软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2021,32(7):1978–1998 [doi: 10.13328/j.cnki.jos.006268]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于事件关系保障识别质量的自适应分析方法*

王 璐, 李青山, 吕文琪, 张 河, 李 昊

(西安电子科技大学 计算机科学与技术学院,陕西 西安 710071)

通讯作者: 李青山, E-mail: qshli@mail.xidian.edu.cn

摘 要: 目前自适应软件正在为众多领域系统提供着对运行环境的适应能力.如何建立一种能够保障识别质量的自适应分析方法,使之可从运行环境中快速且准确地识别出异常事件,是确保自适应软件长期稳定运行所必须考虑的研究问题之一.当前运行环境的不确定性给该问题的攻关带来两方面的挑战:其一,现有分析方法一般通过预先建立环境状态与事件之间的映射关系来识别事件.但在系统运行之前,已无法仅凭经验确定环境状态并建立全面且正确的映射关系.仅依赖映射关系建立分析方法的设计思路已无法保障识别的准确性.其二,不确定环境何时会发生何种事件已变得不可预期.如果采用现有设计思路,定期地获取环境状态再进行事件识别,则无法保障识别效率.然而,目前却缺乏应对这些紧迫挑战的相关工作,因此提出了一种基于事件关系保障识别质量的自适应分析方法(self-adaptation analysis method for recognition quality assurance using event relationships,简称 SAFER).SAFER 采用序列模式挖掘算法、模糊故障树与贝叶斯网络等技术抽取并建模事件因果关系,并基于该类关系与映射关系通过贝叶斯网络的正向推理能力共同识别事件,与传统的仅依赖映射关系的识别方法相比可保证识别的准确性;基于贝叶斯网络的反向推理能力,确定易引发事件的精英感知对象,并动态调整获取精英感知对象状态数据的采样周期,以便于在事件发生后尽快获得相关环境状态,从而保障识别效率.实验结果表明,在自适应软件实际运行过程中,SAFER 可实现对事件的识别并保障识别准确性与识别效率,为自适应软件稳定运行提供了有效支持.

关键词: 自适应软件;质量保障;自适应分析方法;事件识别;贝叶斯网络

中图法分类号: TP311

中文引用格式: 王璐,李青山,吕文琪,张河,李昊.基于事件关系保障识别质量的自适应分析方法.软件学报,2021,32(7):1978–1998. http://www.jos.org.cn/1000-9825/6268.htm

英文引用格式: Wang L, Li QS, Lü WQ, Zhang H, Li H. Self-adaptation analysis method for recognition quality assurance using event relationships. Ruan Jian Xue Bao/Journal of Software, 2021,32(7):1978–1998 (in Chinese). http://www.jos.org.cn/1000-9825/6268.htm

Self-adaptation Analysis Method for Recognition Quality Assurance Using Event Relationships

WANG Lu, LI Qing-Shan, LÜ Wen-Qi, ZHANG He, LI Hao

(School of Computer Science and Technology, Xidian University, Xi'an 710071, China)

Abstract: At present, self-adaptive software is providing the ability to adapt to the operating environment for many systems in different fields.How to establish a self-adaptation analysis method which can recognize abnormal events at runtime quickly and achieve the recognition quality assurance, is one of the research issues that must be considered to ensure the long-term stable operation of the

* 基金项目: 国家自然科学基金青年科学基金(61902288); 国家自然科学基金(61672401, 61972300); 陕西省自然科学基础研究计划(2020JQ-300)

Foundation item: Youth Science Foundation of the National Natural Science Foundation of China (61902288); Foundation item: National Natural Science Foundation of China (61672401, 61972300); Basic Research Program of Natural Science of Shaanxi (2020JQ-300)

本文由"面向非确定性的软件质量保障方法与技术"专题特约编辑陈俊洁副教授、汤恩义副教授、何啸副教授以及马晓星教授推荐.

收稿时间: 2020-09-15; 修改时间: 2020-10-26; 采用时间: 2020-12-14; jos 在线出版时间: 2021-01-22

self-adaptive software. The uncertainty of the runtime environment brings two challenges to this problem. On the one hand, the analysis method usually recognizes the events by pre-establishing the mapping relationships between the environment state and the events. However, due to the complexity of the operating environment and the unknown changes, it is impossible to establish comprehensive and correct mapping relationships based on experience before the system is running, which affect the accuracy of event recognition; On the other hand, the changing operating environment makes it impossible to accurately predict when and which event will occur. If the current way is used to obtain the environmental status using constant sensing period and recognize events, then the recognition efficiency cannot be guaranteed. However, it is still blank about how to deal with these urgent challenges. Therefore, this study proposes a self-adaptation analysis method for recognition of quality assurance using event relationships (SAFER). SAFER uses sequential pattern mining algorithm, fuzzy fault tree (FFT), and Bayesian network (BN) to extract and model the causalities between events. This study uses the event causal relationships and mapping relationships to recognize events through the BN forward reasoning, which can ensure the accuracy of recognition compared with the traditional analysis methods that only rely on mapping relationships. Moreover, this study establishes the elitist set of monitoring objects through the BN backward reasoning, then modifies the sensing period of monitoring objects in elitist set dynamically in order to obtain the environmental status as soon as possible after the abnormal events occurred, so as to ensure the efficiency of recognition. The experimental results show that SAFER can effectively improve the accuracy and efficiency of the analysis process, and support long-term stable operation of self-adaptive software.

Key words: self-adaptive software; quality assurance; self-adaptation analysis method; event recognition; Bayesian network

近年来,随着软件运行环境的日益复杂,自适应软件已广泛应用于航天器控制、远程医疗、电子商务等需长期稳定运行的领域中,为众多领域系统提供着适应运行环境的能力[1].自适应软件是一类特殊的软件系统,通过自适应过程实现对系统参数、行为或结构的动态调整,从而适应变化的运行环境[2].自适应过程包括监测环境状态的感知过程(monitor)、识别异常事件的分析过程(analyze)、产生调整策略的决策过程(plan)与调整软件系统的执行过程(execute)[1].分析过程在自适应过程中处于承上启下的位置,其识别的异常事件(以下简称事件)是触发后续决策与执行过程实现系统调整的依据.但是由于分析过程发生在软件运行阶段,产生的实时识别结果无法在软件运行前经过严格且完整的测试与验证[3].如果识别结果存在错误或未能及时产生识别结果,均会导致系统行为退化,甚至会将系统引入到错误的状态中[4−6].例如,2015 年 9 月,亚马逊 AMS 平台 DynamoDB 由于网络中断引发超时事件,并级联导致了更大面积的断网,产生恶性循环导致故障持续时间长达 5 个小时,对软件质量和用户服务产生了恶劣影响.若在最初服务超时事件发生之后能够及时且准确地识别并处理该事件,或许能够避免后续更大问题的发生.

因此,为确保自适应软件长期且稳定地运行,分析过程需在实现事件识别功能的同时保障识别质量.分析过程通过自适应分析方法实现事件识别功能.自适应分析方法以感知过程输出的环境状态数据为输入,识别其中发生的事件(如软件服务资源异常、应用失效和网络拥塞等),并将事件信息输出给决策过程[2].而分析过程的识别质量则主要通过识别准确性与识别效率进行衡量[7].识别准确性是指正确识别出系统中发生的事件,并且不能误报未发生的事件,用于衡量分析过程产生正确识别结果的能力.识别效率是指从发生事件到识别出事件的时间差,时间差越小,则识别效率越高,用于衡量分析过程及时产生识别结果的能力.

目前,绝大多数自适应软件均运行在开放环境中,存在着环境状态复杂多变、未知环境变化频发等不确定性,对建立一种能够保障识别质量的自适应分析方法带来了两方面的挑战.其一,通常自适应分析方法需预先定义状态数据与事件的映射关系,然后通过匹配状态数据与映射关系以识别事件[8].但是,由于运行环境的多变性,部分环境状态信息须在软件运行后才能被完全确定,因此映射关系的定义不一定完全正确.并且,由于可能发生未知的环境变化以及人类认知的局限性,领域专家也难以在系统运行前定义出完备的映射关系.因此,如果按照传统思路,仅依赖映射关系建立分析方法,则无法完全保证识别的准确性.其二,由于环境的动态性,何种事件何时会发生已无法在软件运行前被有效预测[9],自适应分析方法只能在软件运行过程中根据状态数据动态判断事件发生的情况.因此,如果采用传统思路建立分析方法,定期地获取状态数据再进行事件识别,则无法保障在事件发生后能够立刻获得相关状态数据,进而导致识别效率较低.

然而,目前两类主要的自适应分析方法,即基于规则推理和基于本体推理的方法,基本上都需要预先建立映射关系[10−12],并未考虑如何在不确定环境下保障识别质量.自适应领域仅存在较少研究,考虑将环境状态数据与

映射关系进行模糊匹配[13]以保障识别的准确性,但并未从本质上解决仅依赖映射关系识别事件的缺陷.云计算、网络安全等其他领域存在采用概率理论、日志分析等方式识别事件并同时保障识别准确性的相关研究[14−18],但也存在考虑的因果关系维度单一、识别的事件类型有限等问题.另外,目前关于保障识别效率的研究则主要集中于智能电网、云计算等其他领域[19−22],自适应领域尚缺乏与之直接相关的研究工作.

因此,本文提出了一种基于事件关系保障识别质量的自适应分析方法(self-adaptation analysis method for recognition quality assurance using event relationships,简称 SAFER),通过挖掘并利用反映事件间触发情况的因果关系,对运行环境中发生的事件进行快速且准确的识别,达到实现事件识别功能并同时保障识别质量的目的.SAFER 具体包括以下几个要点.

(1) 关系驱动的事件识别与质量保障.目前,网络安全领域中存在着将事件关系作为识别复杂攻击依据的相关研究.该类研究虽然与本文的研究问题不同,但却为本文提供了可借鉴的研究思路.SAFER 依据序列模式挖掘算法和模糊故障树理论从系统运行日志中挖掘并建模事件因果关系,随后采用贝叶斯网络结合因果关系与映射关系共同识别事件,与仅依赖映射关系进行事件识别的现有工作相比,可有效保障识别的准确性;可定量计算出各类运行环境变化对事件发生的影响程度,并据此动态调整获取这些环境变化数据的采样周期,以便于在变化发生后可尽快地为贝叶斯网络提供相关数据以识别事件,保障识别效率.

(2) 率先优化分析过程的输入.目前,绝大多数关于自适应过程的研究工作普遍认为分析过程只能被动地获得状态数据作为输入,而无法主动提出自己对状态数据的针对性需求[23].SAFER 首次从保障识别效率的需求出发,动态地调整获取特定状态数据的采样周期,为分析过程提供切实所需的关键输入数据,更好地保障了分析过程的识别质量.这种通过优化过程输入以保障过程质量的方式,为后续开展自适应过程的相关研究提供了一种新的研究思路.

(3) 向系统维护人员提供优化建议.通过对系统资源、状态变化等基本事件进行概率重要度与关键重要度分析,SAFER 可提醒软件维护人员这些事件是较为容易导致系统发生整体异常的关键事件.软件维护人员可采取相应措施降低这些关键事件的发生概率,期望能够降低系统发生整体异常的概率,从而使得软件系统能够可靠、平稳地运行.

(4) 保持方法的动态更新以适应环境.运行环境的不可预期、动态多变等特征导致自适应分析方法本身也需不断修正,才能始终保持方法的有效性.SAFER 注重基于实时数据动态更新与方法相关的知识模型或参数设置,可根据最新的运行日志修正事件因果关系、更新贝叶斯网络模型的结构图和条件概率等,以保证本文方法的持续有效性.

本文第 1 节介绍目前关于自适应分析方法的研究现状,并重点分析保障识别质量的相关研究.第 2 节给出 SAFER 的概览.第 3 节主要介绍 SAFER,其中包括事件关系挖掘与建模、基于正向推理的事件识别、基于反向推理的变化监测这 3 部分.第 4 节通过具体实验以说明 SAFER 的有效性.第 5 节总结本文的工作,并对未来研究进行展望.

1 相关工作

目前,自适应分析方法主要可分为基于规则推理[10]、本体推理[11,12]两类方法.其中,基于规则推理的自适应分析方法依据预定义的映射规则,将实时环境状态匹配到相应的事件.基于本体推理的自适应分析方法同样需要依据预定义的系统状态本体、环境状态本体,推理出系统中发生的事件.这两种方法均主要关注分析过程中识别功能的实现,而未考虑对分析过程识别准确性的保障,且其较为简化的规则或者本体将不足以在不确定环境下准确地识别事件.

自适应领域仅存在较少的研究工作关注了如何在实现识别功能的同时保障识别的准确性.例如,相关学者采用模糊推理方法[13],将环境状态数据与事件进行模糊匹配,从而处理众多且不可预测的环境状态,但其本质上仍属于规则推理方法,只能将环境数据匹配到现有规则中,而规则制定的局限性同样会限制该方法的准确性.

除上述自适应领域的研究工作外,云计算、网络安全等其他领域也存在采用概率理论推理[14−16]、日志分

析[17,18]和事件关系分析[24,25]等方式识别事件并同时保障识别准确性的相关工作.其中,基于概率理论推理的方式根据故障表现与状态数据的因果关系求解事件发生的概率,并最终选取概率最大或者较大的事件作为输出.该方式可在一定程度上避免系统运行数据的波动性与随机性对事件识别的影响,从而保障识别的准确性,但其考虑的因果关系维度较为单一.基于日志分析的方式或采用无监督机器学习算法提取日志特征并判别日志序列是否存在异常,或建立异常检测框架将日志聚类成簇,并开展簇的特征与系统关键性能指标(KPIs)的关联分析,以识别出影响 KPIs 的事件.相较于其他方式,该方式可获得除状态数据以外更多的操作信息(如用户登录和用户发送 http 请求等行为信息)以辅助识别事件,从而保障识别的准确性.但该类方法很难判别故障类型,少有可判别故障类型的方法也仅可处理影响系统关键性能指标(KPIs)的事件[18].基于事件关系分析的方法目前主要应用于网络安全和工业设备管理等领域.其中,网络安全领域中的相关方法主要开展对告警信息的关系分析,以准确识别复杂网络攻击.工业设备管理领域中的相关方法则主要对设备声音、状态数据及设备故障进行关联关系分析,以提升设备故障的识别准确性.这些关于告警信息或故障信息的关系分析,启发了本文的研究思路.

关于保障识别效率的问题,目前自适应领域缺乏直接相关的研究工作,相关研究主要集中于智能电网、云计算等其他领域对故障检测效率的保障.例如,在智能电网领域,学者们采用通过改进网络数据监测能力以提升电力网络故障检测的效率[19],还有学者通过电力网络故障预测以提前识别发生的故障[20],或通过缩短检测模型的训练时间以提升检测效率[21].在云计算领域,学者们将故障检测问题建模为线程序列模板匹配问题以提高故障检测效率,或基于相似性图在检测故障的同时保障检测效率[22].这些研究大多关注于对故障检测效率的提升,只能快速判断出系统是否发生了故障,而不能快速识别出发生的具体故障信息.少部分故障预测方法的确可实现对识别效率的保障,但其只针对特定领域展开研究,迁移到其他领域应用时存在一定困难.

综上所述,现有自适应分析方法主要侧重于实现事件识别功能而忽略了对识别质量的保障.自适应领域少数考虑保障识别准确性的研究工作,大多侧重于对映射关系进行模糊化等优化操作,并未从根本上解决在不确定环境中映射关系难以完全预定义、正确性难以保证等问题.本文率先引入事件关系作为自适应分析方法识别事件的新因素,有助于保障识别过程的准确性.在保障识别效率方面,自适应领域缺乏相关工作且其他领域的特定方法无法完全适用.因此,本文提出尽早获取识别事件所需的重要状态数据,从而保障识别效率的研究思路.

2 方法概览

首先,介绍 SAFER 中涉及到的自适应软件领域基础概念.

定义 1(感知对象). 感知对象是指感知过程实时采集并获取状态数据的对象,如反映系统物理资源状态的 CPU、内存、磁盘等;反映系统网络资源状态、网络性能的 I/O 负载等;反映服务状态的响应时间等.通过采集感知对象的实时状态数据可全面且准确地判断当前运行环境是否发生了软件变化,因此感知对象也可称作感知指标.

定义 2(软件变化)[1]. 软件变化是指运行环境发生的各类状态变化,可通过分析感知对象的状态数据变化情况获得,具体包括网络环境的波动、系统计算资源的状态变化等.由于环境可能存在的瞬时波动以及调整策略执行后对运行环境的影响,软件变化的发生不一定说明发生了导致系统需要进行调整的异常事件,因此需要分析过程进行事件识别.

定义 3(异常事件)[8]. 异常事件,简称事件,是指由一个或多个软件变化衍变而来,并且影响系统目标实现的异常或者突发情况,可分为软件资源异常、硬件资源异常、应用失效、网络连接异常等类型.其中,如软件资源异常类型中就包括服务无响应、服务资源过载、服务出错率较高等具体事件.

定义 4(事件关系). 事件关系是指不同事件之间存在的关联关系,包括事件并发关系、事件时序关系和事件因果关系等.以二元事件关系为例,事件并发关系 $CR'(e_1,e_2)$是指事件 e_1和 e_2同时发生,但不强调二者之间的先后顺序.事件时序关系 $TR(e_1,e_2)$是指 e_2 在 e_1 发生后设定的时间间隔内发生.事件因果关系 $CR(e_1,e_2)$是指 e_1 的发生导致了 e_2 的发生.若满足转化约束:$TR(e_1,e_2)$发生的频率远大于 $TR(e_2,e_1)$发生的频率,则表明 e_1 和 e_2 之间存在因果关系[26],$TR(e_1,e_2)$可转化为 $CR(e_1,e_2)$.

定义 5(数据采集系统)[27−29]. 数据采集系统是指感知过程定期采集感知对象状态数据所采用的系统,例如用于监测负载变化情况的 Ganglia 系统[27],用于监测网络、主机运行状态的 JMeter 系统[28],用于监测物理基础设施资源、虚拟资源的 DARGOS 系统[29]等.

定义 6(感知周期)[30]. 感知周期是指数据采集系统获取感知对象状态数据的间隔时间.

其次,给出 SAFER 实现事件识别及识别过程质量保障的工作过程.如图 1 所示,其主要包含 3 部分工作:① 事件因果关系挖掘与建模;② 基于正向推理的事件识别;③ 基于反向推理的变化监测.其中,工作①作为工作②和③的基础,负责从系统运行日志中挖掘形成事件因果关系.工作②和③则依据该因果关系,识别事件并对识别准确性和识别效率加以保障.

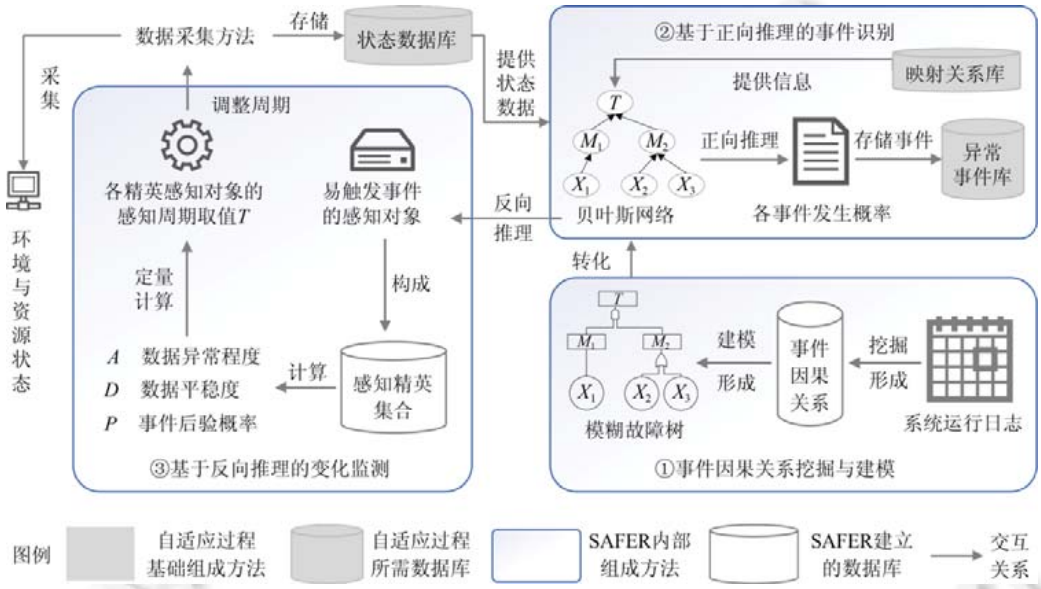


Fig.1 Overview of SAFER

图 1 SAFER 概览

① 事件因果关系挖掘与建模.通过序列模式挖掘算法从系统运行日志中挖掘得到事件因果关系;建立模糊故障树模型以树状图结构和模糊算子描述事件因果关系,并将模糊故障树交于领域专家核查与增补,避免仅依赖系统运行日志或领域知识建立事件因果关系而导致的局限性与不准确性.

② 基于正向推理的事件识别.将模糊故障树通过转换规则转换为贝叶斯网络;根据软件变化与贝叶斯网络中基本事件间的映射关系,判断基本事件的发生情况;利用贝叶斯网络的正向推理能力与基本事件发生情况,计算中间事件至顶事件的发生概率,以此判断各层事件发生情况并保障识别的准确性.

③ 基于反向推理的变化监测.基于贝叶斯网络的反向推理能力和基本事件结构重要度,定量地计算与各类感知对象相关的软件变化对事件发生的影响程度,抽取其中易造成事件的感知对象形成精英感知集合;量化计算精英感知集合中感知对象的状态数据平稳度、状态异常程度及引发事件的后验概率,动态地调整这些感知对象的感知周期,以快速发现软件变化从而保障识别效率,并避免持续高频采样带来的巨大计算和存储开销.

基于上述工作,SAFER 可为自适应软件识别出运行环境或系统资源中发生的事件,并能够保障识别准确性与效率,为确保自适应软件长期且稳定的运行提供有力的支持.

3 基于事件关系保障识别质量的分析方法

3.1 事件关系挖掘与建模

(1) 事件关系挖掘

在各类事件关系中,本文选用能够反映事件间触发情况的因果关系进行事件识别,从而在某一事件发生后可推理出与其相关的后续事件是否会发生.系统运行日志中一般仅记录系统的历史运行状态,从中较难直接挖掘出事件因果关系.但日志记录了各事件的具体发生时间,为抽取事件时序关系提供了可能.因此,本文采用序列模式挖掘算法对系统运行日志进行挖掘,得到事件时序关系,并根据定义 4 将满足转化约束的时序关系转换为因果关系.

序列模式挖掘中常用的算法包括 GSP 算法(generalized sequential pattern mining algorithm)、AprioriAll 算法等,它们通过一系列的序列连接、剪切操作生成包含时序关系的序列.相较于 AprioriAll 算法,GSP 算法引入了时间约束,通过连接和剪切可有针对性地生成序列,算法执行效率较高.因此,本文建立了基于 GSP 算法的事件关系挖掘方法,抽取出潜藏在系统运行日志中的事件因果关系.

定义 8(事件项目). GSP 算法中的事件项目是序列中的最小组成单位,简称项目,记作 i_j,j=1…n.本文将系统资源、环境变量或软件性能指标等超过阈值的异常环境状态作为事件项目.例如,i_1 表示 CPU 使用率超过阈值,i_2 表示网络带宽超过阈值,i_3 表示响应时间超过阈值.

定义 9(项目集). 一个或多个事件项目的集簇组成 GSP 算法中的项目集,简称元素,记作$\{i_1,i_2,\ldots,i_n\}$.

定义 10(元素时间窗大小). 元素中最晚出现的事件项目 i_n 与最先出现的事件项目 i_1 之间的最大允许时间差,记作 ElementTimeWindow,简称 ETW.

定义 11(序列). 不同项目集的有序列表组成 GSP 算法中的序列,记作 $s=\langle\{i_1,i_2\}\{i_3,i_4,i_5\}\{i_6,i_7\}\ldots\{i_p,i_q\}\rangle$.$k$-序列是指序列中包含 k 个事件项目,该序列的长度为 k,如$\langle\{i_1,i_2\}\{i_3,i_4,i_5\}\rangle$表示 5-序列.

定义 12(序列时间窗大小). 序列中最晚出现的事件项目 i_q 与最先出现的事件项目 i_1 之间的时间间隔,记作 SequenceTimeWindow,简称 STW.

定义 13(序列数据库). 多条序列形成 GSP 算法中的序列数据库,记作 S.

定义 14(支持度). 序列 s 的支持度是指序列数据库 S 中包含序列 s 的序列个数.

定义 15(候选序列). 长度为 i 的序列经过连接和剪切操作产生长度为 i+1 的序列,这些序列称为候选序列.

定义 16(频繁序列). 给定最小支持度 minsup,若候选序列的支持度超过最小支持度,则为频繁序列.

本文主要从系统运行日志中挖掘获得事件因果关系.因此,在收集得到系统运行日志后,本文首先对其进行去噪声处理和去缺失处理;然后进行时间约束处理,通过设置元素的时间窗大小 ETW 以及序列的时间窗大小 STW,将日志数据划分为序列,以得到序列数据库 S.因为只有符合时间约束的序列包含的事件项目才具有事件相关性[31],才能使得挖掘结果更符合事件发生的特点.序列数据库 S 则作为本文基于 GSP 算法的事件关系挖掘算法的输入.下面介绍该算法的具体工作过程.

算法 1. 基于 GSP 算法的事件关系挖掘.

输入:序列数据库 S,元素时间窗大小 ETW,序列时间窗大小 STW,最小支持度 minsup;

输出:事件因果关系表 CRT.

1. $C_1=init(S)$ //得到候选 1-序列,为初始种子集
2. $n=S.number$ //S 中序列的数量
3. $F_1=\{f|f\in C_1,f.count/n\geq \text{minsup}\}$ //得到频繁 1-序列
4. for (k=2;$F_{k-1}\neq\varnothing$;k++)
5. $C_k=JoinAndPrune(F_{k-1})$ //连接和剪切生成候选 k-序列
6. for each $s\in S$
7. for $c\in C_k$

8. *addCount*(*c*) //候选 *k*-序列支持度计数
9. endfor
10. endfor
11. if (*c.count*/*n*≥minsup) //判断是否为频繁序列
12. *SupportCount*(*b*|*c*∈C_k,*b*=*backward*(*c*)) //统计频繁序列逆序后的支持度
13. *CRT*=*changToCausal*() //将频繁序列转换为因果关系
14. F_k←*c*,*CRT*←*b*
15. endif
16. endfor

上述基于 GSP 算法的事件关系挖掘方法的相关描述如下.

- 第 5 行中的连接操作为,序列 S_1 与 S_2 连接,当且仅当从 S_1 中去掉第 1 个事件得到的子序列和从 S_2 中去掉最后一个事件得到的子序列相同时,才进行连接操作,此时可能存在两种情况:① 若 S_2 的最后两个事件属于同一个元素,则 S_2 的最后一个事件与 S_1 的最后一个元素组合成为合并后的最后一个元素.例如,⟨(1)(5)(3)⟩与⟨(5)(3 4)⟩连接结果为⟨(1)(5)(3 4)⟩.② 若 S_2 的最后两个事件属于不同的元素,则 S_2 的最后一个事件在合并后的序列中成为连接到 S_1 尾部的单独元素.例如,⟨(1)(2)(3)⟩与⟨(2)(3)(4)⟩连接结果为⟨(1)(2)(3)(4)⟩.
- 第 5 行中的剪切操作为,若某候选序列模式的任意一个子序列不是序列模式,则此候选序列模式不可能是序列模式,将其从候选序列模式中删除.
- 第 13 行根据定义 4,将频繁序列转换为因果关系.
- 最终此方法的输出结果为事件因果关系表,由挖掘得到的各个事件因果关系组成,例如系统中服务任务量较重→对应功能的响应时间变长等.

(2) 事件关系建模

为了能够基于事件因果关系识别事件,需清晰、直观地建模该类关系.事件因果关系一般为单向关系,由底层基本事件指向高层复杂事件,适于采用树状图进行表示.高层复杂事件可作为树状图的最顶端事件,然后将事件因果关系以父子层次结构进行组织与枚举.

故障树采用逻辑门连接上层与下层事件之间的因果关系,能够直观地表达本文抽取出的事件因果关系.但是故障树在应用时存在一定的问题,故障树逻辑门需要获取最下层事件(即基本事件)发生概率的精确值才能向上层推理,并且基本事件发生概率的准确性影响着整个故障树向上推理的准确性.然而,在不确定环境下,这种概率的精确值较难准确获取.为了解决这一问题,本文在采用模糊故障树建模事件因果关系时,以模糊数形式表示事件发生概率,从而无需获取事件概率的精确值.并且,本文以模糊算子代替一般故障树中采用的传统逻辑门算子,可表达事件间逻辑关系的不确定性,更适用于表征不确定环境中事件的因果关系.

具体地,本文首先将事件因果关系建模成模糊故障树中的树状图结构;然后计算模糊故障树中基本事件的发生概率,将其映射为预定义的模糊数;最后根据模糊算子计算基本事件上层事件(即中间事件)的发生概率,直至最上层事件(即顶事件)的发生概率,至此,模糊故障树中所有的模糊算子计算完毕,为后续计算贝叶斯网络中的条件概率提供数据支持.过程如下.

事件因果关系建模为树状图.事件因果关系 *CR*(*a*,*b*)的建模方式如下:将 *a*、*b* 分别作为下层事件与上层事件,将 *a*、*b* 之间的因果联系采用上下层事件间的模糊逻辑门来连接.以这种方式逐层地向上形成一个树状图,最上层的顶事件则为系统中发生的各个故障类型,比如服务资源出现故障问题、节点资源出现故障问题和应用失效问题等.

基本事件发生概率的计算及映射.本文通过统计日志的方式获取基本事件发生频率,以频率代替概率的方式得到基本事件发生概率,并将发生概率映射为三角模糊数,这种方式允许基本事件发生概率值存在一定误差,对概率值的精确程度要求不高.基本事件发生概率对应的隶属函数可见公式(1).通过该公式可计算得到基本事件的发生概率 *x* 隶属于各个预设的模糊数 $\tilde{P}$ 的隶属度,最终,本文选取隶属度最大的模糊数作为基本事件发生

概率对应的模糊数. $\tilde{P}=(m-\alpha,m,m+\beta)$ 表示预先设定的模糊数,m、α、β分别为 $\tilde{P}$ 的均值、上界和下界元素. $\tilde{P}$ 表示基本事件发生概率在 $m-\alpha$和 $m+\beta$之间,但为 m 的可能性最大.

$$\mu_{\tilde{P}}(x)=\begin{cases}0, & \text{其他}\\ 1-(m-x)/\alpha, & m-\alpha\leqslant x\leqslant m\\ 1-(x-m)/\beta, & m<x\leqslant m+\beta\end{cases} \tag{1}$$

模糊算子计算.模糊故障树中事件之间连接的模糊逻辑门分为“与”门和“或”门.“与”门表示下层所有事件均发生时上层事件才会发生,“或”门表示任意一个下层事件发生,上层事件都会发生.根据模糊数的运算法则,模糊故障树的“与”门算子如公式(2),“或”门算子如公式(3).这两种逻辑门算子本质上也代表着该逻辑门连接的上层事件的模糊数,该层模糊数继续作为公式(2)和公式(3)的输入,将得到该层事件与上层事件连接的逻辑门算子,这样逐层向上计算,可得模糊故障树中各个模糊逻辑门算子,为后续计算贝叶斯网络的条件概率提供支持. 其中, $\tilde{P}_i$ 表示各个基本事件的模糊数,为公式(2)和公式(3)的输入,输出为“与”门算子 $\tilde{P}_{AND}$ 和“或”门算子 $\tilde{P}_{OR}$.

$$\tilde{P}_{\mathrm{AND}}=\prod_{i=1}^{n}\tilde{P}_i=\left(\prod_{i=1}^{n}(m_i-\alpha_i),\prod_{i=1}^{n}m_i,\prod_{i=1}^{n}(m_i+\beta_i)\right) \tag{2}$$

$$\tilde{P}_{\mathrm{OR}}=1-\prod_{i=1}^{n}(1-\tilde{P}_i)=\left(1-\prod_{i=1}^{n}(1-m_i+\alpha_i),1-\prod_{i=1}^{n}(1-m_i),1-\prod_{i=1}^{n}(1-m_i-\beta_i)\right) \tag{3}$$

下面以“软件系统节点资源出现问题”为例,介绍基于模糊故障树的事件因果关系模型,如图 2 与表 1 所示.

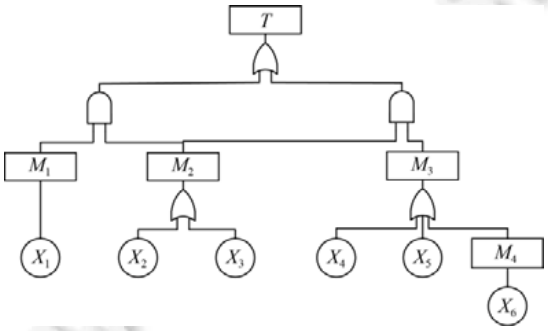


Fig.2 “Node Resource Problem” fuzzy fault tree

图 2 “节点资源出现问题”模糊故障树

Table 1 Symbolic representation

表 1 符号表示

符号	事件名称
T	节点资源出现问题
M_1	节点失联
M_2	多个业务功能受到严重影响
M_3	硬件故障
M_4	上层交换机故障
X_1	心跳间隔达到阈值
X_2	页面响应超时
X_3	页面错误率较高
X_4	网卡故障
X_5	网线故障
X_6	丢包率较高

如图 2 与表 1 所示,在这个模糊故障树中,X_1~X_6 为最底层的基本事件,根据挖掘得到的事件因果关系得出 X_1~X_6 在特定组合后,可能导致的上层事件分别为 M_1、M_2、M_3,其中,M_2 是指多个业务功能受到严重影响,可进一步划分为 M_{21}、M_{22} 等,分别表示 1 号业务受到影响、2 号业务受到影响等.根据事件因果关系,这些中间事件通过逻辑门的组合连接,最终导致顶事件 T,即节点资源出现问题的发生.

上述以一棵模糊故障树为例介绍事件关系模型,但在针对实际软件系统挖掘并建模事件因果关系时,本文从事件关系出发,同时以将复杂事件类型作为模糊故障树中顶事件为目标,建立形成多棵模糊故障树.

为了避免这些模糊故障树之间存在关联关系,即不同树之间存在事件相连的情况,本文进行了如下处理.模糊故障树关联的情况可分为两种:一是不同模糊故障树中的顶事件存在因果关系,此时可将它们合并成为一棵模糊故障树;二是一个模糊故障树中的基本事件连接至另一个模糊故障树的中间事件,此时可在这两个模糊故障树中分别建立这个基本事件节点,断开这种子树之间相连的边,两棵模糊故障树中中间事件互连的解决方法亦如此.通过上述处理,最终形成独立的多棵模糊故障树,每棵模糊故障树单独计算其模糊逻辑门算子,并且单独转换为贝叶斯网络,实现多棵模糊故障树分别独立、快速向上推理,降低事件推理过程的复杂性.

此外,考虑到自适应软件对自适应分析方法的高可靠性要求,本文采取了如下措施以确保模糊故障树本身

的正确性.首先,在模糊故障树建立之后,将交由领域专家核查其合理性和逻辑性,核查通过后才会用于事件识别.其次,本文认为静态事件因果关系模型已不能适应复杂多变的环境状态以及未知的软件变化.因此,为进一步确保正确性,本文将在系统运行过程中对事件因果关系模型进行动态修正与更新.模型的修正与更新时机可根据系统运行情况来决定,通常当系统运行一周时间后,或者系统运行日志累计到 50 000 条时对模型进行更新,这个运行时间以及运行日志累积量可根据系统实际运行情况进行调整与改变.在模型修正与更新时,本文将重新挖掘并探索新的事件因果关系,并重新统计基本事件发生的概率,在此基础上修改与完善事件因果关系模型,包括根据新的事件关系修改模糊故障树模型中的事件逻辑关系,将新的基本事件发生概率重新映射为模糊数,并计算模糊算子,使其能够适应并准确识别出不确定环境中发生的各类事件.

3.2 基于贝叶斯网络正向推理的事件识别

(1) 贝叶斯网络模型建立

本文进一步将模糊故障树转换为推理能力更强的贝叶斯网络,以利用贝叶斯网络的推理能力实现事件识别.如图 3 所示,本文将模糊故障树结构转换为贝叶斯网络的有向无环图.模糊故障树中的基本事件对应转换为贝叶斯网络中的父节点.中间事件和顶事件对应转换为子节点.若模糊故障树中存在多个相同的基本事件和中间事件,则贝叶斯网络中只需建立一个节点.模糊故障树中的逻辑门对应连接贝叶斯网络节点的有向边.

贝叶斯网络结构确立之后,需计算贝叶斯网络中各个父节点与其子节点之间的条件概率.传统贝叶斯网络中条件概率难以准确获取,目前的获取方式主要分为两种,一是专家确立,该方式存在一定的主观性;二是通过日志统计得到,但日志如果选取不当或者缺乏代表性,会使条件概率与真实情况相差较大,进而影响利用贝叶斯网络进行推理的准确性.本文则通过计算模糊故障树中的逻辑门算子,得到贝叶斯网络的条件概率,组成条件概率表,能够克服传统贝叶斯网络建模困难,即条件概率难以准确获取,进而影响贝叶斯网络准确推理的难题[32].

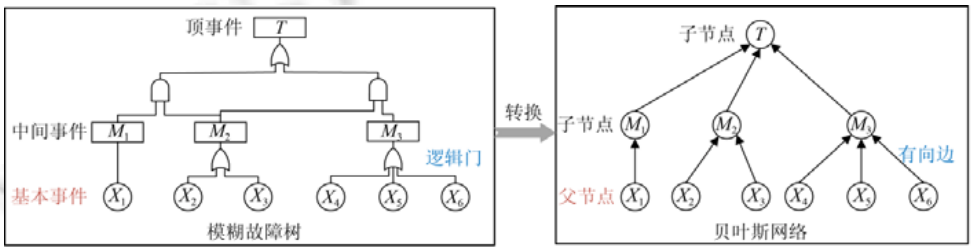


Fig.3 Conversion process from fuzzy fault tree to Bayesian network

图 3 模糊故障树向贝叶斯网络的转换过程

(2) 事件识别过程

本文通过贝叶斯网络正向推理能力识别事件.贝叶斯网络获取到环境状态与资源状态等输入数据,通过这些状态数据与基本事件之间的映射规则判断是否有基本事件发生,然后向上推理计算出中间事件和顶事件的发生概率,计算公式如式(4)所示,$P(upperEvent_i)$表示通过式(4)计算得到的第 i 个上层事件发生的概率,这里,中间事件和顶事件都为上层事件,$P(lowerEvent_s)$表示第 s 个下层事件发生的概率,$P(upperEvent_i|lowerEvent_s)$表示第 s 个下层事件导致其上层事件发生的条件概率,由模糊算子计算而来.这里的上层事件与下层事件是相对而言的,是指通过有向边连接的上层与下层事件.

$$P\left(upperEvent_i\right)=\sum_{s=j}^{k}P\left(lowerEvent_s\right)\times P\left(upperEvent_i\mid lowerEvent_s\right) \tag{4}$$

在向上推理的过程中,当基本事件 j 发生时,其将作为下层事件 $lowerEvent_j$($P(lowerEvent_j)$=1),此时获取 $P(upperEvent_i|lowerEvent_j)$以及与基本事件 j 位于同层并与其上层事件 i 有关联的基本事件 k 的先验概率 $P(lowerEvent_k)$以及 $P(upperEvent_i|lowerEvent_k)$,共同代入式(4)中,计算得到 $P(upperEvent_i)$.继续将 $upperEvent_i$ 作为下层事件 $lowerEvent_i$,代入式(4)中,重复这个过程,直到所有的上层事件发生概率都计算完成,即中间事件

和顶事件发生概率计算完成,最终根据这些概率得到发生的事件,输出事件名称、事件发生时间、触发该此事件的相关感知对象等信息,达到事件识别的目的.

以图2为例,假设基本事件X_4发生,即$lowerEvent_4$发生($P(lowerEvent_4=1)$),此时获取该事件导致其上层事件M_3发生的概率,即$P(upperEvent_3|lowerEvent_4)$,同时获取与$X_4$同层,并与$M_3$有关联的基本事件$X_5$和$X_6$的先验概率$P(lowerEvent_5)$、$P(lowerEvent_6)$以及条件概率$P(upperEvent_3|lowerEvent_5)$、$P(upperEvent_3|\ lowerEvent_6)$,通过式(4)计算出 M_3 发生的概率,即 $P(upperEvent_3)$.计算得到这个中间事件的概率后,同样继续获取与其位于同层,并与其上层事件有关联的M_2与M_1的发生概率,和M_2与M_1分别导致T发生的条件概率.其中,M_2与M_1的发生概率同样由式(4)获得.至此计算得到$P(M_3)$、$P(M_2)$、$P(M_1)$、$P(T)$的发生概率,根据这些概率得到发生的事件.

同样,为保证模型的可靠性,在模糊故障树模型依据运行环境进行动态调整后,本文会对贝叶斯网络模型进行修正与更新,包括调整事件间的逻辑关系,根据新的模糊算子重新计算条件概率等,确保模型能够适应最新的系统结构、环境以及随系统运行而发生的新类型事件.

此外,本文对系统资源、状态变化等基本事件进行了概率重要度与关键重要度分析,观察这些基本事件对顶事件发生的影响,使得软件维护人员可有针对性地采取修改与完善措施,通过降低这些基本事件的发生概率,实现上层事件发生概率的降低,最终实现确保自适应软件可靠、平稳运行的目标.具体分析过程如下.

- 概率重要度表示当基本事件发生与不发生相比时,顶事件发生概率的减小量.概率重要度越大,表示采取措施降低该基本事件发生的概率可以越迅速且有效地降低顶事件发生的概率.此时,软件维护人员可以采取有针对性的措施有效地降低顶事件发生的概率.
- 关键重要度表示基本事件发生概率的变化率与其引起顶事件发生概率的变化率之比.基本事件的关键重要度越大,表明该基本事件所在的部件越薄弱,更易发生异常事件.软件维护人员可通过改进关键重要度较大的部件,降低事件发生的频率,提升软件系统的稳定性和可靠性.

3.3 基于贝叶斯反向推理的变化监测方法

(1) 精英感知集合构建与管理

由于不同基本事件影响顶事件发生的条件概率不同,因此如果能够对条件概率较高,即易造成较大影响的基本事件进行尽早识别,就可在顶事件发生之前及时处理系统异常,从而避免造成重大损失.因此,本文构建了精英感知集合,存储与上述条件概率较高的基本事件相关的感知对象(称为精英感知对象),并可随贝叶斯网络的更新而调整集合内容.通过对该集合内感知对象的密切监控,则可尽快识别基本事件,保证识别效率.具体地,本文通过以下两种方式确定精英感知集合的范围.

首先,通过贝叶斯网络的反向推理能力,本文可计算得到各个基本事件的后验概率,即顶事件发生后,基本事件发生的概率,如公式(5)所示.其中,$P(basicEvent_j)$、$P(topEvent)$分别表示基本事件 j 和顶事件的先验概率,$P(topEvent|basicEvent_j)$表示在基本事件j发生的情况下,顶事件发生的概率,这些概率主要通过日志数据统计分析,以频率估计概率的方式获得.基本事件的后验概率大小,反映了它对顶层事件发生的影响程度.后验概率越大,证明它越可能导致顶事件的发生,因此本文选取与后验概率最大的基本事件相关的感知对象加入到精英感知集合,进行重点监控.

$$P\left(basicEvent_j \mid topEvent\right)=\frac{P\left(basicEvent_j\right)\times P\left(topEvent \mid basicEvent_j\right)}{P\left(topEvent\right)} \tag{5}$$

其次,结构重要度是在已知基本事件发生概率均相等的条件下,表示各项基本事件对顶事件影响概率的大小.结构重要度越大,表示该基本事件的发生对顶事件发生的影响越大.本文将与该基本事件相关的感知对象加入精英感知集合并对其重点监控,以及时、有效地识别事件.

(2) 感知周期动态调整

目前产业界获取软件变化的常用方法可分为两类.其一是采用固定的较小感知周期监测部分与软件变化相关的重要感知对象[33],其二是根据感知对象运行状态实时计算感知周期进行频繁采样.相较于第2类方法,第

1 类方法需始终高频采样感知对象,易造成感知数据量激增且带来巨大的数据处理与存储开销.因此,本文主要借鉴了第 2 类方法的思想,对精英感知对象的感知周期进行动态调整,以获取重要状态数据且不产生过大的系统开销.

目前第 2 类方法主要应用于云计算领域,调整感知周期采用的主要指标为数据平稳度,其反映了一段时间内环境状态数据的波动情况.数据平稳度越大,则运行环境越平稳,云计算领域的环境监测方法可增大感知周期而获得较少的状态数据刻画当前环境,减少采样开销.若平稳度越低,则说明环境波动越明显,监测方法则需减小感知周期对环境状态进行密集采样[34].然而,如果在自适应软件中仅考虑数据平稳度去调整感知周期,则易导致系统自主调整其结构或行为时出现的正常环境状态波动被误认为是发生了事件,而进行不必要的感知周期调整.

因此,在考虑数据平稳度的前提下,本文另外建立状态异常程度指标以评估系统当前的状态,用于判断当前环境状态的波动是正常波动还是发生了事件.同时,引入上节计算的基本事件后验概率去衡量精英感知对象发生的状态变化是否有较大概率造成更严重的顶事件.基于这 3 类计算指标,本文可实现对感知周期的实时调整,并有效避免仅依据数据平稳度调整感知周期的缺陷.

在实时调整精英感知集合中每个感知对象的感知周期时,需针对各个感知对象分别计算其各自的数据平稳度与后验概率.而由于状态异常程度评估的是软件系统的整体运行状态,因,该指标只需计算一次即可用于调整所有感知对象的感知周期.后文以某精英感知对象 s 的感知周期实时调整过程为例,介绍该过程包含的具体环节.

状态异常程度计算.为了计算软件系统运行状态的异常程度,首先需要对系统运行状态加以建模.考虑到自适应软件动态开放的运行环境和错综复杂的系统结构,系统的运行状态将包含多种属性,且属性间联系较少.因此,本文选取能够反映软件状态情况的代表性感知对象,如响应时间、内存负载等,对系统运行状态中各个属性进行清晰的描述.具体地,本文将系统运行状态建模为 $O=\{O_1,O_2,\ldots,O_N\}$,其中,$O_i(i=1,\ldots,N)$表示第 i 个感知对象,N 表示感知对象的数量.感知对象取值按照时间可形成集合 $D=\{d_1,d_2,\ldots,d_t,\ldots\}$.感知对象具体可分为两类,一种是数值型感知对象 $d_{ct}=oc1_{vt},oc2_{vt},\ldots,ocC_{vt}\}$,$oci_{vt}$ 表示第 i 个感知对象在 t 时刻的实时取值,例如内存使用量等;另一种是非数值型感知对象 $d_{dt}=\{od1_{vt},od2_{vt},\ldots,odD_{vt}\}$,例如服务功能是否正常等.

系统正常状态的阈值为 $d_n=\{o1_{vn},o2_{vn},\ldots,oN_{vn}\}$.$t$ 时刻的感知对象取值与系统正常状态间的偏离程度映射为两点之间的空间距离,两点之间的空间距离之和即为数据的异常程度,对于数值型感知对象在 t 时刻的状态异常度表示为 A_{t1},如式(6)所示.

$$A_{t1}=\sum_{k=1}^{c}\sqrt{\left(\max\left(ock_{vt}-ok_{vn},0\right)\right)^2} \tag{6}$$

对于非数值型感知对象,本文首先对其取值进行离散化处理,并定义第 k 个感知对象取值的异常度为 A_{dk},若符合系统正常状态的阈值要求,则取 0,否则,取值为 1.对于非数值型感知对象在 t 时刻的系统状态异常度表示为 A_{t2},如式(7)所示.

$$A_{t2}=\sum_{j=1}^{D}A_{dk}\ /\ D \tag{7}$$

根据式(6)和式(7),t 时刻系统状态的异常度定义为 A_t,如式(8)所示,其中,$\varepsilon1+\varepsilon2=1$.

$$A_t=\varepsilon1\times A_{t1}+\varepsilon2\times A_{t2} \tag{8}$$

数据平稳度计算.定义 t 时刻滑动窗口大小为 m,并判断感知对象 s 的取值是否为非数值型数据,若是,则进行离散化处理,之后汇集处理后的滑动窗口内所有感知数据,分别计算得出数据平均值 D_{ts1} 及数据变化量 D_{ts2},如式(9)、式(10)所示,并将计算结果代入式(11),得出某一感知对象 t 时刻感知数据平稳度 S_{ts}.

$$D_{ts1}=\left(d_{ts-m+1}+d_{ts-m+2}+\ldots+d_{ti}\right)/m \tag{9}$$

$$D_{ts2}=\left(a_m\left|d_{ts}-d_{ts-1}\right|+a_{m-1}\left|d_{ts-1}-d_{ts-2}\right|+\ldots+a_1\left|d_{ts-m+2}-d_{ts-m+1}\right|\right)/(m-1) \tag{10}$$

$$S_{ts}=D_{ts1}/D_{ts2} \tag{11}$$

周期调整系数计算.本文按照周期调整系数γ_{ts}调整感知对象 s 的感知周期,如式(12)所示.本文认为,不同基本事件引发顶事件的概率有所差异,针对有较高概率引发顶事件的基本事件,其相关感知对象的感知周期需对应缩短.其中,T_{re}、$T_{\min}$ 分别表示某一感知对象加入精英集合前的感知周期和最小感知周期,P 表示感知对象引发顶事件的概率.在一段时间内某个基本事件引发顶事件的概率是固定的,假设有一个基本事件 i,且 i 在 4 个时间段 a、b、c 和 d 内引发顶事件的概率分别为 0.1、0.2、0.8 和 0.9,且其实时感知周期分别为 T_{ta}、T_{tb}、T_{tc} 和 T_{td},计算发现$(T_{td}-T_{tc})/T_{tc}$大于$(T_{tb}-T_{ta})/T_{ta}$,即随着基本事件引发顶事件概率的增加,其感知周期缩短幅度也增大.

$$\gamma_{ts}=-(T_{re}-T_{\min})p^2+T_{re} \tag{12}$$

感知周期动态修正.本文根据状态异常程度、数据平稳度及感知对象变化引发顶事件的概率着 3 方面因素的实时计算结果可直接得出感知周期,实现对感知周期的动态修正,进而高效识别事件.其中,状态异常度和感知周期则为反向关系,状态异常程度越高,则感知周期越小,以尽快获取异常事件信息,否则可适当增加感知周期;数据平稳度和感知周期为正向关系,如果该段时间内数据平稳,则可增大感知周期,减少感知成本.如果该段时间内数据频繁发生变化,则需减小感知周期.感知对象变化引发顶事件的概率和感知周期为反向关系,对于有较大概率引发顶事件的基本事件,其相关感知对象需进行更为密切的监控.

但在系统实际运行过程中,为了防止频繁调整感知周期而造成过大的系统开销,SAFER 设定数据平稳度阈值为λ_1、状态异常程度阈值为λ_2.并计算数据平稳度、状态异常程度及周期调整系数,若数据平稳度、状态异常程度的实时计算值未超过设定阈值,则主要考虑基本事件引发顶事件的概率这一因素来计算相关感知对象的感知周期 T_{ts},即为$\gamma_{ts}T_{re}$;在数据平稳度或状态异常程度超出阈值时,则根据该感知周期动态调整方法计算并修改其感知周期.此外,本文将加入精英集合前的感知周期 T_{re} 设为默认感知周期.若计算出的感知周期小于最小感知周期,则用最小感知周期监测感知指标;若计算出的感知周期小于默认感知周期且大于最小感知周期,则用所计算出的感知周期监测感知指标;否则,用默认感知周期进行监测,如式(13)所示,γ_{ts} 表示周期调整系数,α和β作为平滑因子.T_{ts-1} 表示前一时刻的感知周期,S_{ts} 表示数据平稳度,A_t 表示状态异常程度.

$$T_{ts}=\begin{cases}T_{\min}, & T_{ts}\leqslant T_{\min}\\ \gamma_{ts}\times T_{ts-1}\times \mathrm{e}^{\frac{\alpha S_{ts}}{\beta A_t}}, & T_{\min}<T_{ts}<T_{re}\\ T_{re}, & T_{ts}\geqslant T_{re}\end{cases} \tag{13}$$

4 实验分析

本节介绍本文对 SAFER 开展的实验及结果分析工作.首先介绍实验的具体设计方法,然后给出实验的实际结果并进行分析讨论.

4.1 实验设计

基准方法的选择.本文选择将 SAFER 与目前自适应领域中常用的自适应分析方法进行比较,包括杨启亮团队提出的基于模糊规则推理的方法(以下简称模糊推理方法)[13]、Baader 提出的基于本体推理的方法(以下简称本体推理方法)[11]以及在云计算等其他领域中常用的基于概率推理的方法[14−16].其中,基于概率推理的方法中选取了最具代表性的 Zhang 采用的贝叶斯网络分析方法[15].选取上述方法的主要原因总结如下.

- 在自适应领域中,规则推理方法和本体推理方法是最为经典和常用的分析方法,而模糊推理方法是对规则推理方法的改进,使其能够应对众多环境状态带来的不确定性,相比规则推理方法,其准确性更高,因此,本文选择模糊推理方法和本体推理方法进行对比.
- 在云计算等其他领域中,主要包括基于日志分析、事件关系分析和概率推理的 3 类事件分析方法.现有的基于日志分析的方法大多关注检测日志序列,明确是否发生了系统异常,但未能检测出具体的事件信息,与 SAFER 的功能目标不同,因此不被选用作为对比对象.现有的基于事件关系分析的方法主要关注对网络安全事件、入侵事件等网络安全领域事件关系的分析,与自适应领域关注的事件类型不同.因此本文选用了基于概率

推理的方法中最具代表性的贝叶斯网络方法进行对比.

研究问题.为验证 SAFER 的有效性,本文尝试回答以下研究问题.

RQ1. SAFER 在识别事件上的准确性如何?是否做到了对识别准确性的保障?与现有其他分析方法相比,其识别准确性是否较高?

RQ2. SAFER 在识别事件上的效率如何?是否做到了对识别效率的保障?与现有其他识别方法相比,其识别效率是否较高?

实验数据集.自适应领域内研究人员普遍采集案例系统的运行数据,以构建测试数据集开展测试工作,但并未对数据集进行公开发布[35,36].因此,鉴于无法获取恰当的公开测试数据集,本文同样使用满足条件的真实系统作为案例系统,并将其运行时产生的实时数据作为实验数据集进行测试工作.SAFER 的测试需求决定了案例系统需运行在不确定环境下,且具备自适应能力,并支持人为故障的注入.本团队前期开发的 BookStore 网上书城系统[37,38]满足该测试需求,将作为案例系统对 SAFER 进行测试.

BookStore 作为一个典型的 Web 应用软件,主要包含账户管理服务、商品管理服务、购物服务和推荐服务等多种服务.各个服务间通过交互协作共同向用户提供多种完整功能,如展示、浏览、支付、管理、广告等 9 项功能.该系统在运行过程中可能会发生部署集群节点损坏、服务故障、响应超时、网络故障等多种类型的软件变化,这些不同的软件变化还有可能进一步引发未知的软件变化.并且,这些软件变化是随时发生的,发生时间无法预测.BookStore 系统支持故障的人为注入,例如关闭相应节点模拟页面无响应事件,使不确定环境导致的事件发生时间可受人为控制,从而使测试效果更加明显.因此,该系统满足运行在不确定环境的条件,足以支持本文实验的进行.

BookStore 系统拥有基本的自适应控制系统,其中拥有轻量级的自适应感知-分析-决策-执行控制循环,在系统离线状态下,可对其中的自适应分析方法进行修改,为本文不同分析方法的测试提供有力支持;在系统运行过程中,自适应感知环节采用 Ganglia 系统[27]监测节点的 CPU、内存、磁盘利用率等信息,采用 JMeter 系统[28]监测服务页面响应时间、服务页面错误率等信息,并以日志的形式记录运行环境状态,支持人为批量提取,为本文测试需要的数据集构造提供了极大便利.

表 2 为获取的 BookStore 中商品服务的运行日志片段示例,展示了该服务运行中的各个属性,除心跳返回时间外,其他属性均为 3s 采集 1 次.商品服务页面平均响应时间是指从请求服务到返回数据所需响应时间,单位为 s;平均错误率是指商品页面显示的错误率,以百分号为单位;商品服务请求量是指请求该服务的用户数,通过 JMeter 设置访问线程数来模拟访问用户数;商品服务心跳返回时间是指自适应控制系统每 30s 向各个服务发送心跳检测,该商品服务向自适应控制系统返回心跳的时间,单位为 s,这个时间随每次自适应控制系统发送心跳检测的时间而发生变化,即每 30s 或超过 30s 变化 1 次;节点 CPU 利用率和内存使用率是指商品服务部署节点的 CPU 和内存利用率,由 Ganglia 系统监测而来,以百分数形式存储.

本文收集了 BookStore 系统运行日志,并人工标注了日志中的事件,作为测试 SAFER 识别准确性的数据集.具体地,本文采集了 BookStore 运行一天所产生的日志数据,由于该系统运行过程中事件的发生是较为稀疏的,不利于本文测试方法的有效性,因此本文在这一天内对该系统不定期地实现了 20 次故障注入.注入故障的方式包括通过压力测试工具 JMeter 模拟高并发访问引发服务超载,关闭相应节点引发页面无响应,撤销服务引发软件应用失效等.在收集到注入故障的日志数据后,本文手动标记了日志数据中发生的事件,并选取事件发生较为密集的 6 个日志片段代表系统运行时事件发生情况,这 6 个日志片段作为 6 个不同大小的数据集,称为数据集 1~6,大小分别为 1 500 条、1 800 条、2 100 条、2 400 条、2 700 条、3 000 条.

评测指标.针对识别质量,本文通过以下指标[15,17]评测不同的自适应分析方法.

- 准确率(accuracy):正确识别出的已标记事件与未识别出的未标记事件之和占数据总数的比例.
- 召回率(recall):正确识别出的已标记事件占所有已标记事件的比例.
- 识别效率(time efficiency):从注入故障到识别出具体事件所花费的时间.

运行环境配置.完成实验的计算机集群包括运行 SAFER 等方法所提供的自适应控制的主控计算机、提供

不同服务的计算机和访问的客户端计算机共8台.其配置信息均为CPU Inter(R) Core(TM) i5-4570,内存为8GB,操作系统是Windows 64位.

Table 2 Running log snippet display

表 2 运行日志片段示例

序号	商品服务页面响应时间(s)	商品服务页面错误率(%)	商品服务请求量	商品服务心跳返回时间(s)	节点CPU利用率(%)	节点内存使用率(%)	时间戳
1	1.69	0	865	34	69	81	15:41:50
2	2.04	2	709	34	70	80	15:41:53
3	1.45	0	945	34	61	80	15:41:56
4	0.61	0	1 078	34	74	78	15:41:59
5	1.57	0	937	34	77	75	15:42:02
6	3.83	9	1 434	34	82	77	15:42:05
7	4.13	4	1 574	34	85	80	15:42:08
8	3.94	0	1 359	34	84	79	15:42:11
9	3.59	0	1 347	34	87	83	15:42:14
…	…	…	…	…	…	…	…

4.2 算法有效性测试

(1) 识别准确性测试(RQ1)

本文在前述数据集1~6下测试了SAFER的识别准确性.首先挖掘并建模事件关系,然后将其转换为贝叶斯网络模型并进行可视化展示,在此基础上测试了贝叶斯网络模型识别事件的准确性.具体如下.

首先,本文采用事件关系挖掘方法在数据集1~6中挖掘事件关系.通过重复实验,本文将元素的时间窗大小ETW设为5s,序列时间窗大小STW设为20s,最小支持度minsup设为9,以获得有价值的挖掘效果.挖掘结果见表3,本文以表中的第1条事件关系为例展开介绍.若服务多次调用系统API接口失败,服务可能不会获取到所需的数据,或是影响服务的功能实现,因此对应页面的数据展示或者功能将会有一个较高的错误率.

Table 3 Mined event relationships display

表 3 挖掘的事件关系展示

频繁序列	支持度	对应的因果关系
⟨{多次调用系统API接口失败},{对应页面的出错率较高}⟩	10	多次调用系统API接口失败||多次调用第三方组件失败→对应页面的出错率较高
⟨{多次调用第三方组件失败},{对应页面的出错率较高}⟩	9	
⟨{服务任务量较重},{对应功能的响应时间较长}⟩	23	服务任务量较重→对应功能的响应时间较长
⟨{服务心跳返回较慢},{对应页面响应非常慢甚至无响应}⟩	11	服务心跳返回较慢→对应页面响应非常慢甚至无响应
⟨{对应页面响应非常慢甚至无响应},{服务资源故障}⟩	13	对应页面的响应非常慢甚至无响应→服务资源故障
⟨{页面出错率较高},{服务资源过载}⟩	11	页面出错率较高&&页面响应较慢→服务资源过载
⟨{页面响应较慢},{服务资源过载}⟩	16	

其次,将上述挖掘得到的事件关系建模为模糊故障树,其中的顶事件分别设为“服务资源出现问题”“节点资源出现问题”“软件应用失效”等,图4和表4以顶事件为“服务资源出现问题”为例介绍事件关系模型.

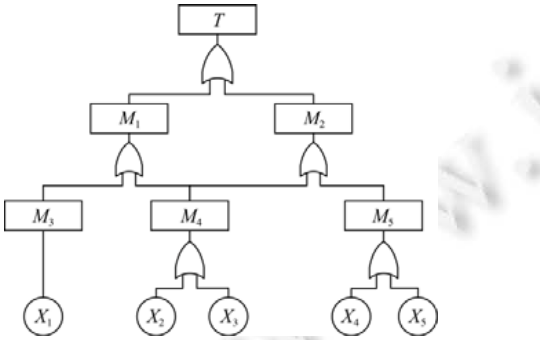


Fig.4 “Service Resource Problem” fuzzy fault tree

图4 “服务资源出现问题”模糊故障树

Table 4 Symbolic representation

表 4 符号表示

符号	事件名称
T	服务资源出现问题
M_1	服务资源故障
M_2	服务资源过载
M_3	页面响应超级慢甚至无响应
M_4	页面出错率较高
M_5	页面响应较慢
X_1	服务心跳返回非常慢
X_2	多次API调用失败
X_3	多次第三方调用失败
X_4	服务的任务量较重
X_5	服务心跳返回较慢

从图 4 和表 4 可以看出,模糊故障树中基本事件为 X_1、X_2、X_3、X_4、X_5,这些基本事件单独或者通过逻辑组合会导致其上层事件的发生,例如,表 3 中第 1 条事件因果关系中 X_2“或”X_3 的发生会导致 M_4 的发生.同样,上层事件通过逻辑组合后会导致更上层事件的发生,直至顶事件 T 的发生.这些因果关系被清晰、直观地以树状图结构加以展示,方便本文利用这些关系逐层向上推理,识别得到系统中发生的不同类型的事件.

再次,本文将“服务资源出现问题”“节点资源出现问题”“软件应用失效”等模糊故障树转换为贝叶斯网络,并以结构图形式展示.如图 5 所示,本文同样以“服务资源出现问题”模糊故障树为例,采用专家学者广泛使用的微软公司 GeNIe 软件将转换后的“服务资源出现问题”贝叶斯网络进行可视化展示,这个贝叶斯网络包含了 5 个父节点、6 个子节点,父子节点之间具备关联强度,通过关联强度可以向上推理出发生的事件.

通过上述步骤得到贝叶斯网络模型后,本文将测试模型的准确性.本文获取系统运行日志中最新的两条数据,根据映射规则判断“服务资源出现问题”“节点资源出现问题”“软件应用失效”等贝叶斯网络中的父节点事件是否发生.若发生,则进行该贝叶斯网络的向上推理过程,计算出此贝叶斯网络中各个子节点的发生概率.然后汇总各个贝叶斯网络计算得到的子节点发生概率,通过这些发生概率得到系统中大概率发生的事件,并与人工标记的事件进行核对,以验证模型的准确性.同样以“服务资源出现问题”贝叶斯网络为例,根据映射规则判断出父节点中“多次调用 API 失败”“服务任务量较重”“服务心跳返回较慢”已经发生,经过正向推理,获得各个子节点的发生概率,见表 5.除上述父节点事件发生以外,系统中还发生的事件为服务资源出现问题中的服务资源过载,事件表征为页面响应较慢,与人工标记的事件一致,证明文中方法可准确识别事件.

本文在验证贝叶斯网络单次识别事件的准确性之后,还在数据集 1~6 中,测试了 SAFER 与模糊推理方法、本体推理方法、贝叶斯网络分析方法的准确率与召回率,测试结果如图 6 所示.

从测试结果可以看出,SAFER 在各个数据集上均具有较好的识别准确性与全面性,相对于没有利用事件因果关系进行推理的本体推理方法、模糊推理方法以及贝叶斯网络这 3 种方法,SAFER 方法在准确率上平均提升了 3.927%、5.982%和 2.68%,在召回率上平均提升了 9.338%、12.37%和 7.09%.这一结果说明:在系统环境状态数据的基础上,加入事件之间的因果关系共同推理,能够根据事件之间的联系更多地发现一些事件表征背后的、不能被直接检测到的复杂事件,因此与仅依靠映射关系的分析方法相比,对召回率的提升更大.

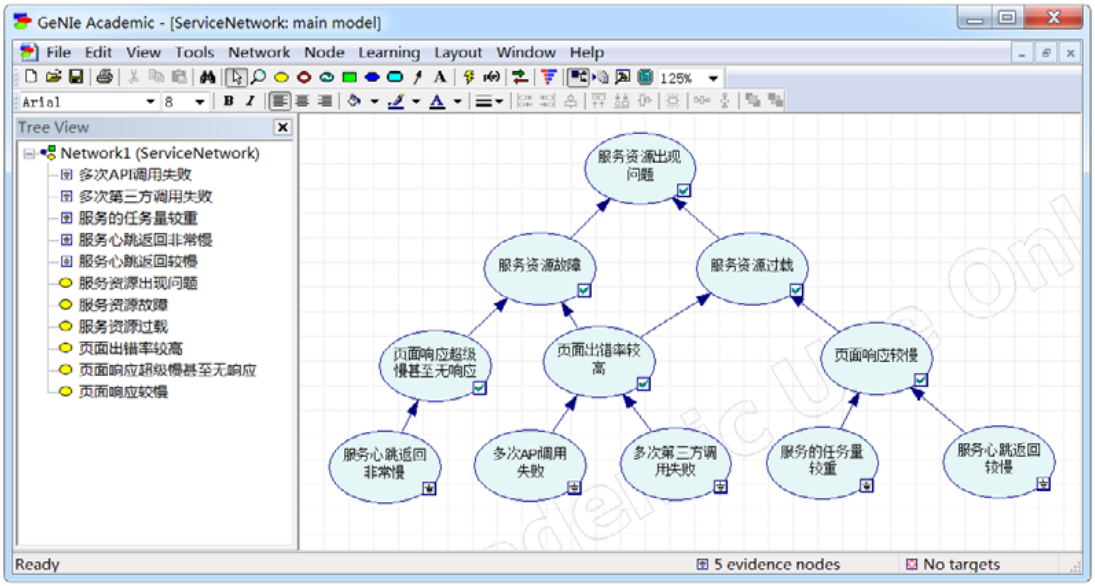


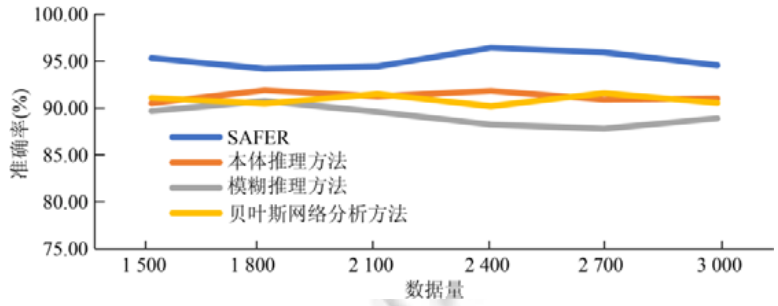
Fig.5 “Secvice Resource Problem” Bayesian network

图 5 “服务出现问题”贝叶斯网络示例

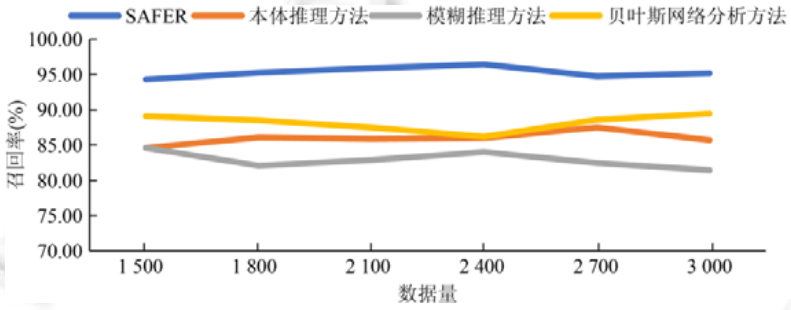
Table 5 Event probability table

表 5 事件发生概率表

事件名称	发生概率
页面响应超级慢甚至无响应	0.166 7
页面出错率较高	0.3
页面响应较慢	0.667
服务资源故障	0.367 5
服务资源过载	0.406 4
服务资源出现问题	0.566 7



(a) 准确率对比



(b) 召回率对比

Fig.6 Accuracy and recall of the SAFER and other methods

图 6 SAFER 与其他方法的准确率与召回率对比

(2) 识别效率测试(RQ2)

本文在系统运行过程中测试了 SAFER 的识别效率.首先完成贝叶斯网络反向推理,然后产生精英感知集合,在此基础上测试了 SAFER 识别事件的效率.具体如下.

通过对 BookStore 系统中的所有顶事件进行逐一反向推理,最终形成精英感知集合,见表 6.

Table 6 Elitist sense set

表 6 精英感知集合

序号	名称
001	网络服务状态
012	服务任务量
017	网络 I/O
009	内存利用率
…	…

表 6 中,每行代表一个由顶事件反向推理出的最易造成该顶事件的基本事件的相关感知对象.本节同样以“服务资源出现问题”顶事件为例详细介绍精英感知集合构建过程.通过贝叶斯网络反向推理能力可计算出所有与“服务资源出现问题”这一顶事件相关的基本事件的后验概率,如图 7 所示.其中,“服务任务量较重”这个基

本事件的后验概率最大,为 0.143.因此,该事件对应的感知对象“服务任务量”将被加入精英感知集合,以保障识别效率.

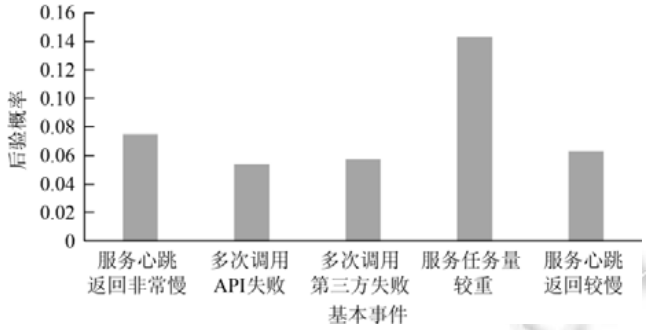


Fig.7 Posterior probability of the basic events

图 7 基本事件的后验概率

为了测试 SAFER 的识别效率,本文设定滑动窗口大小为 5,数据平稳度阈值为 0.58,状态异常程度阈值为 0.62,平滑因子α和β分别取值 0.4 和 0.6,上述取值是综合考虑了 SAFER 的方法开销、系统可接受的最大异常程度等因素,并在 BookStore 系统中进行多次实验后确定的最佳取值.本文建议在其他系统中应用 SAFER 时,可根据不同案例系统对平稳度、异常程度、运行开销等多方面因素的实际需求进行取值调整,以获得最佳效果.

本文具体测试了 SAFER 与现有模糊推理方法、本体推理方法和贝叶斯网络分析方法的识别效率.在 BookStore 系统实时运行过程中,以随机注入不同故障的方式进行 10 次重复实验并统计不同方法识别事件的平均时间,实验结果如图 8 所示.SAFER 对于异常事件的平均识别时间为 7.624s,而模糊推理方法、本体推理方法和贝叶斯网络分析方法的平均识别时间分别为 8.435s、8.721s 和 8.639s,说明 SAFER 在识别效率方面有所提升.但在第 5 次、第 6 次及第 9 次实验中,SAFER 与其余 3 种事件识别方法的识别效率基本持平,这是因为,引发这 3 次实验所注入事件的基本事件是不常见的(如 API 调用失败等),因此与之相关的感知对象并未加入到精英感知集合中进行重点监控.然而在系统实际运行过程中,常见事件的发生概率是远高于不常见事件的,而最易引发常见事件的相关感知对象已被密切监控,因而 SAFER 的优势会较为明显.

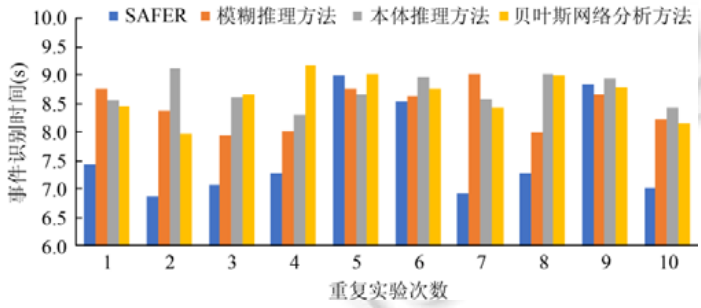


Fig.8 Comparison of the time efficiency between SAFER and other methods

图 8 SAFER 与其他方法的识别效率对比

由于识别效率包括感知数据采集与处理、事件识别两部分的效率,上述实验结果只能说明 SAFER 的识别效率较其他方法有所提升,无法明确识别效率提升的具体原因,因此本文补充测试了 SAFER 以及现有其他方法在第 2 部分,即在事件识别部分中所需时间.测试该时间不需要考虑感知数据采集与处理部分的耗时,因此本文在前述数据集 1~6 中记录了事件识别方法的耗时,具体如下:从未标记事件日志数据中第 1 条开始,识别每一条日志数据中发生的事件,记录其识别所需时间并进行累加,最终得到上述 4 种方法分别在数据集 1~6 中识别事

件所需要的时间.测试结果如图 9 所示,可以看出,SAFER、模糊推理方法和贝叶斯网络分析方法相比于本体推理方法,整体执行时间较短,且这 3 种方法执行时间相差较小,因此能够说明 SAFER 的整体识别效率是由于该方法中感知周期的动态调整,能够尽快获取到事件识别所需要的系统环境与状态等数据因而得到提升.

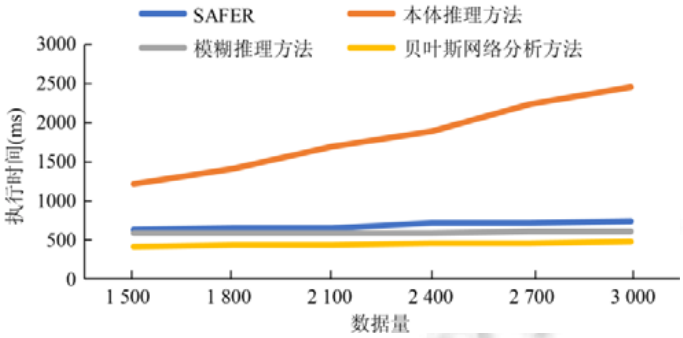


Fig.9 Execution time of SAFER and other methods

图 9 SAFER 与其他方法的执行时间对比

5 结论与展望

自适应软件的分析过程能够尽快产生正确的事件识别结果是确保该类软件长期稳定运行的关键因素之一.因此,本文提出了一种基于事件关系保障识别质量的自适应分析方法(SAFER),可通过序列模式挖掘算法挖掘事件因果关系,并基于贝叶斯网络利用事件因果关系和映射关系共同识别事件,保障了识别的准确性;可定量推理出易导致事件发生的感知对象,并调整感知周期更快地获取相应软件的变化情况,保障了识别效率.并且,SAFER 中构建的模糊故障树、贝叶斯网络、精英感知集合、感知周期等均可在软件运行过程中不断地自动更新修正,更具灵活性和可靠性.实验结果表明,当自适应软件采用 SAFER 后,不仅实现了对事件的识别,并且识别质量也得到了保障.此外,SAFER 识别事件所需时间开销较小,也更适合在实际工程中加以应用.

为进一步增强 SAFER 的能力,本文仍存在很多值得继续进行探讨的工作:(1) 本文后续考虑采用本体建模工具 protégé 建立反映领域专家知识经验的软件故障本体,并通过本体推理机制与 SWRL(semantic Web rule language)规则构建软件故障知识图谱,结合挖掘得到的事件因果关系,实现模糊故障树的自动构建.(2) 当前软件系统规模和环境的不确定性在不断上升,导致异常事件的类型及因果关系的数量也会随之不断增长.当贝叶斯网络的规模非常庞大时,如何保证概率计算的时间效率,是本文后续需考虑的一个问题.(3) 目前学术界提供的自适应软件运行日志、感知数据集合等标准测试数据集较少,我们希望能够从开源项目和商业项目中搜集更多的数据集,以验证本文所得出的实验结论是否具有一般性,并对 SAFER 中如贝叶斯网络更新周期、数据平稳度滑动窗口数量等参数的设置给出一些建议.

References:

[1] Krupitzer C, Roth FM, Vansyckel S, Schiele G, Becker C. A survey on engineering approaches for self-adaptive systems. Pervasive & Mobile Computing, 2015,17:184–206. [doi: 10.1016/j.pmcj.2014.09.009]

[2] Salehie M, Tahvildari L. Self-adaptive software: Landscape and research challenges. ACM Trans. on Autonomous and Adaptive Systems (TAAS 2009), 2009,4(2):1–42. [doi: 10.1145/1516533.1516538]

[3] Luckey M, Engels G. High-quality specification of self-adaptive software systems. In: Proc. of the 8th Int'l Symp. on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2013). San Francisco, 2013. 143–152.

[4] Zimmermann M, Wotawa F. An adaptive system for autonomous driving. Software Quality Journal, 2020,28(3):1189–1212. [doi: 10.1007/s11219-020-09519-w]

[5] Luckey M, Thanos C, Gerth C, Engels G. Multi-staged quality assurance for self-adaptive systems. In: Proc. of the 6th Int'l Conf. on Self-adaptive & Self-organizing Systems Workshops. IEEE, 2012. 111−118. [doi: 10.1109/SASOW.2012.28]

[6] Gabor T, Kiermeier M, Sedlmeier A, Kempter B, Klein C, Sauer H, Schmid R, Wieghardt J. Adapting quality assurance to adaptive systems: The scenario coevolution paradigm. In: Proc. of the Int'l Symp. on Leveraging Applications of Formal Methods (ISOLA 2018). Cham: Springer-Verlag, 2018. 137−157. [doi: 10.1007/978-3-030-03424-5_10]

[7] Wang ZY, Wang T, Zhang WB, Chen NJ, Zuo C. Fault diagnosis for microservices with execution trace monitoring. Ruan Jian Xue Bao/Journal of Software, 2017,28(6):1435−1454 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5223.htm [doi: 10.13328/j.cnki.jos.005223]

[8] Macias-Escriva FD, Haber R, Toro RD, Hernandez V. Self-adaptive systems: A survey of current approaches, research challenges and applications. Expert Systems with Applications, 2013,40(18):7267−7279. [doi: 10.1016/j.eswa.2013.07.033]

[9] Rodrigues A, Vogel T, Pelliccione P. A learning approach to enhance assurances for real-time self-adaptive systems. In: Proc. of the 13th Int'l Symp. on Software Engineering for Adaptive and Self-managing Systems. ACM, 2018. 206−216. [doi: 10.1145/3194133.3194147]

[10] Chaari T, Fakhfakh K. Semantic modeling and reasoning at runtime for autonomous systems engineering. In: Proc. of the 9th Int'l Conf. on Ubiquitous Intelligence and Computing and the 9th Int'l Conf. on Autonomic and Trusted Computing. IEEE, 2012. 415−422. [doi: 10.1109/UIC-ATC.2012.82]

[11] Baader F. Ontology-based monitoring of dynamic systems. In: Proc. of the 14th Int'l Conf. on the Principles of Knowledge Representation and Reasoning. AAAI Publications, 2014. 678−681.

[12] Paola AD. An ontology-based autonomic system for ambient intelligence scenarios. In: Proc. of the Advances onto the Internet of Things. Switzerland: Springer International Publishing, 2014. 1−17. [doi: 10.1007/978-3-319-03992-3_1]

[13] Yang QL, Lü J, Li JL, Ma XX, Song W, Zou Y. Toward a fuzzy control-based approach to design of self-adaptive software. In: Proc. of the 2nd Asia-Pacific Symp. on Internetware. 2010. 1−4. [doi: 10.1145/2020723.2020738]

[14] Wang T, Zhang WB, Xu JW, Wei J, Zhong H. A survey of fault detection for distributed software systems with statistical monitoring in cloud computing. Chinese Journal of Computers, 2017, 397−413 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2017.00397]

[15] Zhang J, Qi MC, Li B. Hybrid diagnosis of fault tree and Bayesian network in BIW automatic welding production line. In: Proc. of the Advanced Information Management, Communicates, Electronic & Automation Control Conf. IEEE, 2016. 297−300. [doi: 10.1109/IMCEC.2016.7867220]

[16] Chen HX, Ren Y, Yu SX. Fuzzy fault tree analysis on temperature control system of conventional rocket propellant. Yuhang Xuebao/Journal of Astronautics, 2017,38(1):104−108 (in Chinese with English abstract). [doi: 10.3873/j.issn.1000-1328.2017.01.014]

[17] Astekin M, Zengin H, Sözer H. DILAF: A framework for distributed analysis of large-scale system logs for anomaly detection. Software: Practice and Experience, 2019,49(2):153−170. [doi: 10.1002/spe.2653]

[18] He S, Lin Q, Lou JG, Zhang H, Zhang D. Identifying impactful service system problems via log analysis. In: Proc. of the 26th ACM Joint Meeting. ACM, 2018. 60−70. [doi: 10.1145/3236024.3236083]

[19] Li ZF, Pang HJ. Study on improved genetic algorithm based real-time detection system of network failure data. Modern Electronics Technique, 2018,41(13):138−141,146 (in Chinese with English abstract). [doi: 10.16652/j.issn.1004-373x.2018.13.031]

[20] Qu CY, Liu XQ, Xin P. Substation equipment failure state identify and prediction model based on hadoop. Software Guide, 2015(3):61−63 (in Chinese with English abstract). [doi: 10.11907/rjdk.1431012]

[21] Huang XD, Wang BY, Liu K, Liu TG. An event recognition scheme aiming to improve both accuracy and efficiency in optical fiber perimeter security system. Journal of Lightwave Technology, 2020,38(20):5783−5790.

[22] Yang YY. Automatic fault diagnosis approach based on similarity graph in cloud computing. Computer Measurement & Control. 2014,22(12):3877−3880 (in Chinese with English abstract). [doi: 10.3969/j.issn.1671-4598.2014.12.011]

[23] Shevtsov S, Berekmeri M, Weyns D, Maggio M. Control-theoretical software adaptation: A systematic literature review. IEEE Trans. on Software Engineering, 2018,44(8):784−810. [doi: 10.1109/TSE.2017.2704579]

[24] Ju AK, Guo YB, Zhu TM, Wang T. Survey on network security event correlation analysis methods and tools. Computer Science, 2017,44(2):38–45 (in Chinese with English abstract).

[25] Jing L, Gu L, Xu GS, Niu XX. A correlation analysis method of network security events based on rough set theory. In: Proc. of the Int'l Conf. on Network Infrastructure & Digital Content. IEEE, 2013. 517–520. [doi: 10.1109/ICNIDC.2012.6418807]

[26] Zhu B, Ghorbani AA. Alert correlation for extracting attack strategies. Int'l Journal of Network Security, 2006,3(3):244–258.

[27] Massie ML, Chun BN, Culler DE. The ganglia distributed monitoring system: Design, implementation, and experience. Parallel Computing, 2004,30(7):817–840. [doi: 10.1016/j.parco.2004.04.001]

[28] Wang JM, Wu JH. Research on performance automation testing technology based on JMeter. In: Proc. of the 2019 Int'l Conf. on Robots & Intelligent System (ICRIS 2019). IEEE, 2019. 55–58. [doi: 10.1109/ICRIS.2019.00023]

[29] Povedano-Molina J, Lopez-Vega JM, Lopez-Soler JM, Corradi A, Foschini L. DARGOS: A highly adaptable and scalable monitoring architecture for multi-tenant Clouds. Future Generation Computer Systems, 2013,29(8):2041–2056. [doi: 10.1016/j.future.2013.04.022]

[30] Han DS, Xing JC, Yang QL, LI JL. Modeling and verification approach for temporal properties of self-adaptive software dynamic processes. Journal of Computer Applications, 2018(3):799–805 (in Chinese with English abstract).

[31] Xu LK, Li JL, Zhu Z, Chen HP, Wang L. Research on pattern mining of alarm sequence based on network and time constraints. Computer & Digital Engineering, 2019,47(9):2364–2368 (in Chinese with English abstract).

[32] Hamza Z, Abdallah T. Mapping fault tree into Bayesian network in safety analysis of process system. In: Proc. of the 4th Int'l Conf. on Electrical Engineering (ICEE 2015). IEEE, 2015. 1–5.

[33] Trihinas D, Pallis G, Dikaiakos M. Monitoring elastically adaptive multi-cloud services. IEEE Trans. on Cloud Computing, 2015, 6(3):1. [doi: 10.1109/TCC.2015.2511760]

[34] Ge JW, Zhang B, Fang YQ. Study on resource monitoring model in cloud computing environment. Computer Engineering, 2011, 37(11):31–33 (in Chinese with English abstract). [doi: 10.3969/j.issn.1000-3428.2011.11.011]

[35] Zhang X, Ying S, Zhang T. Collection and service processing framework of application running log. Computer Engineering and Applications, 2018,54(10):81–89,142 (in Chinese with English abstract).

[36] Wang R, Ying S, Sun CG, Wan HY, Zhang HL, Jia XY. Model construction and data management of running log in supporting SAAS software performance analysis. In: Proc. of the 29th Int'l Conf. on Software Engineering and Knowledge Engineering. 2017. 149–154. [doi: 10.18293/SEKE2017-128]

[37] Wang L. Software adaptation mechanism based on multi-agent theory and parallel search [Ph.D. Thesis]. Xi'an: University of Xidian, 2019 (in Chinese with English abstract).

[38] Zhang H, Li QS, Wang L, Cheng W. Self-adaptive software changes analysis method based on "Detection-Recognition" mechanism. In: Proc. of the 31st Int'l Conf. on Software Engineering & Knowledge Engineering (SEKE 2019). 2019. 287–290. [doi: 10.18293/SEKE2019-049]

附中文参考文献:

[7] 王子勇,王焘,张文博,陈宁江,左春.一种基于执行轨迹监测的微服务故障诊断方法.软件学报,2017,28(6):1435–1454. http://www.jos.org.cn/1000-9825/5223.htm [doi: 10.13328/j.cnki.jos.005223]

[14] 王焘,张文博,徐继伟,魏峻,钟华.云环境下基于统计监测的分布式软件系统故障检测技术研究.计算机学报,2017,40(2):397–413.

[16] 陈慧星,任艳,俞少行.运载火箭常规燃料调温系统模糊故障树研究.宇航学报,2017,38(1):104–108.

[19] 李正芳,庞海杰.基于改进遗传算法的网络故障数据实时检测系统研究.现代电子技术,2018,41(13):138–141,146. [doi: 10.16652/j.issn.1004-373x.2018.13. 031]

[20] 曲朝阳,刘晓庆,辛鹏.基于 Hadoop 的变电站设备故障状态识别与预测模型.软件导刊,2015(3):61–63.

[22] 杨艳燕.云计算中基于相似性图的故障自动诊断方法.计算机测量与控制,2014,22(12):3877–3880.

[24] 琚安康,郭渊博,朱泰铭,王通.网络安全事件关联分析技术与工具研究.计算机科学,2017,44(2):38–45.

[30] 韩德帅,邢建春,杨启亮,李决龙.自适应软件动态过程时间特性建模与验证方法.计算机应用,2018(3):799–805.

[31] 徐立坤,李建路,朱珠,陈海平,王林.基于网络和时间约束的告警序列模式挖掘研究.计算机与数字工程,2019,47(9):2364–2368.

[34] 葛君伟,张博,方义秋.云计算环境下的资源监测模型研究.计算机工程,2011,37(11):31–33.

[35] 张骁,应时,张韬.应用软件运行日志的收集与服务处理框架.计算机工程与应用,2018,54(10):81–89,142.
[37] 王璐.基于多智能体并行搜索的软件自适应机制[博士学位论文].西安:西安电子科技大学,2019.

王璐(1991－),女,博士,讲师,CCF 专业会员,主要研究领域为软件演化与自适应,智能化软件运维.

李青山(1973－),男,博士,博士生导师,CCF 高级会员,主要研究领域为面向智能体的软件工程,软件演化与自适应.

吕文琪(1996－),女,硕士,主要研究领域为软件自适应,自适应感知方法,智能化软件开发.

张河(1997－),女,硕士,主要研究领域为软件自适应,事件识别技术.

李昊(1996－),男,硕士,主要研究领域为自适应软件,软件演化,智能化软件开发.