题，首先提出面向碎片化安全需求和安全场景的 SDSmp 自动控制框架，在此基础上提出软件定义安全中台 QoS 优化模型，实现实时自动化控制。更进一步，本文提出基于深度强化学习的软件定义安全中台 QoS 实时优化算法，并与目前的主流实时算法对照，3 个负载模式下的实验结果表明，所提算法不但提高了 18.7% 的作业调度成功率，而且降低了 34.2%的平均响应时间，具有很好的稳健性，更适用于实时环境，提高了服务质量。未来的研究计划在一个更碎片化的安全场景中调度作业，并将改进算法适用于各种类型的负载模式，更长期目标是在完全碎片化的软件定义安全中台的实时环境中始终保持高 QoS。

参考文献：

[1] LIU Y B, LU X Y, JIAN Y, et al. SDSA: a framework of a software-defined security architecture[J]. China Communications, 2016, 13(2): 178-188.

[2] ALHAJ A N, DUTTA N. Analysis of security attacks in SDN network: a comprehensive survey[C]//Proceedings of Contemporary Issues in Communication, Cloud and Big Data Analytics. Berlin: Springer, 2022: 27-37.

[3] QIU R X, QIN Y T, LI Y C, et al. A software-defined security middle platform architecture[C]//Proceedings of the 5th International Conference on Computer Science and Software Engineering. New York: ACM Press, 2022: 647-651.

[4] WEI Y, PAN L, LIU S J, et al. DRL-scheduling: an intelligent

QoS-aware job scheduling framework for applications in clouds[J]. IEEE Access, 2018, 6: 55112-55125.

[5] ZHANG G, QIU X F, CHANG W. Scheduling of security resources in software defined security architecture[C]//Proceedings of 2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC). Piscataway: IEEE Press, 2018: 494-503.

[6] MOUSSAID N E, TOUMANARI A, AZHARI M E. Security analysis as software-defined security for SDN environment[C]//Proceedings of the Fourth International Conference on Software Defined Systems. Piscataway: IEEE Press, 2017: 87-92.

[7] MOHAMED A, HAMDAN M, KHAN S, et al. Software-defined networks for resource allocation in cloud computing: a survey[J]. Computer Networks, 2021, 195: 108151.

[8] QI C, WU J X, HU H C, et al. Dynamic-scheduling mechanism of controllers based on security policy in software-defined network[J]. Electronics Letters, 2016, 52(23): 1918-1920.

[9] LUO S, BEN S M. Orchestration of software-defined security services[C]//Proceedings of 2016 IEEE International Conference on Communications Workshops. Piscataway: IEEE Press, 2016: 436-441.

[10] 兰巨龙, 张学帅, 胡宇翔,等. 基于深度强化学习的软件定义网络 QoS 优化[J]. 通信学报, 2019, 40(12): 60-67.
LAN J L, ZHANG X S, HU Y X, et al. Software-defined networking QoS optimization based on deep reinforcement learning[J]. Journal on Communications, 2019, 40(12): 60-67.

[11] 吴平, 常朝稳, 左志斌, 等. 基于地址重载的 SDN 分组转发验证[J]. 通信学报, 2022, 43(3): 88-100.
WU P, CHANG C W, ZUO Z B, et al. Address overloading-based packet forwarding verification in SDN[J]. Journal on Communications, 2022, 43(3): 88-100.

[12] MORALES E F, ZARAGOZA J H. An introduction to reinforcement learning[J]. Machine Learning, 2011: doi.10.4018/978-1-60960-165-2.Ch004.

[13] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Playing atari with deep reinforcement learning[J]. arXiv Preprint, arXiv: 1312. 5602, 2013.

[14] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529-533.

[15] CHENG L, KALAPGAR A, JAIN A, et al. Cost-aware real-time job scheduling for hybrid cloud using deep reinforcement learning[J]. Neural Computing and Applications, 2022, 34(21): 18579-18593.

[16] ABUNDO M, DI-VALERIO V, CARDELLINI V, et al. QoS-aware bidding strategies for VM spot instances: a reinforcement learning approach applied to periodic long running jobs[C]//Proceedings of 2015 IFIP/IEEE International Symposium on Integrated Network Management. Piscataway: IEEE Press, 2015: 53-61.

[17] ARULKUMARAN K, DEISENROTH M P, BRUNDAGE M, et al. Deep reinforcement learning: a brief survey[J]. IEEE Signal Processing Magazine, 2017, 34(6): 26-38.

[18] 盛妍, 朱青, 张明杰, 等. 基于数据中台的智能标签关键技术研究与应用[J]. 电子技术应用, 2022, 48(3): 73-77.
SHENG Y, ZHU Q, ZHANG M J, et al. Research and application on key technology of intelligent tag based on data middle platform[J]. Application of Electronic Technique, 2022, 48(3): 73-77.

[19] CHENG F, HUANG Y F, TANPURE B, et al. Cost-aware job scheduling for cloud instances using deep reinforcement learning[J].Cluster Computing, 2022, 25(1): 619-631.

[20] HUANG Y F, CHENG L, XUE L T, et al. Deep adversarial imitation reinforcement learning for QoS-aware cloud job scheduling[J]. IEEE Systems Journal, 2022, 16(3): 4232-4242.

[21] JAIN V, KUMAR B. QoS-aware task offloading in fog environment using multi-agent deep reinforcement learning[J]. Journal of Network and Systems Management, 2022, 31(1): 1-32.

[22] CHO C, SHIN S, JEON H, et al. QoS-aware workload distribution in hierarchical edge clouds: a reinforcement learning approach[J]. IEEE Access, 2020, 8: 193297-193313.

[23] LILLICRAP T P, HUNT J J, PRITZEL A, et al. Continuous control with deep reinforcement learning[J]. arXiv Preprint, arXiv: 1509.02971, 2015.

[24] NEJAD M M, MASHAYEKHY L, GROSU D. Truthful greedy mechanisms for dynamic virtual machine provisioning and allocation in clouds[J]. IEEE Transactions on Parallel and Distributed Systems, 2015, 26(2): 594-603.

[25] WANG L, GELENBE E. Adaptive dispatching of tasks in the cloud[J]. IEEE Transactions on Cloud Computing, 2015, 6(1): 33-45.

[作者简介]

李元诚（1970– ），男，山东烟台人，博士，华北电力大学教授、博士生导师，主要研究方向为密码学、信息安全等。

秦永泰（1998– ），男，甘肃定西人，华北电力大学硕士生，主要研究方向为深度强化学习、软件定义安全等。