Journal of System Simulation

Volume 32 | Issue 3 Article 13

3-25-2020

A Game Cheating Detection System Based on Petri Nets

Zheng Hong
1. East China University of Science and Technology, Shanghai 200237, China;;

Ye Cheng
1. East China University of Science and Technology, Shanghai 200237, China;;

Wenxuan Deng
1. East China University of Science and Technology, Shanghai 200237, China;;

Pan Li
2. Hunan Institute of Science and Technology, Yueyang 414006, China;

Follow this and additional works at: https://dc-china-simulation.researchcommons.org/journal

Part of the Artificial Intelligence and Robotics Commons, Computer Engineering Commons, Numerical Analysis and Scientific Computing Commons, Operations Research, Systems Engineering and Industrial Engineering Commons, and the Systems Science Commons

This Paper is brought to you for free and open access by Journal of System Simulation. It has been accepted for inclusion in Journal of System Simulation by an authorized editor of Journal of System Simulation.

A Game Cheating Detection System Based on Petri Nets

Abstract

Abstract: Game cheating endangers the normal cyber game seriously. The bottlenecks of the anti-cheating technology are the low efficiency and its too much dependence on the signature matching. Not relying on the signature matching, a highly efficient game cheating detection system is proposed, which models by process trees. This can save the storage space without affecting the detection results. The system is tested with the cheating softwares on the popular cyber game and the fusion efficiency of the update model is evaluated. The results show that the storage space reduction rate can reach about 60%, which means effectiveness and feasibility of the system framework.

Keywords

petri nets, workflow nets, process trees, game cheating detection, model fusion

Recommended Citation

Zheng Hong, Ye Cheng, Deng Wenxuan, Pan Li. A Game Cheating Detection System Based on Petri Nets[J]. Journal of System Simulation, 2020, 32(3): 455-463.

This paper is available in Journal of System Simulation: https://dc-china-simulation.researchcommons.org/journal/vol32/iss3/13

第 32 卷第 3 期 系统仿真学报© Vol. 32 No. 3
2020 年 3 月 Journal of System Simulation Mar., 2020

一种基于工作流网的游戏作弊检测系统

郑红[1]，叶成[1]，邓文轩[1]，潘理[2]
（1. 华东理工大学，上海 200237；2. 湖南理工学院，湖南 岳阳 414006）

摘要：游戏作弊对网络游戏危害很大，目前反作弊技术的瓶颈是效率低并且过多依赖于签名匹配。提出一种高效且*不依赖于签名匹配的游戏作弊检测系统，用工作流网对作弊行为建模，通过流程树对作弊行为模型进行融合，在不影响检测结果的前提下节省存储空间。*用热门网游的作弊软件对系统进行实验，存储空间降低率达到约 60%，证明了该系统框架是可行、有效的。

关键词：Petri 网；工作流网；流程树；游戏作弊检测；模型融合
中图分类号：TP391.9 文献标识码：A 文章编号：1004-731X (2020) 03-0455-09
DOI: 10.16182/j.issn1004731x.joss.18-0243

A Game Cheating Detection System Based on Petri Nets

Zheng Hong[1], *Ye Cheng*[1], *Deng Wenxuan*[1], *Pan Li*[2]
(1. East China University of Science and Technology, Shanghai 200237, China;
2. Hunan Institute of Science and Technology, Yueyang 414006, China)

Abstract: Game cheating endangers the normal cyber game seriously. The bottlenecks of the anti-cheating technology are the low efficiency and its too much dependence on the signature matching. *Not relying on the signature matching, a highly efficient game cheating detection system is proposed, which models by process trees. This can save the storage space without affecting the detection results. The system is tested with the cheating softwares on the popular cyber game and the fusion efficiency of the update model is evaluated.* The results show that the storage space reduction rate can reach about 60%, which means effectiveness and feasibility of the system framework.

Keywords: petri nets; workflow nets; process trees; game cheating detection; model fusion

引言

网络游戏是一种以娱乐、休闲、交流和取得虚拟成就为目的、具有可持续性的多人在线游戏，一般由多名玩家通过计算机网络在虚拟环境下对人物角色和场景按照一定规则进行操作[1]。中国互联网络信息中心于 2017-08-04 日发布第 40 次《中国互联网络发展状况统计报告》。报告显示，截至 2017 年 6 月，我国网民规模达到 7.51 亿，互联网普及率为 54.3%。网络游戏用户规模达 56.1%[2]。

近些年中国游戏行业市场中客户端网游的增长率逐渐趋于平稳。受益于电子竞技的飞速发展，电子竞技在客户端网游的收入中占比越来越大。此外，国家对电子竞技的发展也给予了大力支持，除了将电子竞技纳入第 99 个体育项目外，还将电子竞技纳入教育部增补专业。

网络游戏是目前最流行的网络应用之一，游戏作弊对其产生的危害很大。对于作弊软件使用者而言：制作者通过高价出售作弊软件来牟取暴

收稿日期：2018-04-27 修回日期：2018-10-18；
基金项目：国家自然科学基金(61103115，61103172)，湖南省自然科学基金(11JM058，11JJ2037)，上海市科委科技创新行动计划高新技术领域项目(16511101000)；
作者简介：郑红(1973-)，女，江苏淮南，博士，副教授，研究方向为形式化建模、服务计算。

Published by Journal of System Simulation, 2020

Journal of System Simulation, Vol. 32 [2020], Iss. 3, Art. 13

利、通过软件后门来窃取使用者的资料和财产、传播病毒、使其他玩家丧失游戏体验等。对于游戏开发、代理者而言：作弊软件的使用会破坏公平游戏的环境、使玩家流失、对服务器的稳定性造成影响等。作弊软件严重影响了游戏环境，也给开发商和运营商带来了重大损失。如何检测游戏作弊是目前游戏开发者和用户关心的热点问题之一，目前反作弊技术效率不高，主要依赖于签名匹配。为了使作弊检测更加容易，并摆脱对于签名匹配的依赖，可以将作弊流程用工作流网来建模。而不同的游戏作弊类型往往具有相似的特征，且作弊软件的作弊方法在有限的范围内，因此本文提出了一种基于工作流网的游戏作弊检测系统。

1 背景知识和相关工作

1.1 游戏作弊的分类

玩家用来获得相对于其他玩家的优势或在网络游戏中实现目标（如果根据游戏规则或游戏运营商的判断，这个优势或目标是该玩家不应该获得的）的任何行为都是作弊行为[3]。下面介绍几种典型的游戏作弊类型。

1.1.1 利用系统欺诈

很多的作弊程序通过修改客户端的游戏代码和配置数据来实现作弊[4]。作弊者可以操纵游戏数据以获得不公平的优势。例如，在 FPS (First-Person Shooter，第 1 人称射击)游戏中，作弊者可以修改磁盘中的游戏文件或者内存中的数据，来消除武器的后坐力、获得无限的弹药或者使游戏角色获得更快的移动速度[5]。

作弊程序也可以快速地篡改游戏程序来获取原本是对玩家不可见的游戏状态，典型的例子是“开图”(Map-hack)，即显示地图上未经探索的区域。

作弊程序可以修改发送给服务器端或客户端的数据包，以欺诈服务器或者修改本地文件、数据来实现作弊。

1.1.2 修改客户端基础结构

在客户端不修改游戏程序，配置或数据的情况下，玩家可以通过修改客户端基础设施(如操作系统中的设备驱动程序)来作弊。例如，他们可以修改图形驱动程序以使墙壁变得透明，以便透视它，找到应该隐藏在其后面的其他玩家。这就是所谓的“透视”，在一些网络游戏中流行，尤其是 FPS 游戏。

1.1.3 利用游戏漏洞或者 BUG

作弊者利用游戏程序或游戏设计本身的缺陷而不必修改游戏代码或数据。这样的游戏漏洞一旦发现，会给某些玩家带来很大的好处。首批多用户虚拟环境之一的 Lucasfilm's Habitat 便存在这样的漏洞[6]。由于无意的定价错误，游戏中的人可以向当铺出售虚拟物品而获得比自动售货机更高的价格。通过在自动售货机和当铺之间来回穿梭，一些玩家在一夜之间成了虚拟世界的百万富翁。

1.2 反作弊技术

游戏作弊严重影响了游戏环境。主流的商用反作弊系统如 PunkBuster、Valve Anti-Cheat(VAC)和 BattleEye 是附加在客户端上由监控功能来检测作弊的。这些解决方案完全依赖于签名匹配，因而通常被认为是效率不高的。

Kaiser 等提出了一种不需要签名匹配的基于异常检测的作弊检测方法[7]。这种方法依赖对客户端主机进行可靠的监视，但是这样的监视却是无法保证的。文献[8-9]使用符号执行来验证接收的消息是否可以通过服务器端的真实游戏规则来解释，以检测篡改游戏客户端的欺骗行为。一些作弊者破解了在线游戏中使用的事件排序协议。为了防止这种欺骗，Baughman 等[10]提出了锁步协议(Lockstep Protocol)。在 FPS 游戏中有一些关于自动瞄准机器人检测的文献[11-14]。然而，这些文献提出的方法在区分高水平玩家和自动瞄准机

https://dc-china-simulation.researchcommons.org/journal/vol32/iss3/13
DOI: 10.16182/j.issn1004731x.joss.18-0243

器人时容易出现无效的情况，这是自动瞄准机器人检测的关键挑战。Yeung 等[15]提供了一种使用动态贝叶斯网络(Dynamic Bayesian Network，DBN)来检测 FPS 游戏中的自动瞄准机器人的可扩展方法。他们的方法提供了很好的结果，但如果使用更多的特征，可以获得更好的结果。

综上所述，目前反作弊技术的主要瓶颈是效率低并且过多依赖于签名匹配。因此本文提出一种高效且不依赖于签名匹配的游戏作弊检测系统。

2 基于 Petri 网的游戏作弊检测系统

2.1 Petri 网

工作流在检测方面的应用很广，常用于冲突检测、一致性检测等[16-17]。工作流建模的方法有很多种。与其它形式化方法相比，Petri 网具有明确的语义，强大的描述能力，易于编程或模拟[18-20]。在文献[21]中，以客户预定旅行为例，建立了对应的工作流网模型。文献[22]使用 Petri 网对业务流程进行建模，提出基于 Petri 网的映射变迁关系相似性度量方法，对传统的工作流网进行改进。

如果一个流程可以用 Petri 网建模，就可以获得大量有关它的动态行为和静态结构的信息，并改进它的逻辑结构和系统性能。文献[23]运用 Petri 网代数将 Petri 网化简和精细化转换进行了详细的论述，文献[24]文提出一种利用 Petri 网合成规则进行工作流网简化的方法，通过对网中某些节点的细化与实现，形成层次化的网模型，原系统的重要性质被保持，有效降低了建模的复杂度。文献[25]提出了基于搜索树的业务流程 Petri 网模型抽象化简方法。根据具体的化简规则，本系统共有 2 个模块：检测模块和更新模块。作弊模型的更新模块对于新生成的模型，通过转化为流程树，利用 Petri 网的化简规则，将新旧模型进行融合，多个 Petri 网可以被融合成一个 Petri 网[26-27]。在不影响检测结果的情况下，对模型规模进行缩减，可以明显降低存储空间的占用。

2.2 流程树

流程树是一种工作流网中流程的树的表示形式[28]。每个叶节点和每个非叶子节点分别代表流程中的一个动作和一个运算符。本文中用到 3 种运算符，顺序(→)、选择(×)和并行(^)。每个运算符可以映射为工作流网的一部分。

2.3 游戏作弊检测系统

游戏作弊检测系统主要由 2 个模块组成，分别为作弊行为检测模块和作弊模型更新模块。如图 1 所示是作弊检测系统的总体框架。图中左半部分为作弊行为检测模块，右半部分为作弊模型更新模块。其中作弊行为检测模块的功能为：对于疑似作弊行为，用模型库中存在的作弊行为模型进行检测，若没有相应的模型，则根据该作弊行为，建立其对应的工作流网模型，使得该工作流网模型可以用来检测同一类型的作弊；作弊模型更新模块的功能为：将作弊行为检测模块建立的工作流网模型存储到模型库中，并与库中已存在的工作流网模型进行融合，以在不影响检测结果的前提下尽可能节约存储空间。

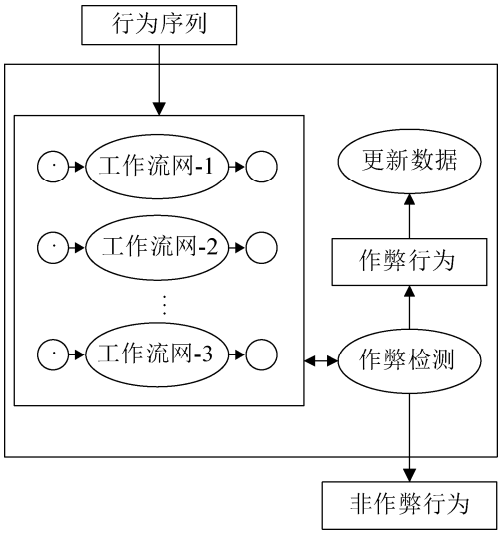

图 1 游戏作弊检测系统的总体框架
Fig. 1 Overall framework of game cheating detection system

2.3.1 检测模块

由于作弊行为具有相似的特征，并且作弊行

Journal of System Simulation, Vol. 32 [2020], Iss. 3, Art. 13

为的种类是有限的，因此一个作弊行为可以建模为一个工作流网。

定义 1 作弊行为模型

对于一个由一组动作序列 Σ 构成的作弊行为 α，工作流网 N 称为 α 的作弊模型，当且仅当 Σ 中的每个动作序列 σ 都能在 N 中由[p_{in}]变换至[p_{out}]，而不包含在 Σ 中的任何动作序列都不能在 N 中由[p_{in}]变换至[p_{out}]。

例如，假设有一个通过修改内存数据实现作弊的作弊软件，其作弊步骤如下：

(1) 搜索内存中的数据；

(2) 获得内存基址；

(3) 修改数据；

(4) 完成作弊行为。

这个流程可以建模为工作流网 N_1，如图 2 所示。

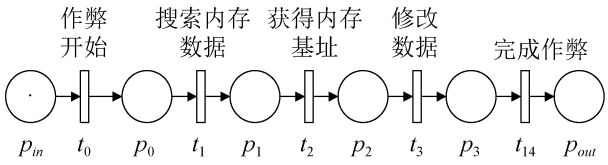

图 2 通过修改内存中数据作弊的工作流网模型 N_1

Fig. 2 Workflow model N_1 of cheatings by modifying data in memory

根据定义 1，可以得到一个性质。

性质 1 作弊行为模型

动作序列 σ 是一个作弊行为，当且仅当存在作弊行为模型 N 使得动作序列 σ 能在 N 中由[p_{in}]变换至[p_{out}]。

根据性质 1，可以设计一个检测过程来检测给定的动作序列是否为特定类型的作弊行为。

检测过程：

输入：若干个动作序列 σ_1，σ_2，…，σ_n，以及特定类型的作弊行为模型 N。

输出：动作序列 σ_i 是否为该类型的作弊行为。

假设有 2 个疑似修改内存数据作弊的作弊行为以及用于检测该类型作弊行为的模型 N_1，2 种作弊行为的动作序列分别为：

$$\sigma_1 = \{t_0, t_1, t_2, t_4, t_{14}\}$$

$$\sigma_2 = \{t_0, t_1, t_2, t_3, t_{14}\}$$

用检测过程 2 检测动作序列 σ_1，得到执行过程序列

$$[p_{in}][N_1, t_0 > [p_0][N_1, t_1 > [p_1][N_1, t_2 > [p_2]$$

可见 σ_1 在[p_2]中无法被触发，因此 σ_1 不是修改内存中数据的作弊行为；检测动作序列 σ_2，得到执行过程序列

$$[p_{in}][N_1, t_0 > [p_0][N_1, t_1 > [p_1][N_1, t_2 > [p_2][N_1, t_3 > [p_3][N_1, t_{14} > [p_{out}]$$

很显然动作序列 σ_2 能在 N_1 中经过有限次触发由[p_{in}]变换至[p_{out}]，因此 σ_2 是修改内存中数据的作弊行为。

由定义 1、性质 2 以及检测过程，可以看出，作弊行为检测模块可以用已知的作弊行为模型对疑似的作弊行为进行检测、判断，并能通过检测过程有效地区分两个非常相似的行为。除了检测作弊行为外，作弊行为检测模块可以用工作流网对新的作弊行为进行建模，得到的作弊行为模型会被转交给作弊模型更新模块，进行模型的存储和融合。

2.3.2 更新模块

市面上的游戏作弊软件很多，不同的作弊软件有不同的工作原理，其作弊行为模型也各不相同。即使是同一款作弊软件，在应对开发商更新的反作弊机制的更新中，也可能改变其作弊行为模型。游戏作弊检测系统需要不断地处理新的作弊行为。新的作弊行为的产生会导致作弊行为模型库中存储的作弊行为模型数量迅速增长。然而，有些模型却和已存在的模型有一定的相似性。因此，作弊模型更新模块的功能除了将新的作弊软件的作弊行为模型添加到模型库中，还需要将相似的检测模型融合成一个大的模型，从而尽可能降低存储空间占用。子模型能检测出的作弊行为在融合后的模型上仍可检测出来，但存储空间可以大幅降低。

https://dc-china-simulation.researchcommons.org/journal/vol32/iss3/13
DOI: 10.16182/j.issn1004731x.joss.18-0243

定义 2 对于作弊行为 α_x 对应的作弊行为模型 N_x 和作弊行为 α_y 对应的作弊行为模型 N_y，工作流网 N 称为 N_x 和 N_y 的融合，此时 Σ_x 和 Σ_y 中的每个动作序列 σ 都能在 N 中由[p_{in}]变换至[p_{out}]，而不包含在 Σ_x 和 Σ_y 中的任何动作都不能在 N 中由[p_{in}]变换至[p_{out}]，其中，作弊行为 α_x 由动作序列组 Σ_x 构成；作弊行为 α_y 由动作序列组 Σ_y 构成。

例如，在 2.3.1 节中提到的 2 种篡改内存的作弊行为只在获得内存基址之后有所区别，因此可以将两者进行融合并得到如下的流程。

(1) 搜索内存中的数据；

(2) 获得内存基址；

(3) 若数据需要被修改，则修改数据，否则删除数据；

(4) 完成作弊行为。

融合之后的模型 N_3 如图 3 所示。再次使用检测过程来检测上一小节提到的 3 个疑似作弊行为。对于动作序列 σ_1，得到执行过程序列

$$[p_{in}][N_1,t_0>[p_0][N_1,t_1>[p_1][N_1,t_2>$$

$$[p_2][N_1,t_4>[p_3][N_1,t_{14}>[p_{out}]$$

即动作序列 σ_1 能在 N_3 中经过有限次触发由[p_{in}]变换至[p_{out}]，因此 σ_1 是篡改内存的作弊行为；检测动作序列 σ_2，得到执行过程序列

$$[p_{in}][N_1,t_0>[p_0][N_1,t_1>[p_1][N_1,t_2>$$

$$[p_2][N_1,t_3>[p_3][N_1,t_{14}>[p_{out}]$$

即动作序列 σ_2 也能在 N_3 中经过有限次触发由[p_{in}]变换至[p_{out}]，因此 σ_2 也是篡改内存的作弊行为。

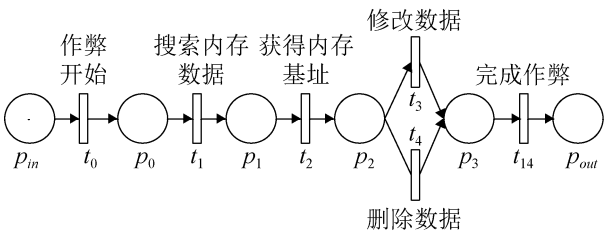

图 3 N_1 和 N_2 融合后的篡改内存作弊的工作流网模型 N_3
Fig. 3 Fusion workflow model of N_1 and N_2#

定义 3 融合规则 R

输入：作弊行为工作流网模型 N_x 和 N_y，

输出：融合的工作流网模型 N_z。

融合规则需要满足的条件：令 Π_x 和 Π_y 分别表示 N_x 和 N_y 的流程树，其中，Π_x 包括两个子树 π 和 π_x，π_x 的根是 π 的一个节点 v 的子节点；Π_y 包括 π 和另一个子树 π_y，π_y 的根是 v 的子节点，且 π_x 和 π_y 不相交。

根据如下的流程树 Π_z 构造工作流网模型 N_z：Π_z 包含子树 π、一个选择运算符、π_x 和 π_y。其中 π_x 和 π_y 的根节点是此选择运算符的子节点，而选择运算符是 的子节点。

性质 2 融合性质

对于作弊行为模型 N_x 和 N_y，称 $R(N_x, N_y)$ 是 N_x 和 N_y 的一个融合。

图 4 是融合规则在内存篡改作弊行为中应用的一个例子。

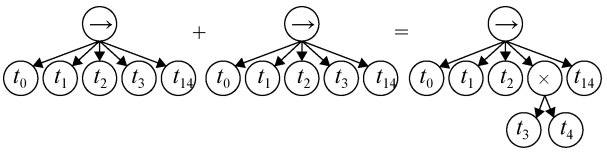

图 4 融合规则在数据篡改作弊行为中的应用举例
Fig. 4 Example of application of fusion rules in data tampering cheating

3 系统的应用与分析

3.1 系统应用举例

游戏作弊检测系统更新模块的应用以当前最热门的网游《绝地求生：大逃杀(Playerunknown's Battlegrounds，PUBG)》为例。

为这款游戏设计的作弊软件主要通过篡改本地数据和数据包实现作弊。因此，本节对这两种的作弊软件进行了测试。其中，作弊软件 A 篡改内存数据(其作弊行为模型即 2.3.2 节中工作流网模型 N_3，如图 3 所示)，作弊软件 B 篡改数据包。

作弊软件 B 的作弊方式与作弊软件 A 有所不同，其作弊过程如下：

(1) 抓取从服务器端发出的数据包或抓取从

Journal of System Simulation, Vol. 32 [2020], Iss. 3, Art. 13

客户端发出的数据包；

(2) 筛选可用于作弊的数据；

(3) 修改数据；

(4) 若该包是从服务器端发出的则将修改过的数据包发往客户端，否则将修改过的数据包发往客户端；

(5) 完成作弊行为。

根据作弊软件 B 的作弊过程，可以给出作弊软件 B 的工作流网模型(记为 N_4)，如图 5 所示。

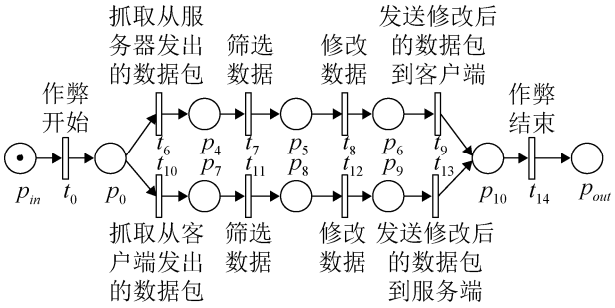

图 5 通过修改数据包作弊的工作流网模型 N_4

Fig. 5 Workflow model N_4 of cheatings by tampering data in packets

作弊模型更新模块应用融合规则 R 将工作流网模型 N_3 和 N_4 对应的等价流程树 PT_3 和 PT_4 融合成一个流程树 PT_5，其过程如图 6 所示。

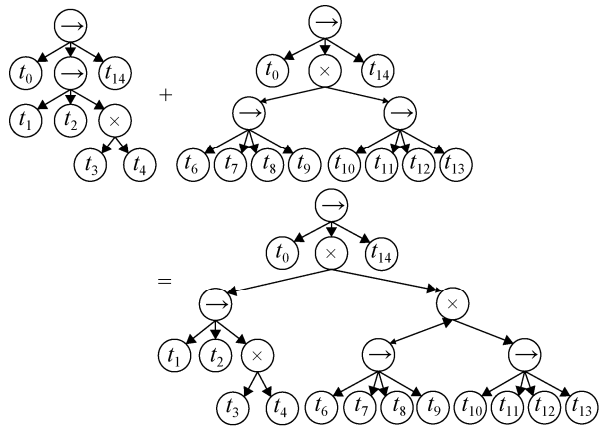

图 6 流程树 PT_3 和 PT_4 的融合过程

Fig. 6 Fusion process of process tree 3 and 4

3.2 系统分析与评价

3.2.1 系统分析与评价指标

更新模块的主要作用是在不影响检测结果的情况下尽量降低存储空间，因此降低的存储空间可以用来反应作弊模型更新模块的效率高低。一般地，工作流网模型的规模可以直观地用库所和变迁的数量和来表示，即，

$$S_N=|P|+|T| \tag{1}$$

式中：$|P|$为库所的数量；$|T|$为变迁的数量。

用存储空间降低率 Rr 表示作弊模型更新模块的效率，即，

$$R_{\mathrm{r}}=1-\frac{S_{N_{fusion}}}{\sum S_{N_i}} \tag{2}$$

式中：$S_{N_{fusion}}$ 为融合后的工作流网模型的大小。存储空间降低率 R_{r} 的值越大，表示作弊模型更新模块融合的效率越高。

工作流网模型 N_1、N_2、N_3 和 N_4 的大小分别为 $S_{N_1}=11$，$S_{N_2}=11$，$S_{N_3}=12$ 和 $S_{N_4}=20$。若将 N_1、N_2 和 N_4 直接存入作弊行为模型库，则需要存储空间为 $S_{N_1}+S_{N_2}+S_{N_4}=42$。而融合后的工作流网模型 N_5 的大小为 $S_{N_5}=30$。则工作流网模型 N_5 相对于 N_1，N_2 和 N_4 的存储空间降低率为 $Rr=1-30/42\approx 28.57\%$，即作弊模型更新模块的效率达到 28.57%。

3.2.2 存储空间降低率

除了 3.1 节提到的两款作弊软件，本文一共对 7 款不同的用于《绝地求生》的作弊软件进行了实验与测试。这 7 款软件共涉及到 10 种不同的作弊行为，如篡改内存、篡改数据包、API 拦截、WinSock 拦截等。

10 种不同的作弊行为对应的工作流网模型的参数如表 1 所示。

原始模型和融合模型的规模对比曲线图如图 7 所示。

图 7 中横坐标表示加入模型库的模型数量，纵坐标表示模型规模。从图 7 中可以看到，不使用作弊模型更新模块，直接将作弊行为模型存入模型库，需要占用的存储空间增长很快，但使用了作弊模型更新模块将作弊行为模型融合之后，存储空间增长速度得到明显的抑制，10 个模型的

https://dc-china-simulation.researchcommons.org/journal/vol32/iss3/13
DOI: 10.16182/j.issn1004731x.joss.18-0243

总规模 ΣS_N=177，而融合后的模型规模 S_N=75，存储空间降低率达到了 $Rr=1-75/177=57.63\%$。实验结果说明融合规则 R 和提出的游戏作弊检测系统框架是有效的。

表 1　10 种不同作弊行为模型的参数
Tab. 1　Parameters of 10 different models

模型	模型大小	总大小	融合后模型大小	存储空间降低率/%
1	15	15	/	/
2	15	30	16	46.67
3	24	54	34	37.04
4	17	71	37	47.89
5	28	99	43	56.57
6	19	118	48	59.32
7	16	134	58	56.72
8	15	149	67	55.03
9	15	164	72	56.10
10	13	177	75	57.63

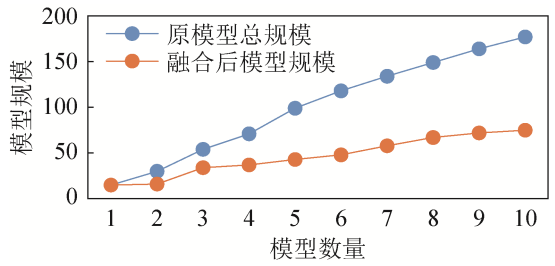

图 7　原始模型和融合模型的规模对比曲线图
Fig. 7　Ccomparison of the size between original models and fused model

图 8 是存储空间降低率折线图。从图中拟合虚线可以得出，随着模型数量增多，存储空间降低率不断增大，并且当模型数量足够多时，存储空间降低率收敛。这是因为随着添加到作弊行为模型库中的模型越多，新模型与之前已添加的模型相似的几率越高。实验结果显示作弊模型更新模块是可行有效的。

在安全性方面，这种方法通过提高作弊行为流的可读性使检测作弊行为更加方便有效。此外，还可以了解作弊的常见行为，以便采取有效的防范措施。

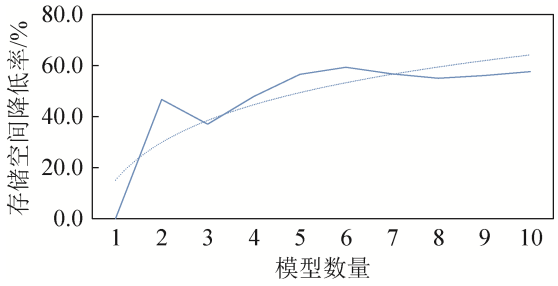

图 8　存储空间降低率折线图
Fig. 8　Line chart of reduction rate

3.2.3 真实作弊环境下效率对比

主流的商用反作弊系统有 PunkBuster、Valve Anti-Cheat(VAC)和 BattleEye 等，其主要检测方法均是通过某些手段获取若干作弊软件样例，并提取其签名。通过匹配签名，从而确定用户是否有作弊行为。本次实验选取采用以上 3 款不同作弊检测软件的 3 款游戏来进行试验。对于每一款游戏，用 10 款不同的作弊软件进行测试。分别与基于工作流网的检测方法进行对比，结果如表 2~4 所示。

表 2　PunkBuster 与基于工作流检测方法比较
Tab. 2　PunkBustercompare with Cheating Detection System Based on Petri Nets　/%

系统名称	检测到	未检测到	检测率
PunkBuster	6	4	60
基于工作流网的检测方法	6	4	60

表 3　VAC 与基于工作流检测方法比较
Tab. 3　VAC compare with Cheating Detection System Based on Petri Nets　/%

系统名称	检测到	未检测到	检测率
VAC	5	5	50
基于工作流网的检测方法	6	4	60

表 4　BattleEye 不同行为时的误判结果统计
Tab. 4　BattleEyecompare with Cheating Detection System Based on Petri Nets　/%

系统名称	检测到	未检测到	检测率
BattleEye	4	6	40
基于工作流网的检测方法	7	3	70

主流的作弊检测系统均是基于签名的原理。对于已知其签名的作弊软件，有很好的效果，但

是对于未知签名的作弊软件的检测率就显得力不从心。基于工作流的检测方法不使用签名的方式，在对未知外挂的检测率上要更好。

4 结论

数十年来网络游戏产业不断发展，游戏作弊发展更加迅速。目前的反作弊方法不能有效地检测插件。本文提出了一个基于 Petri 网的游戏作弊检测系统框架。该系统由 2 个模块组成。检测模块通过构造作弊行为模型来帮助检测作弊行为，而更新模块将作弊模型融合成一个模型以减少数据库的存储空间占用。最后给出了一个应用举例来分析该框架的可行性和有效性。但该系统在处理大规模的模型时，效率不高，下一步的研究重心是提高大规模模型时框架的效率、进一步降低存储空间占用。

参考文献：

[1] 智研咨询集团. 2015-2020 年中国网络游戏行业深度调研及市场前景预测报告[R/OL]. 2015.10/2018.08. http://www.chyxx.com/research/201510/347879.html.
2015-2020 In-depth research and market prospect forecast report of China's online game industry [R/OL]. 2015.10/2018.08. http://www.chyxx.com/research/201510/347879.html.

[2] 中国互联网络信息中心. 第 40 次中国互联网络发展状况统计报告[R/OL]. 2017.07/2018.08. http://www.cac.gov.cn/2017-08/04/c_1121427728.htm.
China Internet Network Information Center. The 40th China Statistical Report on Internet Development[R/OL]. 2017.07/2018.08. http://www.cac.gov.cn/2017-08/04/c_1121427728.htm.

[3] Jeff Y, Randell B. A systematic classification of cheating in online games[C]. ACM SIGCOMM Workshop on Network and System Support for Games ACM. New York, USA: Association for Computing Machinery, 2005: 1-9.

[4] Pritchard M. How to Hurt the Hackers: The Scoop on Internet Cheating and How You Can Combat It[R/OL]. 2000.07/2018.08.https://www.gamasutra.com/view/feature/131557/how_to_hurt_the_hackers_the_scoop_.php.

[5] Liu D, Gao X, Zhang M, et al. Detecting Passive Cheats in Online Games via Performance-Skillfulness Inconsistency[C]. IEEE/IFIP International Conference on Dependable Systems and Networks. Colorado, USA: IEEE, 2017: 615-626.

[6] Morningstar C, Farmer F R. The lessons of Lucasfilm's habitat[J]. Cyberspace First Steps(S1941-8477). 2008 (1): 273-302.

[7] Kaiser E, Feng W C, Schluessler T. Fides: remote anomaly-based cheat detection using client emulation[C]. ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November. Chicago, USA: DBLP, 2009: 269-279.

[8] Bethea D, Cochran R A, Reiter M K. Server-side verification of client behavior in online games[J]. Acm Transactions on Information & System Security (S1094-9224), 2008, 14(4): 1-27.

[9] Cochran R, Reiter M. Toward online verfication of client behavior in distributed applications[C]. NDSS 2013. San Diego, California: 20th Network & Distributed System Security Symposium, 2013.

[10] Baughman N E, Levine B N. Cheat-proof playout for centralized and distributed online games[C]. INFOCOM 2001. Twentieth Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE. Alaska, USA: IEEE, 2001, 1: 104-1131.

[11] Alayed H, Frangoudes F, Neuman C. Behavioral-based cheating detection in online first person shooters using machine learning techniques[C]. Computational Intelligence in Games. Niagara Falls, Canada: IEEE, 2013: 1-8.

[12] Galli L, Loiacono D, Cardamone L, et al. A cheating detection framework for Unreal Tournament III: A machine learning approach[C]. Computational Intelligence and Games. Seoul, South Korea: IEEE, 2011: 266-272.

[13] Yu S Y, Hammerla N, Yan J, et al. Aimbot Detection in Online FPS Games Using a Heuristic Method Based on Distribution Comparison Matrix[C]. International Conference on Neural Information Processing. Doha, Qatar: Springer, Berlin, Heidelberg, 2012: 654-661.

[14] Yu S Y, Hammerla N, Yan J, et al. A statistical aimbot detection method for online FPS games[C]. International Joint Conference on Neural Networks. Brisbane, Australia: IEEE, 2012: 1-8.

[15] Yeung S F, Lui J C. Dynamic Bayesian approach for detecting cheats in multi-player online games[J]. Multimedia Systems(S0942-4962), 2008, 14(4): 221-236.

[16] 孟庆丰，李海波. 面向过程的资源组合排斥冲突检测方法[J]. 计算机应用与软件, 2016, 33(4): 22-24, 43.

https://dc-china-simulation.researchcommons.org/journal/vol32/iss3/13
DOI: 10.16182/j.issn1004731x.joss.18-0243

Meng Qingfeng, Li Haibo. Process-oriented Collision Detection Method for Resources Composition Exclusion[J]. Computer Applications and Software, 2016, 33(4): 22-24, 43.

[17] 孙亚男, 杜玉越, 刘伟. 高速公路冰雪应急流程挖掘与模型改进方法[J]. 计算机科学与探索, 2015, 9(12): 1523-1530.
Sun Ya'nan, Du Yuyue, Liu Wei. Highway Snow and Ice Emergency Response Process Mining and Model Improvement[J]. Journal of Frontiers of Computer Science & Technology, 2015, 9(12): 1523-1530.

[18] Labadi K, Chen H. Modelling, analysis, and optimization of supply chains by using Petri net models: The state-of-the-art[J]. International Journal of Business Performance and Supply Chain Modelling (S1758-9401), 2010, 2(3/4): 188-215.

[19] Chen H, Amodeo L, Chu F, et al. Performance Evaluation and Optimization of Supply Chains modelled by Batch Deterministic and Stochastic Petri Nets[J]. IEEE transactions on Automation Science and Engineering (S1545-5955), 2005, 2(2): 132-144.

[20] 孙静, 秦世引, 宋永华, 等. 一种基于Petri网和概率信息的电力系统故障诊断方法[J]. 电力系统自动化, 2003, 27(13): 10-14, 23.
Sun Jing, Qin Shiyin, Song Yonghua, et al. A fault diagnosis method for power system based on Petri net and probability information[J]. Automation of Electric Power Systems, 2003, 27(13): 10-14, 23.

[21] 庞善臣, 蒋昌俊. Petri 网在工作流系统建模和分析中的应用[J]. 系统仿真学报, 2005, 17(增 1): 100-104.
Pang Shanchen, Jiang Changjun. Petri net and its application in workflow system modeling and analysis[J]. Journal of System Simulation, 2005, 17(S1): 100-104.

[22] 许荣斌, 石军, 张鹏飞, 等. Petri 网的映射变迁关系相似性度量[J]. 浙江大学学报, 2017, 51(6): 1205-1213.
Xu Rongbin, Shi Jun, Zhang Pengfei, et al. Similarity measurement of transition mapping relation using Petri net[J]. Journal of Zhejiang University, 2017, 51(6): 1205-1213.

[23] Li J, Zhou M C, Dai X Z. Reduction and refinement by algebraicoperations for Petri net transformation [J]. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans (S1083-4427), 2012, 42(5): 1244-1255.

[24] 杜玉越, 孙亚男, 刘伟. 基于 Petri 网的模型偏差域识别与模型修正[J]. 计算机研究与发展, 2016, 53(8): 1766-1780.
Du Yuyue, Sun Ya'nan, Liu Wei. Petri Nets Based Recognition of Model Deviation Domains and Model Repair[J]. Journal of Computer Research and Development, 2016, 53(8): 1766-1780.

[25] 方欢, 何路路, 方贤文, 等. 基于搜索树的业务流程 Petri 网模型抽象化简方法[J]. 控制理论与应用, 2018, 35(1): 92-102.
Fang Huan, He Lulu, Fang Xianwen, at al. A search-tree-based abstraction method for business process Petri nets models[J]. Control Theory and Applications, 2018, 35(1): 92-102.

[26] Li J Q, Fan Y S. Workflow Model Validation Based on Petri Net Simplification Method[J]. Information and Control (S0019-9958), 2001, 6: 492-497.

[27] Tan W, Xu W, Yang F, et al. A framework for service enterprise workflow simulation with multi-agents cooperation[J]. Enterprise Information Systems (S1751-7575), 2013, 7(4): 523-542.

[28] Aalst W V D, Buijs J, Dongen B V. Towards Improving the Representational Bias of Process Mining[J]. Lecture Notes in Business Information Processing(S1865-1348), 2017, 116: 39-54.