计算机科学
COMPUTER SCIENCE

基于强化学习的智能化渗透路径规划与求解优化

李成恩, 朱东君, 贺杰彦, 韩兰胜

引用本文

李成恩, 朱东君, 贺杰彦, 韩兰胜. 基于强化学习的智能化渗透路径规划与求解优化[J]. 计算机科学, 2024, 51(11): 329-339.

LI Cheng'en, ZHU Dongjun, HE Jieyan, HAN Lansheng. Intelligent Penetration Path Planning and Solution Optimization Based on Reinforcement Learning [J]. Computer Science, 2024, 51(11): 329-339.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.231000207

基于强化学习的智能化渗透路径规划与求解优化

李成恩[1]　朱东君[1]　贺杰彦[1]　韩兰胜[1,2]
1 华中科技大学网络空间安全学院　武汉 430000
2 武汉金银湖实验室　武汉 430000
(m202271758@hust.edu.cn)

摘　要　在大数据技术广泛应用的背景下,传统渗透测试过于依赖专家经验和人工操作的问题日益显著。自动化渗透测试旨在解决上述问题以达到更准确全面地发现系统安全漏洞的效果,而寻找最优渗透路径是自动化渗透测试中最重要的任务。然而,当前的主流研究试图在包含大量冗余路径的原始解空间中规划最优路径,导致问题的求解复杂度大幅提升;此外,当前研究对漏洞利用和正奖励获取动作的评估不够。通过剔除大量冗余渗透路径,并采取漏洞利用样本增强方法和正奖励样本增强方法,可以简化问题并优化训练过程。基于此,结合解空间转换和样本增强,提出了 MASK-SALT-DQN 算法,并定性和定量地分析了该方法对模型求解过程的影响,通过压缩比来衡量解空间转换给模型完成目标所带来的收益。实验表明,原始解空间中冗余解路径的比例始终保持在 83%以上,证明了解空间转换的必要性。此外,在标准场景下,理论压缩比为 57.2,实验压缩比与理论压缩比的误差仅为 1.40%,且相比基线方法,MASK-SALT-DQN 在所有实验场景下均有最优的表现,证明了其有效性和先进性。

关键词:渗透路径规划;强化学习;解空间转换;样本增强;压缩比
中图分类号　TP393

Intelligent Penetration Path Planning and Solution Optimization Based on Reinforcement Learning

LI Cheng'en[1],ZHU Dongjun[1],HE Jieyan[1] and HAN Lansheng[1,2]
1 School of Cyber Science and Engineering,Huazhong University of Science and Technology,Wuhan 430000,China
2 Wuhan Jinyinhu Laboratory,Wuhan 430000,China

Abstract　In the background of the widespread application of big data technology,the problems that traditional penetration testing overly relies on expert experience and manual operation have become more significant. Automated penetration testing aims to solve the above problems,so as to discover system security vulnerabilities more accurately and comprehensively. Finding the optimal penetration path is the most important task in automated penetration testing. However,current mainstream research suffers from the following problems:1)seeking the optimal path in the original solution space,which contains numberous redundant paths,significantly increases the complexity of problem-solving;2)evaluation of vulnerability exploitation and positive reward obtainment actions is not enough. The problem-solving can be optimized by eliminating a significant number of redundant penetration paths and employing exploit sample enhancement and positive reward sample enhancement methods. Therefore,this paper proposes the MASK-SALT-DQN algorithm by integrating solution space transformation and sample enhancement methods. It qualitatively and quantitatively analyzes the influence of the proposed algorithm on the model solving process,proposing the compression ratio to measure the benefits of solution space transformation. Experiments indicate that the proportion of redundant solution paths in the original solution space consistently remains over 83%,proving the necessity of solution space transformation. In addition,in standard experiment scenario,the theoretical compression ratio is 57.2,and the error between the experimental compression ratio and theoretical value is only 1.40%. Moreover,in comparison to baseline methods,MASK-SALT-DQN has the optimal performance in all experiment scenarios,which confirms its the effectiveness and superiority.

Keywords　Penetration path planning,Reinforcement learning,Solution space transformation,Sample enhancement,Compression ratio

到稿日期:2023-10-30　返修日期:2024-04-16
基金项目:国家重点研发项目(2022YFB3103402);国家自然科学基金(62072200,62172176,62127808)
This work was supported by the National Key Research and Development Program of China(2022YFB3103402) and National Natural Science Foundation of China(62072200,62172176,62127808).
通信作者:韩兰胜(hanlansheng@hust.edu.cn)

1 引言

渗透测试指模拟真实攻击者的行为，发掘系统漏洞，从而评估系统安全性的方法。通过渗透测试，不仅可以发现目标系统存在的安全漏洞，还可以评估其安全防护能力，并进行有针对性的加固。常规渗透测试大都依赖人工操作，然而，该方法存在高度依赖专家经验和主观偏差等问题。尤其在大数据技术广泛应用的当下，大数据平台对应的渗透测试场景更加复杂多变，漏洞暴露的风险更大，对测试人员的要求更高，且由于其组件更多，容易出现由人为疏忽导致的漏报和误报。因此，越来越多的研究者开始关注自动化渗透测试[1-3]。自动化渗透测试可以有效降低对专家经验的依赖，减少人为错误，从而对系统安全性进行更加客观全面的评估。

自动化渗透测试早期的研究使用攻击树[4]和攻击图[5]来对系统进行结构化描述，在此基础上利用节点和连接关系搜索攻击路径，从而发现和评估系统的安全隐患。然而，构建攻击图或者攻击树面临着难以预先获取系统全部信息和大规模场景下资源消耗大的问题。强化学习是一种通过奖励引导并自动地与环境实时交互的算法，通过不断试错学习，寻找能够获取最大化累计收益的策略。强化学习可以与系统环境交互逐步获取信息，而且构建的模型规模对场景复杂度不敏感，从而避免了上述攻击树和攻击图的缺陷；同时，渗透测试也是一种通过攻击收益引导并与系统实时交互的决策过程，因此可以使用强化学习来解决自动化渗透测试问题。

强化学习算法主要分为基于表格的强化学习算法[6-7]和深度强化学习算法[8-10]。然而，基于表格的强化学习算法难以应用到大规模网络场景；深度强化学习将神经网络引入强化学习算法中，利用神经网络的特点处理复杂的非线性问题。在基于深度强化学习算法的自动化渗透测试中，由于神经网络的输入输出维度固定，需要预先定义动作的个数。对于一个包含 M 台主机的网络环境，假设有 N 种可选动作，则需要定义 $M*N$ 的输出维度。然而，$M*N$ 个动作中包含大量的无效动作，例如探索初期无法触及网络深部的主机，因此候选动作中不应包含该主机的相关动作。事实上，深度强度学习在迭代初期策略未成型时会有大量的无效动作产出，导致产生大量冗余渗透路径，并且随着网络规模扩大，冗余渗透路径的比例也会相应增加，进而影响算法性能。

强化学习中解决无效动作问题常用的方法有两种：无效动作惩罚和无效动作遮蔽[11]。前者即对模型在探索过程中选择的无效动作施加负面的反馈，以达到回避无效动作的效果。采用无效动作惩罚需要根据具体环境设计惩罚力度，并进行大量调整与训练。无效动作遮蔽则限制智能体只在有效动作列表中进行选择，从而加快模型收敛。

此外，寻找最优渗透路径的关键在于选择合适的主机与漏洞进行利用，并且由于漏洞利用动作的不确定性较大，因此对其进行准确评估更加困难。另外，由于正奖励获取动作较少，因此模型难以获取正奖励，存在稀疏奖励问题[12]。然而，模型的随机探索策略导致不同动作被采样到的概率相同，漏洞利用和正奖励获取动作样本数量不足。

本文的研究思路是通过解空间转换筛选掉大量冗余渗透路径，降低问题求解的复杂度，并通过漏洞利用样本增强方法和正奖励样本增强方法增加相关样本数量，分别强化模型对漏洞利用动作的评估，缓解稀疏奖励问题，加速模型迭代并提升其适应能力。本文的主要贡献如下：

1)对基于强化学习的智能化渗透路径规划问题进行了详尽的分析与建模，并提出解空间转换方法以降低问题求解的复杂度。

2)通过无效动作的形式化定义给出解空间转换的界限，并对无效动作的含义进行扩展，为智能体引入先验知识，从而使模型更具灵活性和可扩展性。

3)从理论层面分析解空间转换的有效性，以及对解空间转换在模型训练初期探索步数的影响进行评估，并提出压缩比来定量分析解空间转换带来的收益。

4)提出漏洞利用样本增强方法和正奖励样本增强方法，分别强化了模型对漏洞利用动作评估的稳定性和准确性，缓解了稀疏奖励问题；并提出了 MASK-SALT-DQN 算法——解空间转换和样本增强与基于深度 Q 网络(DQN)算法的渗透路径规划模型相结合。

5)对 Schwartz 等[13]设计的网络攻击模拟器 NASim 进行了改进，使其更符合真实场景；基于不同规模的模拟场景对 MASK-SALT-DQN 算法和领域先进方法进行多角度的实验，实验结果证明了冗余渗透路径问题的严重性以及 MASK-SALT-DQN 算法在不同场景下都有优秀的性能和适应能力。

2 相关工作

渗透测试过程充满不确定性，包括网络环境的多变性、攻击工具结果的不确定性以及防守方布置的防御手段[14]的多样性，因此，研究人员采用部分可观测马尔可夫决策过程(Partially Observable Markov Decision Process，POMDP)和马尔可夫决策过程对渗透测试环境进行建模。

部分可观测马尔可夫决策过程包含一个观测集，环境状态无法直接获取，而是通过条件概率获取对当前真实状态的观测，以此来模拟渗透测试过程的不确定性。Sarraute 等[3]分解网络结构，再利用 POMDP 寻找单个主机的攻击路径，然后将其组合为整个网络的攻击路径。Sarraute 等[15]给出了 POMDP 建模渗透测试过程的形式化定义，并首次将信息搜集引入渗透测试问题中，提供了能将信息搜集和漏洞利用相结合的方法，更接近现实的渗透测试情景。为了解决经典规划算法在实际应用中表现下降和 POMDP 算法不能扩展到大规模网络的问题，Shmaryahu 等[16]提出一种折中的方法，将渗透测试建模为部分可观测的偶发问题，并设计了偶发规划树算法来寻找攻击路径。Schwartz 等[14]考虑到防御方的主动防御行为，进而引入一个信息衰减因子变量来表示防御方的行为。该变量为防御方从分析网络到主动防御网络所花费的预期时间，并利用伯努利过程建模防御方的行为。实验表明该方法比基线算法具有更好的性能。

尽管 POMDP 在模拟渗透测试的不确定性方面表现出色，但其在复杂场景下求解难度极大，导致很难被应用到大规模网络场景中。因此，越来越多的研究者开始将渗透测试过程建模为马尔可夫决策过程(Markov Decision Process，

MDP),其优点在于可以降低模型的复杂度,提高决策效率。

Zennaro 等[17]关注简化的渗透测试——网络攻防竞赛(Capture The Flag,CTF),并将其建模为 MDP,使用基于表格的 Q-learning 算法解决该问题;同时,为了缓解状态空间和动作空间维度爆炸问题,提出了一种使用模仿学习来为代理提供先验知识的方法。Yousefi 等[18]使用 MulVal[19]生成攻击图,并将攻击图矩阵用于 MDP 建模,但是该方法只适用于小规模状态空间和动作空间的情况。因此,Hu 等[20]使用深度强化学习算法优化复杂场景下模型的表现,并使用深度优先搜索算法对攻击图矩阵进行简化,只保留可以到达目标的路径,基于简化的矩阵完成攻击路径发现。Zhou 等[21]将信息搜集过程加入渗透测试过程中,同时提出一种基于网络信息增益的自动化攻击规划算法,该方法利用执行动作前后的信息增益作为奖励引导智能体探索环境。实验证明,该方法在发现攻击路径的有效性上有所提升,并且缩短了训练时间。Schwartz 等[14]设计了一个开源的网络攻击模拟器 NASim,该模拟器能够通过预先定义的配置文件模拟真实网络环境与渗透动作。基于 NASim 模拟环境构建 MDP 模型用于强化学习算法可以降低自动化渗透测试研究的复杂度和成本,因此,后续许多研究都基于 NASim 展开。Zhou 等[22]提出一种名为 NDSPI-DQN 的深度 Q 网络,该方法整合了包括噪声网络、软 Q 学习、决斗架构、优先经验回放和内在好奇心模型 5 个不同的扩展,以提高模型探索效率;此外还提出了动作解耦,将动作分解为主机和操作,然后使用神经网络分别计算动作的两个部分,以减小动作空间。因此,该方法具有更好的有效性和扩展性。为了解决大规模场景下动作维度过大的问题,Nguyen 等[23]提出一种基于 Wolpertinger architect 架构的多层级动作嵌入的方法,通过嵌入将动作表示为 n 维向量的形式,同时还可以保持动作的性质和关系。相比基线算法,该方法具有更高的敏感主机攻击成功率。Sultana 等[24]基于 NASim 模拟器构建了 5 个不同拓扑、配置等信息的网络,并测试了各种深度强化学习算法的稳定性和泛化性。实验结果表明,DQN 和近端策略优化算法的回报比较稳定,然而应用于与智能体训练不同的场景时,两种算法的泛化能力明显不足。Li 等[25]提出一种分层强化学习架构来解决自动化渗透测试问题,并引入专家先验知识来指导代理的决策,从而缓解无效探索的问题。

NDSPI-DQN 算法提出了动作解耦,一定程度上缓解了动作空间扩大导致的复杂度增加的问题,但是模型输出中仍包含大量无效动作组合。Li 等[25]提出引入专家先验知识的方法来缓解无效探索的问题,但是该方法没有将信息搜集相关动作引入动作空间中以及缺乏对渗透测试过程不确定性的考虑,并且依赖专家经验给出动作建议,限制了模型的探索能力。本文将渗透路径规划过程建模为 MDP 过程,使用 NASim 模拟具体场景,所提出的 MASK-SALT-DQN 算法通过解空间转换极大地降低了问题的求解复杂度;此外,还提出了漏洞利用样本增强方法和正奖励样本增强方法分别强化模型对漏洞利用动作评估的稳定性和准确性,以缓解稀疏奖励问题,优化模型训练过程。

3 问题分析与建模

3.1 问题描述与分析

智能化渗透路径规划的目标是解决如何以最小的代价完成渗透目标的问题。为了完成这一目标,代理需要从初始状态开始,根据当前观测到的环境状态评估动作空间中所有动作的好坏,并选择合适的动作与系统环境交互,然后根据环境的反馈不断调整策略,最终达到可以规划出最优渗透路径的目的。

智能化渗透路径规划过程中存在动作评估不充分和稀疏奖励的问题,并且由于存在冗余渗透路径,使得问题的求解空间过大:一方面,由于动作需要预先设定,然而并非所有动作都能改变当前的状态,智能体可能执行无法改变环境状态的动作,而包含这种动作的渗透路径显然不是最优解,因此会产生冗余渗透路径,导致问题的解空间增大;另一方面,漏洞利用动作执行结果的不确定性较大,使得漏洞利用动作评估的准确性和稳定性较差。此外,由于只有敏感主机设置主机价值,环境中的正向奖励十分稀疏,因此智能体的探索比较困难。然而,如果对过多的主机设置奖励,模型会倾向于获取全部的奖励以致于执行了不必要的动作,从而改变了模型的最优策略。

3.2 问题建模

渗透路径规划需要进行收益评估和智能决策,而马尔可夫决策过程模型能够实现交互的量化和可计算化,因此根据马尔可夫决策过程模型,将渗透路径规划过程用四元组〈S,A,R,P〉表示并进行详细的描述。

1)状态空间 S。状态空间由系统中全部主机的状态信息组成,单个主机的状态包含 3 个部分,分别是地址信息、配置信息和辅助信息。地址信息格式为〈subnetAddr,hostAddr〉,其中该主机所在的网段编号是 subnetAddr,该主机所在子网中的编号是 hostAddr;配置信息包括主机运行的操作系统、服务以及进程信息;辅助信息描述该主机是否可访问、是否被发现、是否被攻陷以及访问权限。单个主机的状态通过 one-hot 方式编码,记为 s_i,$1\leqslant i\leqslant N$,N 为主机数量,而智能体观测得到的状态 $s\in S$ 是当前环境中所有可见主机状态编码的连接,表示为 $s=s_{v_1}\oplus s_{v_2}\oplus\cdots\oplus s_{v_n}$;$v_j\in V$,$1\leqslant j\leqslant n$。$V$ 表示可见主机的编号集合,大小为 n。

2)动作空间 A。为了符合真实的渗透测试场景,智能体最初并不知晓目标环境的全部信息,而是需要与目标系统进行一系列操作交互获取信息。动作空间描述了所有可能的操作。动作包含动作类型和动作目标,表示为〈target,type〉,其中,target 为目标主机的地址信息,type 是动作类型,分为信息搜集、漏洞利用和权限提升。信息搜集包含操作系统扫描、服务扫描、进程扫描和子网扫描。漏洞利用和权限提升的具体动作需要指定漏洞,因此其动作个数与目标系统中相应的漏洞数量一致。综上,一个包含 N 个主机、存在 M 个服务漏洞和 K 个权限提升漏洞的环境,动作空间的大小为 $N*(M+K+4)$,其中 4 表示信息搜集的 4 种可能动作。

3)奖励函数 R。不同的主机对渗透测试目标的贡献程度不一,因此每个主机都会设置一个主机价值,当智能体获取

主机的 root 权限时，可以获取与主机价值相应的奖励。根据主机价值将目标环境中的主机分为两种：敏感主机和非敏感主机。对敏感主机设置一定的奖励值，而主机价值小的非敏感主机，奖励值则设为 0。此外，对主机采取的动作存在时间、金钱等成本因素，因此对每个动作设置相应的代价，一方面，这样更符合真实场景；另一方面，可以防止智能体浪费探索的步数，从而寻找到最优的策略。通过奖励和代价可以对整个探索过程进行评估，即累积每个主机提供的奖励与每一步动作的代价的差值，如式(1)所示：

$$R=\sum_{h\in H} value(h)-\sum_{a\in A} cost(a) \tag{1}$$

其中，H 为探索路径上的主机集合，A 为探索中执行的动作集合，$value(h)$为攻陷主机后获取的奖励，$cost(a)$为执行动作的代价。

4)状态转移函数 P。不同的动作会产生不同的结果，可能导致智能体观测得到的状态发生变化。例如，通过服务扫描可以增加对目标主机运行服务的观测。观测的变化就是状态之间的转移，转移受执行的动作、动作成功率以及目标系统的防御措施等因素的影响，例如漏洞利用的成功率根据 CVSS 的漏洞利用难度的不同而不同[26]。因此，状态转移函数获取当前状态 s 和智能体执行的动作 a，输出状态转移 T 中的一种可能 s_{t_j}，其中 $T=[(s_{t_1},p_1),(s_{t_2},p_2),\cdots,(s_{t_m},p_m)]$，$s_{t_j}\in S$，$p_j$ 表示转移到状态 s_{t_j} 的概率，$\sum_{1\leqslant j\leqslant m} p_j=1$。

4 模型与分析

首先给出解空间转换的方法来简化问题求解复杂度，然后给出解空间转换的实现和收益评估，最后提出样本增强方法来优化模型的训练过程。

4.1 解空间转换

智能化渗透路径规划问题的原始解空间记为 $\varphi=(PATH,R,P)$，其中 $PATH$ 为渗透路径集合；R 为奖励函数，用来评估路径的好坏；P 为状态转移函数，用来约束路径中状态的转移关系。智能体从初始状态开始执行动作并转移到新状态，然后不断迭代直至达到目标状态。该过程得到的转移序列即为一条渗透路径，记为 $path=(s_0,a_0,s_1,\cdots,s_{n-1},a_{n-1},s_n)$，其中 s_0 为初始状态，s_n 为目标状态。所有的渗透路径构成集合 $PATH$，对于任何一个包含 n 个状态的路径，任意 s_i,a_i,s_{i+1} 满足 $P(s_i,a_i)=s_{i+1}$，且 s_n 为目标状态。根据奖励函数 R 可以评估不同渗透路径的优劣，从而找到最优渗透路径。

最优渗透路径需要以最小的代价到达目标状态，然而在状态 s 下执行某些动作并不会改变当前状态，对到达目标状态没有帮助，这些动作被称为无效动作。虽然执行这些动作不影响渗透路径的生成，但会使渗透路径包含多余的部分。如果一条渗透路径包含无效动作，则该路径是冗余渗透路径。根据上述定义，对于一条冗余渗透路径，一定可以找到转移 (s_i,a_i,s_{i+1})，其中 $s_i=s_{i+1}$，故可以得到一条更短的渗透路径 $(s_0,a_0,s_1,\cdots,s_i,a_{i+1},s_{i+2},\cdots,s_{n-1},a_{n-1},s_n)$。因此，冗余渗透路径显然不是最优渗透路径，可以将这些渗透路径剔除以降低问题求解的复杂度。记简化后的解空间 $\phi=(PATH',R,P)$为目标解空间，其中 $PATH'$ 为剔除冗余渗透路径后的渗透路径集合。从原始解空间到目标解空间的转换即是对问题的简化，这一过程通过解空间转换 π 来完成，如式(2)所示：

$$\pi(\varphi)=\phi \tag{2}$$

然而解空间转换并非无限制的的转换，从上述解空间转换的描述中可以看出，转换的关键在于找出每个状态下的无效动作，因此，通过无效动作来限定解空间转换的界限。一个状态下的无效动作指执行结果不会使状态发生转移的动作，如式(3)所示：

$$f_{\text{trans}}(s,a)\equiv s \tag{3}$$

其中，f_{trans}表示状态转移函数，s 表示状态，a 表示无效动作。需要注意的是，无效动作 a 不能脱离状态 s 来定义，因为不同状态下动作的无效性是不同的。

下面给出得到无效动作的限制条件，并将这种限制分为 3 类来考虑。1)目标限制。若目标还未被发现，则对该目标采取的所有动作都是无效动作。2)依赖限制。在单个主机的渗透过程中采取的动作是存在依赖的，一些动作执行成功的结果是某个动作的前置条件，因此，在这些动作完成之前，该动作是无效的。根据动作的依赖关系将单个主机的渗透测试过程划分为 3 个阶段，分别是信息搜集阶段、漏洞利用阶段和后渗透阶段。信息搜集阶段包含的动作类型为服务扫描和操作系统扫描；漏洞利用阶段包含的动作类型为漏洞利用；后渗透阶段包含的动作类型为进程扫描、权限提升和子网扫描。例如，需要先执行服务扫描动作获取一个主机运行的服务，才可以对主机执行服务相关的漏洞利用动作。此外，由于漏洞利用动作需要指定具体的漏洞，然而对于一个主机而言，并非包含环境中所有的漏洞，因此只有主机上包含的漏洞相应的漏洞利用动作才是有效的。同样，主机运行的进程和操作系统符合权限提升动作的条件时，该动作才是有效的。3)信息增益限制。智能体在与环境交互的过程中，可以观测到环境的状态和动作执行的结果，若动作执行成功，则后续该动作无法带来更多收益，应变为无效动作。

通过上述对动作的 3 种限制可以得到一个判断状态 s 下动作是否为无效动作的预言机 $Oracle$，如式(4)所示：

$$Oracle(a_i\mid s_0,a_0,s_1,\cdots,s_{i-1},a_{i-1},s_i)=\begin{cases}1, & a_i \text{ 为无效动作}\\ 0, & a_i \text{ 为有效动作}\end{cases} \tag{4}$$

其中，输出 1 代表动作为无效动作，输出 0 代表动作为有效动作。在生成渗透路径的每一步时，智能体首先向 $Oracle$ 查询当前状态下不同动作的有效性，得到无效动作集合 A'，然后将该集合从动作空间 A 中剔除，并在剩余的有效动作空间中选择动作，直至到达目标状态。通过该过程可以确保没有无效动作被选择，进而达到剔除冗余渗透路径的效果。

此外，为了进一步简化问题的原始解空间，对无效动作的定义做一定的扩展，使得其涵盖范围更广。智能体的目标是以尽可能少的步数获得敏感主机的 root 权限，对于非敏感主机应以攻陷主机为目标而不必获取其 root 权限。若目标主机不是敏感主机，则该主机相关的进程扫描和权限提升操作都是无效动作。因此，扩展后的无效动作并非只包含不会成功的动作，还需要根据智能体目标以及渗透测试经验加入一些限制，从而使智能体获取任务相关的

先验知识，进一步提升其性能。

4.2 求解简化与收益评估

DQN 算法根据当前状态评估动作列表中每个动作执行后所能带来的未来累计收益的期望来选择下一步动作。DQN 算法包含 eval_network 和 target_network 两个网络。智能体首先观测到状态 s 并根据 eval_network 计算每个动作的未来收益期望，即 Q 值，然后根据 ε 贪婪规则选取动作 a 执行，获取奖励 r 并观测到新状态 s'，不断迭代该过程得到一系列四元组(s,a,r,s')，并存放到经验回放池中。模型每轮从经验回放池中选取一组四元组，根据均方误差计算 $loss$。对于选取的每个四元组(s,a,r,s')，target_network 在 s' 下计算每个动作的值，并选择一个最大的 Q 值用于计算目标 Q 值，且每隔一定步数将 eval_network 的参数复制给 target_network。DQN 的训练流程如图 1 所示。

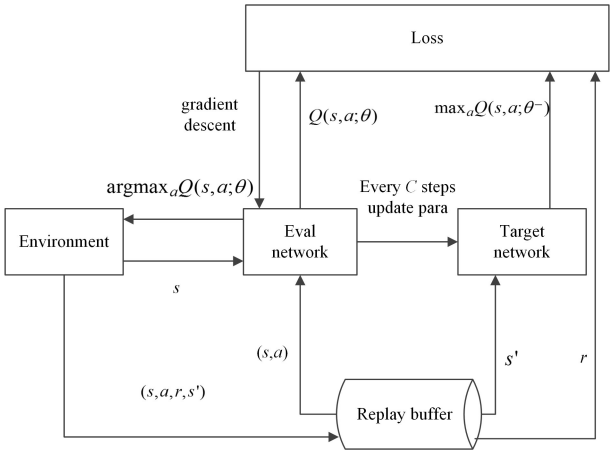


图 1 DQN 训练流程

Fig. 1 DQN training process

目标 Q 值的计算式如式(5)所示：

$$y_i = r + \gamma \max_{a'} Q(s',a';\theta_i^-) \tag{5}$$

loss 的计算式如式(6)所示：

$$L_i(\theta_i) = E_{(s,a,r,s')\sim D}[(r+\gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i))\nabla_{\theta_i} Q(s,a;\theta_i)] \tag{6}$$

其中，D 为经验回放池；θ_i 为 eval_network 的参数；y_i 为目标 Q 值；γ 为介于 0～1 之间的折扣因子，用来平衡即时奖励和未来奖励的重要性。

然后采用梯度下降法进行梯度更新，loss 的梯度公式如式(7)所示：

$$\nabla_{\theta_i} L_i(\theta_i) = E_{(s,a,r,s')\sim D}[(r+\gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i))\nabla_{\theta_i} Q(s,a;\theta_i)] \tag{7}$$

基于无效动作遮蔽的解空间转换通过修改上述模型中的动作选取流程实现。其中，eval_network 中的无效动作遮蔽分为两种情况，如图 2 所示，在第一种情况下，模型以 ε 的概率随机选择动作，为了排除无效动作的选择，首先需要根据当前状态获取有效动作列表，然后将选择范围限定在该列表中；在第二种情况下，模型以 1−ε 的概率选取最大 Q 值的动作，需要根据当前状态 s 获取无效动作列表，然后将无效动作的 Q 值替换为极小值 M(-inf)，再让模型选取最大 Q 值的动作。而 target_network 总是按照上述第二种情况实现遮蔽。

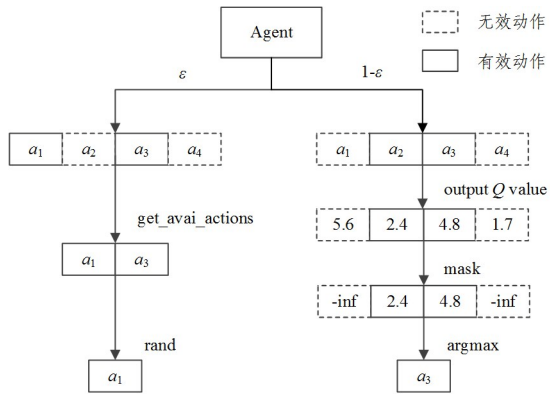


图 2 基于无效动作遮蔽的智能体动作选择

Fig. 2 Agent action selection based on invalid action masking

接下来分析上述模型处理中解空间转换的有效性。在 DQN 训练的每一步过程中，上述解空间转换流程覆盖了选择动作的所有情况，确保模型不会选择到无效动作。从式(6)描述的损失函数中可以看出，参与计算损失的分别为 eval_network 在当前状态 s 下选择的动作 a 的 Q 值和 target_network 在转移后的状态 s' 下选择的动作 a' 的 Q 值。由于进行了解空间转换，动作 a 和 a' 都是有效动作，因此，计算得到的 loss 及其梯度都不受无效动作的影响，即无效动作不会参与到模型的迭代更新中去。综上所述，本文提出的解空间转换方法既保证了智能体在探索过程中不会选择执行无效动作，同时又不会影响模型的更新过程。

解空间转换处理的有效性得以验证，但是解空间转换具体如何影响模型探索以及为模型带来多少收益，还需要进行深入的研究。解空间转换直接改变了经验回放池的组成，从而影响了模型训练使用的数据。Schaul 等[27]在先前的研究工作中探索了训练数据对模型的影响，提出使用 TD-误差(TD-error)来衡量经验池中数据的好坏。TD-error 是当前状态下选择的动作的 Q 值和目标 Q 值的差值，代表了模型对收益评估的不准确程度。TD-error 绝对值越大表示模型对当前数据的评估越不准确，可以从该数据中习得的经验就越多。因此，TD-error 绝对值大的数据越容易被选中，模型的训练速度越快。在渗透路径规划的模型训练过程中，能获得的正奖励是稀疏的，比如只有成功获取敏感主机 root 权限的动作才能获得较大的收益。在探索初期时，模型得出的 Q 值往往较小，只有在能够获取正奖励的情况下 TD-error 绝对值才较大。解空间转换改变了经验回放池的数据分布，增大了 TD-error 绝对值较大的数据被选取的概率，从而加快了模型学习的速度。

根据上述说明可知，在模型探索过程中，第一次完成目标获取的正奖励使得该次转移的 TD-error 很大，从而显著影响了模型的调整。解空间转换可以使得模型在选择动作时只在有效动作候选列表中选取，而有效动作相比无效动作有更大的概率使状态向目标状态转移，因此，遮蔽后的模型在探索初期可以在更少的步数内完成目标。令包含解空间转换处理的模型和一般模型分别为 M_1，M_2，假设 M_1 需要平均探索 n 步来初次达到目标状态，记 M_1 在探索第 i 步时的无效动作比例为 β_i，$0 < \beta_i < 1$，那么用随机变量 X_i 描述 M_2 相较于 M_1 的

这一阶段探索到有效动作所需步数。X_i 的分布函数 $F(X_i)$ 如式(8)所示：

$$F(x)=P\{X_i\leqslant x\}=\sum_{j=1}^{x}\beta_i^{j-1}(1-\beta_i) \tag{8}$$

则 M_2 第一次探索到目标状态所需的步数 sum 是 $X_1, X_2, \cdots, X_n$ 的和，如式(9)所示。用 sum 的期望 E_{step} 表示 M_2 第一次探索成功所需的平均步数，期望计算式如式(10)所示。

$$sum=\sum_{1\leqslant i\leqslant n}X_i \tag{9}$$

$$E_{step}=E(sum)=\sum_{1\leqslant i\leqslant n}E(X_i)=\sum_{1\leqslant i\leqslant n}\frac{1}{1-\beta_i} \tag{10}$$

可以看出，在 M_1 探索初期，平均 n 步到达目标状态时，没有进行解空间转换的 M_2 需要花费远超 n 的 E_{step} 步，因此，解空间转换实际上压缩了第一次成功达成目标所需的步数。为了量化解空间转换的效果，使用两个模型初次达成目标的期望步数计算压缩比 η，如式(11)所示：

$$\eta=\frac{E_{step}}{n} \tag{11}$$

压缩比越大，说明解空间转换使智能体第一次探索成功越快，模型调整速度也就越快。而且，随着网络规模的扩大，探索初期无效动作的比例 β_i 越大，因此压缩比越大。

然而，在实际模型训练过程中，每一回合的探索存在步数限制，记为 N，当模型探索步数超出 N 时，会重新开始探索。因此，需要对 sum 进行一定的修正，假设 $sum<N$ 的概率为 p，则模型探索的回合数 S 的分布函数 $R(S)$ 如式(12)所示：

$$R(s)=P\{S\leqslant s\}=\sum_{1\leqslant j\leqslant s}(1-p)^{j-1}p \tag{12}$$

当探索回合数为 s 时，探索步数应为 $N*(s-1)+m'$，其中，m' 与 sum 有相同的分布，但是限制 $m'<N$。综上，修正后的 M_2 第一次探索完成目标的所需步数的期望 E'_{step} 由式(13)计算得出。

$$\begin{aligned}E'_{step}&=E_s(N*(s-1))+E(m')\\&=\sum_{s=1}^{\infty}P\{S=s\}*(s-1)+E(m')\\&=\frac{1-p}{p}N+E(m')\end{aligned} \tag{13}$$

4.3 样本增强

对不同漏洞的准确评估是寻找最优渗透路径的关键。同一时刻，智能体有多个可选的攻击目标和漏洞，且漏洞利用动作执行结果的不确定性较大，智能体需要更多的样本数据才能准确地评估并选择合适的漏洞利用动作。然而，在模型训练时，尤其在训练初期，动作的选择主要依赖模型随机探索，且不同动作被选择的概率是相同的，因此，产生的漏洞利用动作相关的样本数量并没有优势。此外，随着网络规模的扩大，网络深部的主机被探索到的概率越来越小，目标的漏洞利用相关样本更加匮乏，对相应动作的评估更加不足，导致模型收敛过程不够稳定。因此，模型需要较多的训练步数才能收敛到最优渗透路径，可以通过改善经验回放池中数据组成的方式来优化求解的过程。

通过提出漏洞利用样本增强的方法，来增加模型训练期间漏洞利用动作相关样本数据的生成。当模型选择到漏洞利用类型动作时，智能体首先获取当前所有可对动作目标执行的漏洞利用动作，然后重复执行这些动作若干次。同时，为了防止生成过多的漏洞利用动作相关样本，使得其他动作相关样本被抽样到的概率减小，并进一步影响对这些动作的评估和模型的收敛，需要设置一定的阈值。当模型的探索率小于该阈值后，就不再进行漏洞利用样本增强，并且可以减少样本的生成数量，缩短训练时间。此外，由于网络深部的主机的漏洞利用样本相对更加匮乏，为了防止不同主机的漏洞利用样本比例失衡，需要更加注重对网络深部主机的漏洞利用样本增强。

算法实现流程如算法 1 所示，其中 δ 为设定的阈值。该方法一方面保证了随机探索阶段不同动作被选择的概率仍然是相同的；另一方面，增加了漏洞利用相关动作的样本数量，且主机所有可能的漏洞利用动作均被考虑在内，使得智能体对不同漏洞利用动作的评估更加充分、准确。

算法 1 漏洞利用样本增强算法

输入：状态 s，动作 a，探索率 ε，漏洞利用样本增强阈值 δ，增强轮数iter

输出：漏洞利用样本增强生成的若干四元组($s, \hat{a}, \hat{r}, \hat{s}$)

```
1. if ε>δ and is_exploit(a) then / * 符合阈值要求且动作为漏洞利用
   动作 * /
2.    avai_action←get_avai_actions(s) / * 获取有效动作 * /
3.    exploits←get_all_exploit(a,avai_action) / * 获取动作目标所有
      可利用漏洞利用动作 * /
4.    for i=1,iter do
5.       for e=1,len(exploits) do
6.          â←exploits_e / * 获取漏洞利用动作
7.          执行动作 â并观测到奖励 r̂和下一个状态 ŝ
8.       存储转移 (s,â,r̂,ŝ)到经验回放池 D
9.       重设当前状态为 s
10.      end for
11.   end for
12. end if
```

由于渗透路径规划过程重点关注敏感主机，只有获取敏感主机 root 权限时才能获得较大的正奖励，智能体可以获得的正奖励是十分稀疏的，因此，除了进行漏洞利用样本增强，还需要进行正奖励样本增强来增加经验回放池中正奖励相关样本的数量，从而达到加速模型收敛的目的。具体实现方法为，当模型获取正奖励时，重复执行该动作若干次并将样本存入经验回放池中。同样地，需要为该过程设置相应的阈值。算法实现流程如算法 2 所示。

算法 2 正奖励样本增强算法

输入：四元组(s,a,r,s′)，探索率 ε，正奖励样本增强阈值 δ，增强轮数 iter

输出：正奖励样本增强生成的若干四元组(s,a,r,$\hat{s}$)

```
1. if ε>δ and r > 0 then
2.    for i=1,iter do
3.       将当前状态重设为 s
4.       执行动作 a 并观测到奖励 r 和下一个状态 ŝ
5.       存储四元组(s,a,r,ŝ)到经验回放池 D
6.    end for
7. 将下一个状态设置为 s′
```

在模型训练过程中，增强上述相关动作样本只会影响到模型对其收益的评估准确性，而不会影响到动作收益本身。这是由于动作成功率固定，且动作的影响只分为成功和失败两种情况，因此样本增强不会改变相关动作正负样本的比例，也就不会影响到模型收敛的方向，即最优策略不变。

综上，结合解空间转换和样本增强，MASK-SALT-DQN 算法流程如算法 3 所示。其中，mask 将无效动作的 Q 值替换为一个极小值(-inf)，VEST 为漏洞利用样本增强算法，PEST 为正奖励样本增强算法。

算法 3 MASK-SALT-DQN 算法

输入：经验回放池容量 N，训练回合数 M，每回合训练步数 T，探索率 ε，目标网络更新频率 C，样本增强参数(ε_1，δ_1，ε_2，δ_2)

输出：动作价值函数 Q，目标动作价值函数 $\hat{Q}$

1. 将经验回放池 D 初始化为容量 N
2. 使用随机权重 θ 初始化动作价值函数 Q
3. 使用权重 $\theta^- = \theta$ 初始化目标动作价值函数 $\hat{Q}$
4. 初始化环境 env
5. for episode=1，M do
6. $s_0 \leftarrow$ env. reset()/ * 将环境重设为初始状态
7. for t=1，T do
8. $$a_t \leftarrow \begin{cases} \text{rand}(\text{get\_avai\_action}(s_t)) & \text{(a)} \\ \text{argmax}_a \text{mask}(Q(s_t, a; \theta)) & \text{(b)} \end{cases}$$
/ * 以概率 ε 按式(a)随机选取动作，否则按照式(b)选择动作 * /
9. VEST(s_t，a_t，ε，δ_1，$iter_1$)/ * 漏洞利用样本增强 * /
10. 执行动作 a_t 并观测到奖励 r_t 和下一个状态 s_{t+1}
11. 存储转移(s_t，a_t，r_t，s_{t+1}) 到经验回放池 D
12. PEST(s_t，a_t，r_t，s_{t+1}，ε，δ_2，$iter_2$)/ * 正奖励样本增强 * /
13. 从经验回放池 D 中随机抽样小批量样本(s_j，a_j，r_j，s_{j+1})
14. $$y_j = \begin{cases} r_j & \text{(a)} \\ r_j + \gamma \max\limits_{a'} \text{mask}(\hat{Q}(s_{j+1}, a'; \theta^-)) & \text{(b)} \end{cases}$$
/ * 如果第 j+1 步回合结束，按式(a)计算 y_j，否则按式(b)计算 y_j * /
15. 使用式(7)对参数 θ 执行一次梯度更新
16. 每 C 步更新 $\hat{Q} = Q$
17. end for
18. end for

5 实验与结果分析

通过实验验证本文方法的有效性。首先在标准场景下验证其有效性，然后扩大实验场景的规模，验证其可扩展性，最后根据实验结果进行分析。

5.1 实验设置

使用网络攻击模拟器 NASim[13] 来模拟系统环境，NASim 可以根据配置文件生成模拟环境，用于人工智能算法的训练。然而该模拟器虽然将服务扫描等信息搜集动作纳入智能体动作列表中，但是在与模拟环境交互的过程中，NASim 使各主机的操作系统、服务、进程等信息包含在智能体观测到的初始状态中，导致智能体不必执行信息搜集动作就能获取主机配置信息。因此，为了更贴合真实场景，对该模拟器进行修改，使环境中所有主机的配置信息不包含在初始状态中，而是需要在智能体与环境的交互过程中探索获取。

首先，使用企业常见的网络拓扑结构作为标准环境(场景 1)，如图 3 所示，该环境包含 5 个子网、16 台主机，其中，子网 1－2 为 DMZ 区，子网 3－5 为内网；并通过防火墙控制不同子网之间可以访问的服务，例如图中子网 1 可以访问子网 2 中的 SMTP 服务，子网 2 可以访问子网 1 中的 SSH 服务，而子网 1 不能直接与子网 3 通信。各个主机的详细信息如表 1 所列，其中敏感主机的价值为 100，非敏感主机的价值为 0。

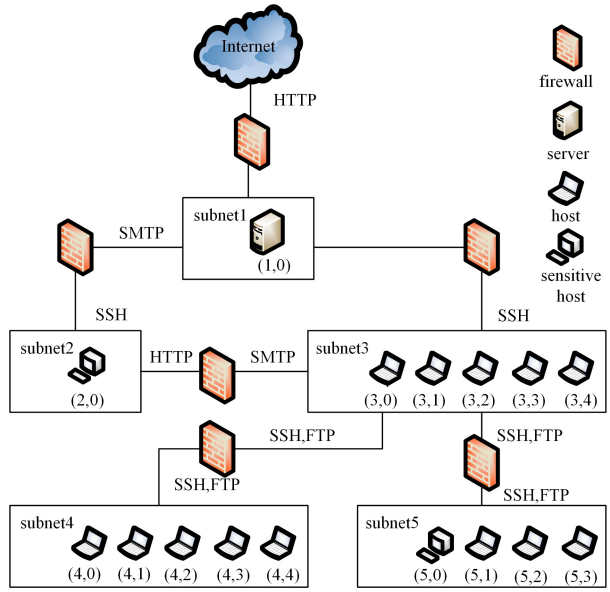


图 3 场景 1 网络拓扑图

Fig. 3 Network topology of scenario 1

表 1 场景 1 中各主机详细配置信息

Table 1 Detailed configuration information of each host in scenario 1

地址	操作系统	服务	进程	主机价值
(1,0)	Linux	HTTP	—	0
(2,0)	Windows	SMTP	Schtask	100
(3,0)	Windows	FTP	Schtask	0
(3,1)	Windows	FTP,HTTP	Daclsvc	0
(3,2)	Windows	FTP	—	0
(3,3)	Windows	FTP	Schtask	0
(3,4)	Windows	FTP	Schtask	0
(4,0)	Linux	SSH	—	0
(4,1)	Linux	SSH	—	0
(4,2)	Linux	SSH	—	0
(4,3)	Windows	SSH,FTP	Tomcat	0
(4,4)	Windows	SSH,FTP	Tomcat	0
(5,0)	Linux	SSH,SAMBA	—	100
(5,1)	Linux	SSH,HTTP	Tomcat	0
(5,2)	Linux	SSH	—	0
(5,3)	Linux	SSH	—	0

表 2 列出了该环境下智能体可以采取的动作，其中包含 4 种扫描动作，以及针对各种服务的漏洞利用动作和针对各种进程的权限提升动作；同时还给出了动作目标操作系统限制、动作执行代价、动作执行成功率和动作执行成功后获得目标主机的权限等信息，其中“－”代表没有相应信息。漏洞利用动作的成功率根据 CVSS 漏洞利用难度高中低分别设置为 0.2，0.5，0.8，其他动作的成功率设置为 1。

表 2 场景 1 中智能体的动作列表

Table 2 Actions of agent in scenario 1

名称	操作系统	代价	成功率	权限
Service-Scan	—	1	1	—
Os-Scan	—	1	1	—
Process-Scan	—	1	1	—
Subnet-Scan	—	1	1	—
SSH-Exp	Linux	3	0.8	User
FTP-Exp	Windows	1	0.5	Root
HTTP-Exp	—	2	0.8	User
SMB-Exp	Windows	2	0.2	Root
SMTP-Exp	Windows	3	0.5	User
Tomcat-PE	Linux	1	1	Root
Daclsvc-PE	Windows	1	1	Root
Schtask-PE	Windows	1	1	Root

为了验证算法的可扩展性，还需要扩大网络规模，增加主机、服务、进程的数量，然后使用 NASim 根据配置随机生成相应的漏洞利用和权限提升动作。同时，敏感主机的数量固定为 2。随着网络规模的增大，奖励将会越来越稀疏，同时动作空间和状态空间会快速增大，从而使得环境的探索更加困难。表 3 列出了增加的 3 个扩展场景的详细信息。

表 3 各场景的详细信息

Table 3 Detailed information of each scenario

场景	主机	子网	服务	漏洞	进程	权限提升
场景 2	50	12	10	20	3	3
场景 3	100	21	10	20	3	5
场景 4	150	31	10	20	4	5

实验选取 DQN 算法和 NDSPI-DQN 算法作为基线算法。DQN 是自动化渗透测试领域中广泛使用的深度强化学习算法；NDSPI 基于 DQN 算法引入了噪声网络、软 Q 学习等 5 种不同的扩展，并提出动作解耦的方法来缓解动作空间过大的问题，取得了不错的效果。记不加入样本增强的算法为 MASK-DQN，通过比较 MASK-DQN、MASK-SALT-DQN 和两种基线算法在不同场景下的收敛过程，验证了解空间转换的必要性和对模型的优化效果。

算法基于 python 3.9.16 下的 pytorch 框架开发，操作系统为 Windows10，硬件配置为 AMD Ryzen 7 5800H CPU、NVIDIA GeForce GTX 1650 GPU。实验的超参数如表 4 所列。

表 4 超参数设置

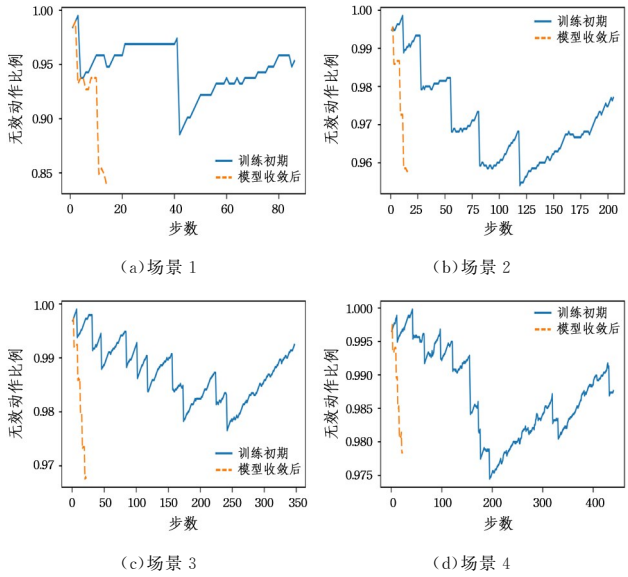
Table 4 Hyperparameter settings

参数名称	参数含义	参数值
batch size	批次大小	64
learning rate	学习率	0.0005
discountfactor, γ	折扣因子	0.9
replay memory size	经验回放池大小	100000
hidden size	隐藏层大小	[256,256]
max steps per episode	每回合最大步数	2000
target network update frequency	目标网络更新频率	1000

5.2 实验结果与分析

5.2.1 无效动作比例

由于无效动作比例比较容易获得，而且可以用无效动作比例来估算冗余渗透路径比例的下界，因此分别统计本文模型在训练初期和模型收敛后一个回合中无效动作占全部动作的比例，结果如图 4 所示。



(a)场景 1 (b)场景 2

(c)场景 3 (d)场景 4

图 4 无效动作比例在训练初期和模型收敛后一个回合内的变化情况

Fig. 4 Changes in the proportion of invalid actions in one round at the beginning of training and after model converges

从图 4(a)中可以看出，对于场景 1 而言，无论是训练初期还是模型收敛之后，无效动作的比例都会经历两次大幅度的下降，分别发生在发现子网 2,3 和发现子网 4,5 的时间节点。子网发现导致观测到的主机增加，使一些动作从无效变为有效，因此，无效动作比例下降。然而不同的是，训练初期，智能体在每次发现主机之后会进行充分的探索；而模型收敛之后，智能体倾向于选择最优动作。由于探索过的动作会变为无效动作，因此相较于训练初期，模型收敛后发现主机只会导致无效动作的比例在一个小范围内浮动。

从图 4(b)—4(d)可以看出，在场景 2—场景 4 下，训练初期无效动作比例大幅度下降的次数多于模型收敛后无效动作比例下降的次数。根据前面的分析，无效动作的比例大幅下降说明智能体通过子网扫描发现了更多的主机，模型收敛后，智能体只对最优攻击路径上的主机进行攻击，而训练初期智能体会更多地对网络中的子网进行探索，因此，两条曲线大幅下降的次数会存在差异。

最后，从整体上来看，对于 4 个场景，无效动作比例的初始值分别为 98.4%(场景 1)、99.5%(场景 2)、99.7%(场景 3)、99.7%(场景 4)，并始终保持在较高的水平(>83%)，而且随着网络规模的扩大，无效动作的比例在一回合内的下降越来越有限。因此，在智能体与环境交互的过程中，无效动作问题十分严重，并且随着网络规模的扩大逐渐加重。

5.2.2 环境探索与奖励

通过记录不同模型在 4 个场景中第一次获得正奖励的步数、第一次完成目标(获取全部敏感主机的 root 权限)的步数、前十个回合平均的步数，比较本文方法和基线方法在探索性能上的差距，结果如表 5 所列。从表中可以看出，本文方法在 3 个指标上均取得了最优值。另一方面，由于样本增强算法对前十个回合的影响不大，可以看出，MASK-DQN 和 MASK-

SALT-DQN 算法在相关指标上的表现十分接近。因此，后续以 MASK-DQN 算法为例与两种基线算法进行比较。

表 5 4 个场景下不同算法的 3 个指标

Table 5 Three indicators of different algorithms in four scenarios

指标名称	算法名称	场景 1	场景 2	场景 3	场景 4
第一次正奖励所需步数	MASK-VEST-DQN	53.1	**62.1**	47.8	52.5
	MASK-DQN	49.4	62.5	59.3	**43.7**
	NDSPI-DQN	560	1561.1	140660	460110.1
	DQN	1546.4	48132.8	—	—
第一次完成目标所需步数	MASK-VEST-DQN	88.1	**196**	**325.2**	**416.2**
	MASK-DQN	**86.5**	197.4	335.5	421.8
	NDSPI-DQN	2720.7	—	—	—
	DQN	4879.1	—	—	—
前十个回合平均步数	MASK-VEST-DQN	**84.3**	**165**	**298.4**	**326.6**
	MASK-DQN	**84.3**	172.6	306	338
	NDSPI-DQN	1486	2000	2000	2000
	DQN	1845.6	2000	2000	2000

从第一项指标的数据中可以看出，MASK-DQN 在 4 个场景下均可以在 70 步内探索到第一次正奖励，而 NDSPI 和 DQN 算法获取第一次正奖励所需的步数远超过 MASK-DQN，并且在场景 3 和场景 4 下，DQN 算法几乎不能探索到正奖励。此外，可以看出，随着网络规模增大，MASK-DQN 获得第一次正奖励所需的步数变化不大，而 NDSPI 和 DQN 算法对此十分敏感。网络规模的增大意味着动作空间扩大，训练初期智能体主要依靠随机探索来选取动作。由前面的实验可知，在训练初期，无效动作的比例占动作空间的 98%以上，因此，随着网络规模的扩大，未经过无效动作处理的模型需要花费大量的步数来达成有效探索，而 MASK-DQN 排除了无效动作的干扰。由于第一个敏感主机在场景中的位置相对固定，因此 MASK-DQN 的表现相比其他两个算法更加稳定。

对于第一次完成目标步数的指标，MASK-DQN 在 4 个场景下均可以在较少的步数内完成，并且随着网络规模的增大，步数增长相对缓慢。而对于 NDSPI 和 DQN，只有在场景 1 下才能完成目标，并且步数都远超 MASK-DQN 的步数。和第一项指标的结果类似，NDSPI 和 DQN 受网络规模的影响很大。两项指标数据都说明了 MASK-DQN 在很大程度上缓解了稀疏奖励和动作空间过大导致的探索困难问题。

此外，根据第一次完成目标的实验数据，可以对式(9)—式(13)描述的压缩比进行验算。通过多次实验得到 MASK-DQN 第一次完成目标所需平均步数 n，同时根据每步无效动作的比例 β_i 和式(8)、式(9)可以得出 DQN 相应的步数 sum 的分布，然后分别计算 sum 小于最大探索步数的概率 $p=0.387$，sum 小于最大探索步数的分布部分的期望 $E(m')=1783.5$，根据式(13)可得出 $E'_{\text{step}}=4946.3$，接近 DQN 算法多次实验后得到的平均步数，计算得出理论压缩比 $\eta_{\text{theory}}=57.2$，实验压缩比 $\eta_{\text{exp}}=56.4$，通过式(14)计算误差 Δ 为 1.40%。

$$\Delta=\frac{|\eta_{\text{theory}}-\eta_{\text{exp}}|}{\eta_{\text{theory}}} \tag{14}$$

对于前十个回合平均步数，MASK-DQN 算法所需的步数在 4 个场景下均很少，分别为 84.3(场景 1)、168(场景 2)、312.3(场景 3)和 309(场景 4)，而 NDSPI 和 DQN 都只在场景 1 下的步数少于 2000，分别为 1486 和 1845.6。对于场景 2—场景 4 中探索的每个回合，NDSPI 和 DQN 均使用完了最大探索步数，因为它们无法在最大步数内完成目标。

5.2.3 收敛回合

统计 4 种方法在 4 个场景下的收敛性和收敛回合数，如图 5 所示。从图中可看出，在场景 1 下，DQN 算法每回合获得的奖励小幅度上升，但是最后并不收敛，而 NDSPI 算法可以在约 350 个回合之后收敛。由于 DQN 和 NDSPI 算法在场景 2—场景 4 下均无法完成目标，因此图 5 中只给出了场景 1 下不同算法的训练曲线。从图中可以看到本文提出的算法明显优于两种基线算法，这是由于解空间转换处理使智能体忽略了大量的冗余渗透路径，相较于其他两个算法，MASK-DQN 在整个训练过程中只需要进行少量渗透路径的评估和选择；而且从前面的实验结果分析可知，MASK-DQN 能够在训练初期以较少的步数完成目标，同时另外两种方法只能在场景 1 下才能完成目标，因此在相同的回合数下，MASK-DQN 能够更快速地调整模型，减少训练成本，从而快速收敛。

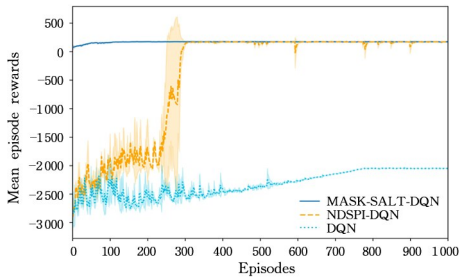


图 5 场景 1 下不同算法的每回合奖励随训练回合数变化的曲线

Fig. 5 Curve of each round reward of different algorithms changes with the number of training rounds in scenario 1

图 6 给出了 4 个场景下 MASK-DQN 和 MASK-SALT-DQN 算法的训练曲线，可以看出，MASK-SALT-DQN 算法在 4 个场景下分别在 100 回合(场景 1)、250 回合(场景 2)、300 回合(场景 3)和 350 回合(场景 4)左右收敛。在 4 个场景下，MASK-SALT-DQN 相比 MASK-DQN 算法收敛速度更快，在相同回合数下，能够获得更高的奖励，且可以在更少的回合数内收敛。图中曲线的阴影部分代表多次实验取值的标准差，因此，阴影面积越小代表模型越稳定，可以看出 MASK-SALT-DQN 算法更加稳定。最后，从训练曲线上也可以看出，MASK-SALT-DQN 算法收敛后的波动较小，奖励值偏高。

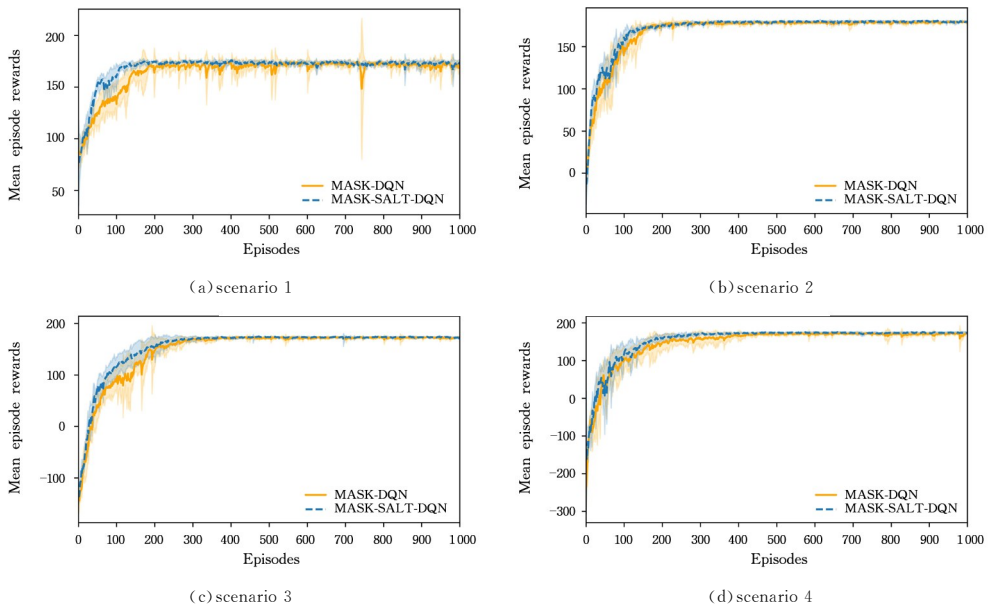


图 6 4 个场景下不同算法的每回合奖励随训练回合数变化的曲线

Fig. 6 Curve of each round reward changes with the number of training rounds of different algorithms in four scenarios

结束语 当前自动化渗透测试研究存在求解复杂度过高、部分样本评估不充分等问题，因此难以应用于目前广泛流行的大数据场景的渗透测试任务中。针对该问题，本文将渗透路径规划过程视为求解，并针对该过程中存在大量冗余渗透路径的问题进行了详尽的描述和定义，讨论了冗余渗透路径的成因及其对模型求解过程的影响，提出了基于解空间转换的 MASK-DQN 模型来解决该问题，并从理论上分析了解空间转换对模型的作用，使得对模型收益进行定量分析成为可能，并基于此提出压缩比的方法来衡量解空间转换带来的收益。此外，还提出漏洞利用样本增强方法和正奖励样本增强方法分别强化模型对漏洞利用动作的稳定性和准确性，缓解稀疏奖励问题，优化了模型训练过程。最后基于改进的 NASim 在不同的模拟场景下对 MASK-DQN、MASK-SALT-DQN 和多种基线方法进行了实验，验证了所提模型的有效性和先进性。

然而，目前的实验仍基于模拟环境，对真实场景的因素考虑欠缺，如主机的实时状态变化、防守方的防御措施等。未来可以考虑在更真实的环境中建模与实验。此外，当前的研究训练场景和验证场景需要保持一致，如何训练具有泛化能力的模型是一个值得研究的问题。

参 考 文 献

[1] CUI Y, ZHANG L J, WU H. Automatic Generation Method for Penetration Test Programs Based on attack graph[J]. Journal of Computer Applications, 2010, 30(8): 2146-2150.

[2] ZENG Q W, ZHANG G M, XING C Y, et al. Intelligent Attack Path Discovery Based on Hierarchical Reinforcement Learning [J]. Computer Science, 2023, 50(7): 308-316.

[3] SARRAUTE C, BUFFET O, HOFFMANN J. POMDPs make better hackers: Accounting for uncertainty in penetration testing [C]// Proceedings of the AAAI Conference on Artificial Intelligence. 2012, 26(1): 1816-1824.

[4] SCHNEIER B. Attack trees [J]. Dr. Dobb's Journal, 1999, 24(12): 21-29.

[5] PHILLIPS C, SWILER L P. A graph-based system for network-vulnerability analysis[C]// Proceedings of the 1998 Workshop on New Security Paradigms. 1998: 71-79.

[6] SUTTON R S, BARTO A G. Reinforcement learning: An introduction[M]. MIT press, 2018.

[7] WATKINS C J C H, DAYAN P. Q-learning[J]. Machine Learning, 1992, 8: 279-292.

[8] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Playing atari with deep reinforcement learning[C]// Neural Information Processing Systems Deep Learning Workshops. NIPS, 2013.

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

[10] MNIH V, BADIA A P, MIRZA M, et al. Asynchronous methods for deep reinforcement learning[C]// International Conference on Machine Learning. PMLR, 2016: 1928-1937.

[11] HUANG S, ONTAÑÓN S. A closer look at invalid action masking in policy gradient algorithms[C]// Proceedings of the Thirty-Fifth International Florida Artificial Intelligence Research Society Conference. FLAIRS, 2022.

[12] YANG W Y, BAI C J, CAI C, et al. Survey on Sparse Reward in Deep Reinforcement Learning [J]. Computer Science, 2020, 47(3): 182-191.

[13] JONATHON S, HANNA K. NASim: Network Attack Simulator [EB/OL]. https://networkattacksimulator. readthedocs. io/.

[14] SCHWARTZ J, KURNIAWATI H, El-MAHASSNI E. Pomdp+ information-decay: Incorporating defender's behaviour in autonomous penetration testing[C]// Proceedings of the International

Conference on Automated Planning and Scheduling. 2020：235-243.

[15] SARRAUTE C，BUFFET O，HOFFMANN J. Penetration testing＝＝POMDP solving? [C]// Working Notes for the 2011 IJCAI Workshop on Intelligent Security(SecArt). 2011.

[16] SHMARYAHU D，SHANI G，HOFFMANN J，et al. Partially observable contingent planning for penetration testing[C]// Iwaise：First International Workshop on Artificial Intelligence in Security. 2017.

[17] ZENNARO F M，ERDŐDI L. Modelling penetration testing with reinforcement learning using capture-the-flag challenges：Trade-offs between model-free learning and a priori knowledge [J]. IET Information Security，2023，17(3)：441-457.

[18] YOUSEFI M，MTETWA N，ZHANG Y，et al. A reinforcement learning approach for attack graph analysis[C]// 2018 17th IEEE International Conference On Trust，Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE). IEEE，2018：212-217.

[19] OU X，GOVINDAVAJHALA S，APPEL A W. MulVAL：A logic-based network security analyzer[C]// USENIX Security Symposium. 2005，8：113-128.

[20] HU Z，BEURAN R，TAN Y. Automated penetration testing using deep reinforcement learning[C]// 2020 IEEE European Symposium on Security and Privacy Workshops(EuroS&PW). IEEE，2020：2-10.

[21] ZHOU T，ZANG Y，ZHU J，et al. NIG-AP：a new method for automated penetration testing [J]. Frontiers of Information Technology & Electronic Engineering，2019，20(9)：1277-1288.

[22] ZHOU S，LIU J，HOU D，et al. Autonomous penetration testing based on improved deep q-network[J]. Applied Sciences，2021，11(19)：8823.

[23] NGUYEN H V，NGUYEN H N，UEHARA T. Multiple level action embedding for penetration testing[C]// Proceedings of the 4th International Conference on Future Networks and Distributed Systems. 2020：1-9.

[24] SULTANA M，TAYLOR A，LI L. Autonomous network cyber offence strategy through deep reinforcement learning[C]// Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications III. SPIE，2021：490-502.

[25] LI Q，ZHANG M，SHEN Y，et al. A Hierarchical Deep Reinforcement Learning Model with Expert Prior Knowledge for Intelligent Penetration Testing[J]. Computers & Security，2023，132：103358.

[26] BACKES M，HOFFMANN J，KÜNNEMANN R，et al. Towards automated network mitigation analysis[C]// Proceedings of the 34th ACM/SIG APP Symposium on Applied Computing. 2019：1971-1978.

[27] SCHAUL T，QUAN J，ANTONOGLOU I，et al. Prioritized experience replay[C]// International Conference on Learning Representations. ICLR，2016.

LI Cheng'en，born in 2001，postgraduate. His main research interest is cyberspace security.

HAN Lansheng，born in 1972，Ph. D，professor，Ph. D supervisor. His main research interests include network security protection，malicious code analysis and big data security.

（责任编辑：杨雪敏）