控制与决策
Control and Decision

基于深度强化学习的资源受限条件下的DIDS任务调度优化方法

赵旭, 黄光球, 江晋, 李巾

引用本文:
赵旭,黄光球,江晋,李巾. 基于深度强化学习的资源受限条件下的DIDS任务调度优化方法[J]. 控制与决策, 2022, 37(11): 3052−3057.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2021.0448

基于深度强化学习的资源受限条件下的DIDS任务调度优化方法

赵 旭[1,2], 黄光球[1†], 江 晋[3], 李 巾[4]

(1. 西安建筑科技大学 管理学院，西安 710055；2. 西安工程大学 电子信息学院，西安 710048；
3. 西安工程大学 人文学院，西安 710048；4. 陕西省社会科学院，西安 710061)

摘 要: 在节点性能有限的边缘计算环境下进行分布式入侵检测系统(distributed intrusion detection system, DIDS)的任务分配,是一种典型的资源受限任务调度问题. 针对该问题,提出基于深度强化学习的DIDS低负载任务调度方案. 该方案将任务调度过程描述为马尔科夫决策过程(Markov decision process, MDP)并建立模型的相关空间和价值函数,找到保持DIDS低负载状态的最优策略. 针对状态和动作空间过大且高维连续的问题,提出通过深度循环神经网络进行函数拟合. 实验表明,所提出方案可使DIDS在网络变化中动态调节调度策略,保持系统整体的低负载,而安全指标没有明显降低.

关键词: 资源受限；任务调度；深度强化学习；深度循环神经网络；入侵检测；边缘计算

中图分类号: TP393.08；C931.2 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2021.0448

引用格式: 赵旭,黄光球,江晋,等. 基于深度强化学习的资源受限条件下的DIDS任务调度优化方法[J]. 控制与决策, 2022, 37(11): 3052-3057.

An optimization method for DIDS task scheduling under resource-constrained conditions based on deep reinforcement learning

ZHAO Xu[1,2], *HUANG Guang-qiu*[1†], *JIANG Jin*[3], *LI Jin*[4]

(1. College of Management，Xi'an University of Architecture and Technology，Xi'an 710055，China；2. School of Electronic and Information，Xi'an Polytechnic University，Xi'an 710048，China；3. College of Humanities，Xi'an Polytechnic University，Xi'an 710048，China；4. Shaanxi Academy of Social Sciences，Xi'an 710061，China)

Abstract: The task assignment of distributed intrusion detection systems (DIDS) in the edge computing environment with limited node performance is a typical resource-constrained task scheduling problem. To solve this problem, a DIDS low-load task scheduling scheme based on deep reinforcement learning is proposed. The task scheduling process is first described as a Markov decision process and the relevant space and value function of the model are established to find the optimal strategy for maintaining the low-load state of the DIDS. To solve the problem of excessively large action space and high-dimensional continuity, a deep recurrent neural network is proposed to perform function fitting. The experimental results show that the proposed scheme enables the DIDS to dynamically adjust the scheduling strategy during network changes, keeping the overall system load low, and the safety indicators are not significantly reduced.

Keywords: resource-constrained；task scheduling；deep reinforcement learning；deep cyclic neural network；intrusion detection；edge computing

0 引 言

边缘计算是指将运算任务由网络中心节点迁移至网络边缘节点处理,以此保证数据处理的实时性,并降低云计算中心的计算负载. 云计算中存在的网络安全威胁在边缘环境中会因为计算模式的复杂性和数据的多源异构性而表现得更加复杂,云计算通常采用分布式入侵检测系统(DIDS)进行安全防护,这些DIDS可以依赖数据中心中高性能的硬件设备完成异常检测. 但是在网络边缘,因为节点的处理能力和存储容量受到限制,传统的DIDS需要被改造以适

收稿日期: 2021-03-18；**录用日期:** 2021-07-30.
基金项目: 国家自然科学基金项目(71874134)；西安市科技计划项目(21XJZZ0024)；陕西省教育厅专项项目(20JX014).
责任编委: 王凌.
†通讯作者. E-mail: huangnan93@163.com.

应低负载环境,这种对DIDS的改造将面临资源受限项目调度问题.

1 相关研究

资源受限任务调度问题在运筹学领域又被称为资源受限项目调度问题(resource-constrained project scheduling problem, RCPSP), 研究在有限资源约束下如何合理安排任务调度,以实现某个目标的最优化[1]. 目前,求解RCPSP问题的算法主要有精确算法、下界计算和启发式算法. 精确算法主要基于分支定界的方法枚举局部调度,该方法虽然能得到最优解,但是时间往往过长,不适合实时性要求高的环境. 下界计算基于线性规划并且通过放松部分逻辑关系约束条件实现,计算过程中需要在下界的质量与计算时间之间寻求权衡. 启发式算法通过一定的启发式规则得到理想解,虽然在短时间内能够求得可接受的理想解,但理想解与最优解之间容易存在偏差[2].

在启发式算法中,元启发式算法目前应用得最多,具体包括进化算法、蚁群算法等. 例如, Kaur等[3]通过Tchebycheff分解的多目标进化算法实现效率与等待时间以及带宽之间的权衡. Zhang[4]通过建立风暴调度模型,解决边缘计算环境中风暴节点的调度优化. Lin等[5]提出SDMMF分配算法对边缘计算环境下IDS完成资源分配. 元启发式算法中以进化算法应用最为广泛,例如Zhao等[6]将遗传算法与FFD近似算法相结合,将不同类型流量调度至对应检测引擎进行检测.

随着机器学习的发展,强化学习(reinforcement learning)逐渐受到重视,其通过智能体与环境交互过程中获得的奖励指导行为,从而使智能体达到获得最大奖励的目标. 而深度强化学习(deep reinforcement learning, DRL)能够将深度学习的感知能力与强化学习的决策能力相结合,为复杂系统的感知决策问题提供了解决思路. 与强化学习相比,深度强化学习可以在状态和动作空间过大且是高维连续时,通过神经网络进行函数拟合,解决强化学习使用表格存储Q值带来的内存太大等问题. 为了解决状态空间中的高维问题, Chen等[7]提出一种基于双深层Q网络(DQN)的计算分流算法,以在不了解网络动力学先验知识的情况下学习最优策略. 将Q函数分解技术与双重DQN相结合,产生一种用于解决随机计算卸载的新型学习算法.

边缘计算的出现引起对资源受限环境下任务调度问题的关注,目前对该问题的研究中,深度强化学习成为一种主流方法. 但相关文献的主要研究目标大多是优化调度过程的响应时间和能耗问题,对边缘计算环境下的DIDS调度问题关注得很少,而以低负载作为研究目标更为少见.

本文针对以上问题,提出基于深度强化学习的DIDS低负载任务调度方案. 该方案将任务调度过程描述为马尔科夫决策过程(markov decision process, MDP)并建立模型的相关空间和价值函数,找到保持DIDS低负载状态的最优策略. 针对状态和动作空间过大且高维连续的问题,提出通过深度循环神经网络进行函数拟合. 实验表明,所提出方案可使DIDS在网络变化中动态调节调度策略,保持系统整体的低负载,而安全指标没有明显降低.

2 分布式入侵检测架构

本文设计的面向边缘计算环境的DIDS核心部分由调度器和多个检测引擎组成. DIDS启动后,首先对检测引擎进行性能评估,确定每个检测引擎的性能等级;然后将捕获的流量进行预处理,在预处理过程中可对数据包进行包括负载评估等预先检测工作. 调度器将根据深度强化学习确定的最优策略,向检测引擎分配待检测的数据包. 检测引擎将数据包内容与规则库中的规则进行模式匹配,如果符合规则,则表明可能带有危险信息,随即报警或录入日志[8].

为了客观评估各个检测引擎的性能,设计如下方法: 预先对每个检测引擎用同一流量进行测试,收集其对测试流量的数据量da(单位bit)、检测时dt(单位ms)、内存占用mu(单位Mb)和检测引擎i的CPU频率F_i(单位GHz)信息,并定义检测引擎的性能指标pi(performance index),计算模型如下:

$$\mathrm{pi}=\frac{\mathrm{da}\times F_i}{\mathrm{dt}\times \mathrm{mu}}. \tag{1}$$

采用数据包长度与以太网最大传输单元(通常为1 500 bytes)的比值确定数据包对系统产生的负载的评估[9].

3 MDP建模

马尔科夫决策过程是用于序贯决策的数学模型,其特点是在环境交互过程中,根据环境给予的奖励和惩罚不断学习,从而修正自己的行为以获得最大利益.

与监督学习和非监督学习的各种算法相比,马尔科夫决策过程具备主动学习能力,能通过执行某些动作去探索,并从环境中获得反馈以调整动作,而监督学习和非监督学习都只能在给定的数据集上学习. 本文所研究的调度问题根据网络流量的变化及时调整策略,所以通过马尔科夫决策过程建模更为适

合.

本文所设计的DIDS有D个不同性能等级的检测引擎. 在预处理阶段,数据包将被分为K个负载等级. 检测时间服从指数分布, 数据包的到达过程可以看作K个独立的泊松过程. 考虑数据包到达和检测结束的时刻,此时嵌入链为马尔科夫链. 建立模型以低负载为主要优化目标,通过状态-行为价值函数确定最小的平均负载准则,从而找到实现DIDS最小负载的最优策略.

3.1 状态空间

将$s=(N(D,K),B(K),r)$设为状态,其中$N(D,K)$为一个向量,具有形式$(n_{10},n_{11},\dots,n_{1K-1},n_{20},\dots,n_{DK-1})$,描述了DIDS的工作状态,包括尚未分配检测任务的检测引擎的分布以及正在为各等级数据包检测的检测引擎状况;$B(K)$也为一个向量,具有形式$(b_1,b_2,\dots,b_K)$,描述了正在等待检测的数据包情况,包括各种数据包的数量;r取值于集合$K,K-1,\dots,1,0$,描述最后一个到达的数据包的情况. 当队列长度的限制b确定后,便可以定义一个含有所有可能状态的集合X,如下所示:

$$X=\{(N(D,K),B(K),r)\}. \tag{2}$$

下面列出集合X中几种典型的可能状态:

1) 系统中有空闲的检测引擎,刚好有一个数据包到达,经过负载评估是第j等级数据包,那么X_1作为X集合中的一个状态,如下所示:

$$X_1=\left\{(N(D,K),B(K),j)\Big|\sum_{d=0}^{D}n_{d0}>0,\sum_{k=1}^{K}b_k=0\right\}. \tag{3}$$

其中:d为检测引擎的性能等级,$d=1,2,\dots,D$;状态$(N(D,K),B(K),j)$表示新到的数据包带来了第j等级的检测需求;n_{d0}为尚未分配使用的d等级的检测引擎数量;b_k为等待检测的k等级数据包排队的队列长度.

2) 系统中没有可用的检测引擎时的所有可能状态X_2可以表示为

$$X_2=\left\{(N(D,K),B(K),0)\Big|\sum_{d=1}^{D}n_{d0}=0,\sum_{k=1}^{K}b_k\leqslant b\right\}. \tag{4}$$

3) 系统中仍有空闲的检测引擎且无数据包等待检测(此时$r=0$)的所有可能状态X_3可以表示为

$$X_3=\left\{(N(D,K),B(K),0)\Big|\sum_{d=0}^{D}n_{d0}>0,\sum_{k=1}^{K}b_k=0\right\}. \tag{5}$$

4) 系统中只有一个空闲的检测引擎且有等待检测的数据包的所有可能状态(这种情况较为少见)可以表示为

$$X_4=\left\{(N(D,K),B(K),0)\Big|\sum_{d=0}^{D}n_{d0}=1,\sum_{k=1}^{K}b_k>0\right\}. \tag{6}$$

3.2 动作空间

上面几种情况中,对于X_1中的状态,调度器需要选择指派哪一等级的检测引擎处理该数据包;对于X_4中的状态,系统需要考虑目前唯一空闲的检测引擎应该检测队列中哪一等级数据包;对于X_2和X_3中的状态,系统不需要做出选择. 所以状态空间的动作空间可定义为

$$A(s)=\begin{cases}\{d|n_{d0}>0\},\ s\in X_1;\\ \{0\},\ s\in X_2\bigcup X_3;\\ \{k|b_k>0,\ k=1,2,\dots,K\},\ s\in X_4.\end{cases} \tag{7}$$

3.3 转移速率与转移概率

转移概率依赖于系统当前所处的状态和调度器选取的行动决定. 本文使用的是马尔科夫决策过程,所以转移概率可以通过转移速率求得. 转移速率可以由如下几种情况确定:

1) 对于X_1,可能的转移状态有两种,即$s'\in X_3$和$s'\in X_1\bigcup X_2$. 对于前者,对应的转移速率为

$$p(s'|s,d)=\begin{cases}n_{ij}\mu_{ij},\ i\neq d;\\ n_{ij}\mu_{ij},\ i\neq d,\ j\neq k;\\ (n_{ij}+1)\mu_{ij},\ i\neq d,\ j=k.\end{cases} \tag{8}$$

对于后者,对应的转移速率为$p(s'|s,d)=\lambda_j$. 其中:n_{ij}为正在检测j等级数据包的i等级检测引擎数,μ_{ij}为i等级的检测引擎对j等级数据包的检测率,λ_j为j等级数据包到达率,$j=1,2,\dots,K$.

2) 对于X_2,可能的转移状态有两种,即$s'\in X_2$和$s'\in X_4$. 对于前者,对应的转移速率为$p(s'|s,0)=\lambda_j$;对于后者,对应的转移速率为$p(s'|s,0)=n_{ij}\mu_{ij}$.

3) 对于X_3,可能的转移状态有一种,即$s'\in X_1$,对应的转移速率也为$p(s'|s,0)=n_{ij}\mu_{ij}$.

4) 对于X_4,可能的转移状态有两种,即$s'\in X_2$和$s'\in X_1\bigcup X_4$. 对于前者,对应的转移速率为$p(s'|s,k)=\lambda_j$;对于后者,对应的转移速率与式(8)相同.

转移速率矩阵的对角元素可以定义为

$$p(i'|i,a)=\sum_{j\in X}-p(j|i,a),\ a\in A(i). \tag{9}$$

对任意确定性策略π,可以得到对应的转移速率矩阵$p(\pi)$. 根据连续时间的马尔科夫决策过程理论,得到转移概率矩阵$p(\pi)$为

$$P(\pi)=\lambda^{-1}[p(\pi)]+I. \tag{10}$$

其中λ满足$0<\lambda=\sup\limits_{i\in S,a\in A(i)}-p(i|i,a)\leqslant M$. 对于转移速率矩阵$p(\pi)$,将每一行除以该行对应对角线上的元素后,再加上一个单位矩阵,也可以得到一个嵌入马尔科夫链的转移概率矩阵$p'(\pi)$. 通过这两种不同方法得到的系统,其最优策略和对应的值函数都是相同的.

3.4　价值函数和最优策略

考虑到检测时间的分布通常是指数分布,在状态s时采取行动a的期望负载,即基于策略π的状态-价值函数为

$$v_\pi(s)=\begin{cases}0,\ s\in X_3\bigcup X_2;\\ l_k+\int_0^\infty l_{dk}t\mathrm{d}(1-\mathrm{e}^{-\mu_{dk}t}),\ \text{otherwise}.\end{cases} \tag{11}$$

其中:l_k为检测第k等级数据包对检测引擎带来的最小负载,l_k依赖于要检测的数据包的负载等级k;l_{dk}为平均负载,取决于检测引擎的性能等级d和数据包的负载等级k. 那么,实现最低负载的状态-行为价值函数$q^*(s,a)$为

$$q^*(s,a)=\liminf_{n\to\infty}\frac{E_\pi\left\{\sum\limits_{i=0}^{n}q(Y_i,\pi(Y_i))|s\right\}}{E_\pi\left\{\sum\limits_{i=0}^{n}\tau_i|s\right\}}. \tag{12}$$

其中:Y_i为决策时刻的状态,s为初始状态,τ_i为决策时刻i的平均滞留时间.

3.5　神经网络架构

通常当状态空间和动作空间较小且维数不高时,可以使用表格形式存储每个状态和动作对应的Q值(即$q(s,a)$的值). 就本文所涉及的问题而言,状态和动作空间过大且高维连续,所以使用表格存储Q值将带来内存太大等诸多问题. 对于该问题,本文通过神经网络进行函数拟合,利用神经网络接受外部的状态信息,使相近的状态得到相近的输出动作.

由于网络流量中含有的大量视频音频等均属于时间序列数据,存在时间关联性和整体逻辑特性. 与卷积神经网络相比,循环神经网络(recurrent neural network, RNN)更适合处理时间序列数据的建模,本文选择使用深度循环神经网络. 所设计的深度循环神经网络结构包括输入层、隐藏层和输出层. 相对于普通的全连接神经网络,在隐藏层中多增加了信息记忆功能,即每一时刻隐藏层的输入不仅是输入层的输出,还包含上一时刻隐藏层的输出. 所以对具体某个隐藏层而言,在t时刻其状态s_t的计算公式为

$$s_t=\tanh(Ux_t+Ws_{t-1}). \tag{13}$$

其中:s_{t-1}为$t-1$时刻的状态,W为状态s的权重参数矩阵,x_t为t时刻的输入,U为输入的序列信息的权重参数矩阵. t时刻,状态s_t的输出为

$$\hat{y}_t=\mathrm{softmax}(Vs_t). \tag{14}$$

其中:softmax为输出的激活函数,V为输出序列信息的权重参数矩阵.

由于处理的信息量过大,为了增加模型的表达能力,在深度循环神经网络中堆叠多个隐藏层. 深度循环神经网络与状态动作和调度器的工作关系如图1所示.

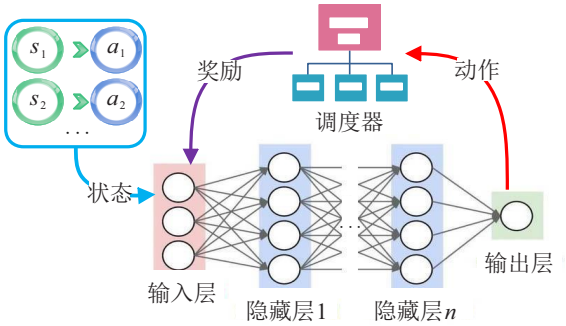

图 1　深度循环神经网络与调度器

为了度量循环神经网络模型输出产生的误差,使用交叉熵的损失函数优化权重参数矩阵U、W和V,使得输入的序列数据经过循环神经网络处理后的输出值更加接近真实的输出值.

设输出的时间序列总数为T,循环神经网络模型的总损失函数为

$$L=\sum_{t}^{T}-y_t\log\hat{y}_t. \tag{15}$$

其中:y_t为t时刻真实值,$\hat{y}_t$为t时刻预测值. 训练过程中使用的算法是时间反向传播算法(backpropagation through time, BPTT). 该算法沿着需要优化的参数的负梯度方向不断寻找更优的点直至收敛,具体步骤如下.

step 1: 前向计算每个神经元的输出值.

step 2: 沿向上和向前两个方向,反向计算每个神经元的误差项,误差项同时也是误差函数对神经元的加权输入的偏导数.

step 3: 计算每个权重的梯度.

step 4: 用随机梯度下降算法更新权重.

4 实验及结果分析

实验通过仿真环境对所提出方案在DIDS上进行测试,测试过程中将所提出方案与SDMMF算法[5]、混合遗传算法(Hybrid GA)[8]和迭代局部搜索(ILS)算法[10]进行对比. 测试目的是判断使用所提出方案后,调度器能否在DIDS检测能力没有明显降低的情况下具有更好的低负载优势. 在实验中,为了防止在出现突发性网络流量激增的情况下过度的低负载带来丢包率升高的可能,采用前期研究中提出的低负载与丢包率平衡原则[9],即加入2个参数:丢包率PLR的低阈值T_L和高阈值T_H. 平衡原则是当丢包率升高时,调度器增加给高效率检测引擎分配任务的概率. 反之,当DIDS整体负载较高时,调度器减少给低效率的检测引擎分配任务的概率,具体为

$$p_\pi(d,n)=\frac{\eta_\pi(d)}{\eta_\pi}\sum_{s\in X_b(d,n)}\mathrm{pb}^*_\pi(s),\ T_L<\mathrm{PLR}<T_H. \tag{16}$$

其中$\mathrm{pb}^*_\pi(s)$为s状态下与策略π相对应的平稳概率分布,$\eta_\pi(d)$为策略π下d等级检测引擎的检测效率,η_π为策略π下系统整体的检测效率,$X_b(d,n)$可表示为

$$X_b(d,n)=\left\{s\in(n_{dk},B(K),r)\in X\Big|\sum_{k=1}^{K}n_{dk}=n\right\}. \tag{17}$$

在实验中,测试内容分为性能和安全两个方面,性能方面包括内存占用、检测数量和系统负载,安全方面包括丢包率和恶意特征检测率.

4.1 实验环境

本文基于EdgeCloudSim构建一个仿真实验系统. 在EdgeCloudSim中,有5个主要的基本模块,分别是核心仿真模块、联网模块、负载生成模块、移动模块和边缘协调模块. 实验中,在核心仿真模块运行Edge Computing环境;在联网模块连接各检测引擎和规则库,并处理传输队列;在边缘协调模块内放入调度器;在负载生成模块发送测试数据集. 采用的测试数据集分别为NSL-KDD数据集和WSN-DS数据集. 在边缘计算环境下,靠近WSN终端的一些攻击类型都可以在这两个数据集中体现.

4.2 性能测试

1) 内存占用.

在实验中,首先对不同性能等级的检测引擎测算其工作效率,分为5个不同的性能等级,根据组距分组方法“一组数据所分的组数应不少于5组”的原则进行划分,通过工作效率数据上限与下限的差确定组距,最后整理成频数分布表将所有检测引擎分组,同时将低阈值TL设置为对丢包率宽容度较大的20%.

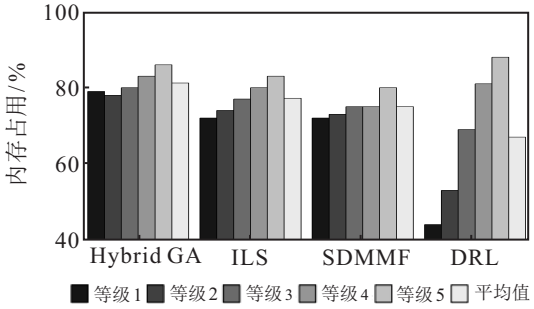

图2 内存占用对比

由图2可见,所提出方案的内存占用比其他算法都小,与轻量级的SDMMF相近. 而混合遗传算法因为时间和空间复杂度较大的问题,占据的内存空间最大. 因此,所提出方案在内存占用方面相比其他算法具有更少的优势.

2) 系统负载.

系统负载通过对不同网速下各检测引擎的平均负载进行累加得出,即

$$\mathrm{avgload}=\sum_{i=0}^{D}\left(l_k+\int_0^{\infty}l_{dk}t\mathrm{d}(1-\mathrm{e}^{-\mu_{dk}t})\right). \tag{18}$$

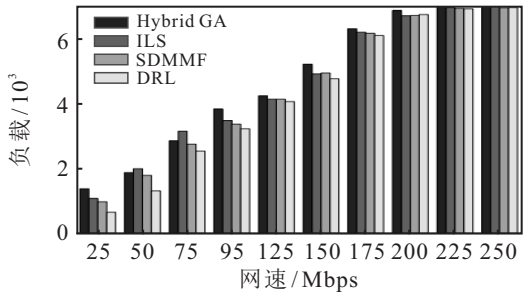

图3 负载测试

由图3可见,所提出算法在模拟边缘网络的低速流量中有明显的低负载优势. 只有在模拟核心网络的高速流量下,调度器才开始逐渐向低丢包率原则倾斜,与同样强调轻量级任务调度的SDMMF算法结果相对接近. 因此,所提出方案在边缘计算环境下能有效降低系统负载.

4.3 安全测试

1) 丢包率.

在实验中,如果对低阈值T_L设置为对丢包率宽容度较大的20%,会发现在低于20%丢包率(对应网速约小于95 Mbps)的区间内并没有显露出低丢包率的优势. 为了降低所提出方案在低网速阶段的丢包

率,将低阈值设置为10 %. 测试结果如图4所示.

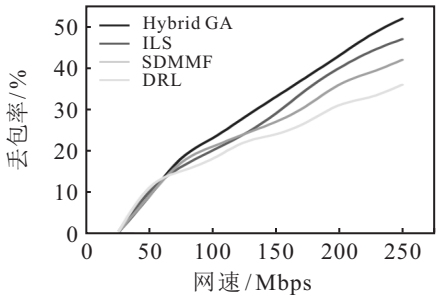

图 4　丢包率($T_L = 10\,\%$)

图4中,各算法10 %的丢包率大约在50 Mbps后出现,所以DRL算法在50 Mbps后丢包率的上升趋势与其他算法相比有所降低. 当网速为250 Mbps时,DRL算法的检测率低于其他算法8.1 %～20.3 %.

2) 恶意特征检测率.

恶意特征检测率由每种算法对NSL-KDD和WSN-DS两个数据集所检测到的恶意特征数量除以恶意特征总数得到. 在实验中,当T_L设置为20 %时,所提出方案的恶意特征检测率仅在网速为25～50 Mbps时相比其他算法降低0.620 %～1.720 %,当网速提升到100 Mbps后,检测率普遍高于其他算法. 当T_L设置为10 %时,所提出方案的检测率优势更为明显,在50 Mbps时已经开始超过其他算法. 当网速为250 Mbps时,所提出方案的检测率高于其他算法7.8 %～19.4 %.

5　结　论

针对边缘环境中设备存在处理性能受限的问题,本文提出基于深度强化学习的DIDS低负载任务调度方案. 通过该方案可以确定保持DIDS低负载状态的最优策略. 实验表明,所提出方案与其他算法相比,具有更好的低负载性能. 未来会使用长短期记忆网络解决循环神经网络对序列数据的长依赖问题,以提高训练性能.

参考文献(References)

[1] 王凌, 郑环宇, 郑晓龙. 不确定资源受限项目调度研究综述[J]. 控制与决策, 2014, 29(4): 577-584.
(Wang L, Zheng H Y, Zheng X L. Survey on resource-constrained project scheduling under uncertainty[J]. Control and Decision, 2014, 29(4): 577-584.)

[2] 陈俊杰, 同淑荣, 叶正梗, 等. 资源受限多项目调度问题的两阶段算法[J]. 控制与决策, 2020, 35(8): 2013-2020.
(Chen J J, Tong S R, Ye Z G, et al. Two-stage algorithm for resource-constrained multi-project scheduling problem[J]. Control and Decision, 2020, 35(8): 2013-2020.)

[3] Kaur K, Garg S, Aujla G S, et al. Edge computing in the industrial Internet of things environment: Software-defined-networks-based edge-cloud interplay[J]. IEEE Communications Magazine, 2018, 56(2): 44-51.

[4] Zhang L. Dynamic programming algorithm and bat algorithm based storm nodes scheduling in edge computing[J]. International Journal of Innovative Computing Information and Control, 2020, 16(3): 1021-1033.

[5] Lin F H, Zhou Y T, An X S, et al. Fair resource allocation in an intrusion-detection system for edge computing: Ensuring the security of internet of things devices[J]. IEEE Consumer Electronics Magazine, 2018, 7(6): 45-50.

[6] Zhao X, Huang G Q, Mousoli R. A multi-threading solution to multimedia traffic in NIDS based on hybrid genetic algorithm [J]. International Journal of Network Security, 2020, 22(3): 425-434.

[7] Chen X F, Zhang H G, Wu C, et al. Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning[J]. IEEE Internet of Things Journal, 2019, 6(3): 4005-4018.

[8] Zhao X, Jiang J, Reza M. An improved solution for multimedia trafc in NIDS based on elitist strategy[J]. International Journal of Circuits, Systems and Signal Processing, 2019, 13: 40-45.

[9] Zhao X, Huang G Q, Gao L, et al. Low load DIDS task scheduling based on Q-learning in edge computing environment[J]. Journal of Network and Computer Applications, 2021, 188: 103095.

[10] An X S, Lü X, Yang L, et al. Node state monitoring scheme in fog radio access networks for intrusion detection[J]. IEEE Access, 2019, 7: 21879-21888.

作者简介

赵旭 (1978−), 男, 副教授, 博士, 从事网络安全、边缘计算等研究, E-mail: zhaoxu@xpu.edu.cn;

黄光球 (1964−), 男, 教授, 博士生导师, 从事网络安全等研究, E-mail: huangnan93@163.com;

江晋 (1981−), 女, 讲师, 从事调度优化等研究, E-mail: 40466544@qq.com;

李巾 (1981−), 女, 副研究员, 从事智能优化等研究, E-mail: 21zero@163.com.

(责任编辑：郑晓蕾)