基于粗糙集的容器云系统健康度评价建模

张可颖[1],龙士工[2,3],吕尚青[4],吕晓丹[3]
(1. 贵州大学 大数据与信息工程学院,贵州 贵阳 550025;
2. 贵州大学 贵州省公共大数据重点实验室,贵州 贵阳 550025;
3. 贵州大学 计算机科学与技术学院,贵州 贵阳 550025;
4. 北京邮电大学 信息与通信工程学院,北京 100000)

摘 要:现在容器云平台容器数目日益增加,相关监控数据爆炸式增长,而现有的运行在容器内的微服务监控软件监控指标不仅种类繁多,配置繁琐,并且往往只是直接给出监控数据,没有根据得到的监控指标对系统的健康度进行度量。针对该问题,提出了一种新的基于粗糙集的容器云系统健康度评价模型。通过建立的粗糙集云系统健康度评价模型,可以直观地反映整个集群的健康程度。首先通过信息熵对监控到的连续属性进行断点分割,离散化处理,然后利用粗糙集理论实现对监控数据进行知识约简、一致性检查和决策表建立,从而建立了基于粗糙集和信息熵的集群健康度指标模型。最后,通过 Kubernetes 容器云平台分别进行计算密集负载和网络密集负载仿真实验,实验结果表明,该模型能够反映集群的性能和对异常进行检测。

关键词:系统健康度;粗糙集;信息熵;云平台
中图分类号:TP319 **文献标识码**:A **文章编号**:1673-629X(2020)04-0063-06
doi:10.3969/j.issn.1673-629X.2020.04.012

Modeling of Health Evaluation of Container Cloud System Based on Rough Set

ZHANG Ke-ying[1],LONG Shi-gong[2,3],LYU Shang-qing[4],LYU Xiao-dan[3]
(1. School of Big Data and Information Engineering,Guizhou University,Guiyang 550025,China;
2. Guizhou Provincial Key Laboratory of Public Big Data,Guizhou University,Guiyang 550025,China;
3. School of Computer Science and Technology,Guizhou University,Guiyang 550025,China;
4. School of Information and Communication Engineering,Beijing University of Posts and Telecommunications,Beijing 100000,China)

Abstract:Now the number of containers of container cloud platform is increasing,and the related monitoring data is exploding. The existing monitoring indicators of micro-service monitoring software running in the container are not only diverse and complicated to configure,but also often just give monitoring data directly without measuring the health of the system based on the monitoring indicators obtained. For this,a new rough set based container cloud system health evaluation model is proposed,by which the health of the entire cluster can be intuitively reflected. Firstly,through the information entropy,the monitored continuous attributes are segmented and discretized. And then the knowledge reduction,consistency check and decision table establishment of monitoring data are completed by using rough set theory,so as to establish a cluster health indicator model based on rough set and information entropy. Finally,the computational intensive load and network intensive load simulation experiments are carried out through the Kubernetes container cloud platform,which show that this model can reflect the performance of the cluster and detect the anomaly.

Key words:system health;rough set;information entropy;cloud platform

收稿日期:2019-04-24 **修回日期**:2019-08-26 **网络出版时间**:2019-12-05
基金项目:贵州省科技重大专项计划(20183001)
作者简介:张可颖(1997-),女,CCF 会员(82186G),研究方向为云调度;通讯作者:龙士工,博士,教授,研究方向为应用密码学、网络空间安全与隐私保护。
网络出版地址:http://kns.cnki.net/kcms/detail/61.1450.TP.20191205.1134.056.html

0 引 言

近年,云计算领域蓬勃发展。特别是容器技术的出现颠覆了原有的虚拟化技术,其在自动配置和计划资源调度方面展现了良好性能。而且由于启动和终止开销较低,容器正迅速取代许多云部署中的虚拟机 VM[1],带来了技术架构的变革,从 SOA 到微服务,成为如今国内外大规模云服务产商的关注焦点,如国外 Google、AWS、Azure 和国内蚂蚁金服、腾讯云等。同时,容器云的出现也带来了新的挑战。基于微服务思想的容器云平台相比传统 IaaS 云平台,监控指标呈爆炸式增长,比如服务器节点指标、容器指标、应用性能指标、自定义业务指标等等。虽然出现了 Prometheus 这样一个统一的平台去处理监控问题,但是 Prometheus 只是一套开源的系统监控报警框架,并没有细粒度对系统健康度进行分析,繁琐复杂的监控指标还需要根据业务进行整合加工才能反映系统健康度。而且,目前对容器云平台的研究多集中于云资源调度和数据分析,对监控指标建模进行健康度分析讨论是比较欠缺的。

针对容器云平台微服务监控指标复杂,配置繁琐的问题,提出了一种基于粗糙集的容器云系统健康度建模方法。通过建立的健康度指标能够更直观地反映系统的健康度,从 POD 到 NODE 到整个集群,选取一系列指标从各个粒度和多个维度全面监控和整合数据,方便运维人员快速掌控集群实时情况,定位错误。同时对容器云健康度的建模分析也有助于对集群资源调度的优化。

1 相关工作

近年,学术界对容器云系统健康度建模研究比较欠缺,多集中于对特定应用背景容器平台监控架构设计和平台性能测试。鲜有学者专研系统健康度指标,提出一套通用的健康度评价体系。Seunghyun Seo 等人针对异构云平台提出了一套联合监控系统,主要包括四个模块即组件管理器、聚合管理器、注册表管理器和切片管理器[2]。Víctor Medel 等人通过 Petri 网性能模型分析了 Kubernetes 性能,并设计和规划了基于 Kubernetes 的弹性应用程序[3];Xie X 等人通过实验比较了基于 Kubernetes 的 Docker,Rkt 和裸机的性能,得出容器虚拟化技术的资源成本接近于“裸机”资源成本[4];Leila Abdollahi Vayghan 等人对 Kubernetes 的 HA 架构进行了一系列实验,表明微服务中断远高于预期[5]。

云平台健康度评价问题往往需要做出大量的不确定决策,而粗糙集在对不确定性信息处理方面得到了广泛的应用[6-9],是一种潜在的合适的解决方案。处理不确定性的数据工具除了粗糙集外,还有概率论、模糊集、模糊粗糙集等变种[10-11]。粗糙集在云平台方面应用的研究几乎没有。TeJen Su 等人将模糊集理论应用在云环境下的电力监控系统,通过实验证明了其可行性[12]。

在业界,有许多容器云指标监控解决方案,可用于分析 Kubernetes 性能,无论是开源的还是商业性质的。比如 Prometheus, Docker Stats API, cAdvisor, Sysdig, Heaper, Librato 等。Docker Stats API 直接输出 Json 很难整合;cAdvisor 在每个节点上提供一个运行守护进行,处理和导出每个节点运行容器详细资源使用信息,但是,它没有一个系统的编排,难以转化为更复杂的聚合和测试;Sysdig 是一种付费的解决方案;Heapster 只能监控主机级别和容器级别的数据,无法监控容器内服务级别数据;Librato 更适合结合 AWS 或 Heroku。而 Promethues 提供多维的数据模型,借助这种多维性所提供的灵活查询语言,理论上服务支持 Prometheus 端点或是提供 http 端口的转换器都可以手动拉取数据,既适用于面向服务器等硬件指标的监控,也适用于高动态的面向服务架构的监控[3]。故文中提出的建模方法使用 Python 借助 Promethues 中 Node exporter API 提取各项监控指标数据进行系统健康度建模。

2 资源监控平台架构

2.1 容器云平台架构和监控指标

容器云平台采用 Kubernetes 和 Docker 实现,架构为主从架构,监控维度从整个集群到 Node 再到 Pod,一个 Node 有多个 Pod,一个 Pod 可能有多个容器,如图 1 所示。Master 节点上安装有重要系统组件:API Server、Scheduler、Controller 和 Etcd,以 Pod 形式存在。根据业务不同,主要可分为计算密集业务、网络密集业务和 IO 密集业务。Node 监控维度包括 Node 的 CPU、内存、网络和 IO 利用率等。Pod 监控维度包括 Pod 的 CPU、内存、网络和 IO 相关指标。详细参数参考下面章节内容。

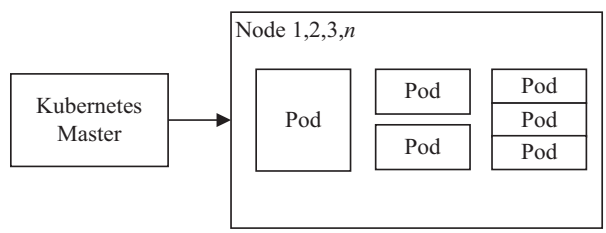

图 1 云平台架构

2.2 资源监控数据获取流程

Node-Exporter 对 Prometheus 暴露出系统可以被“抓取”的 Metrics 信息。Prometheus 调用 api 端口信息完成查询,Grafana 调用信息完成数据可视化。文中

使用 Python 抓取从 exporter api 暴露的 Metrics 信息,然后进行基于信息熵连续属性离散化处理,最后建立基于粗糙集的决策表,推导出决策规则。

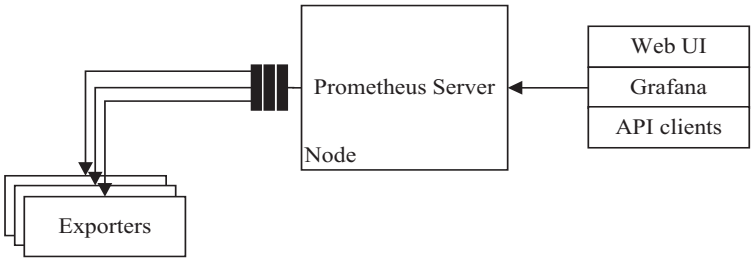

图 2　资源监控数据获取流程

3　基于粗糙集的容器云系统健康度建模

3.1　基于系统粗糙集建立决策表

定义 1(粗糙集分类):粗糙集健康指标决策系统可以定义为四元组[13] $S = < U,R,V,F >$。U 表示对象的有限集,其中 $U = \{x_1,x_2,\cdots,x_n\}$ 为对象的非空有限集合,即论域; $R = \{a_1,a_2,\cdots,a_n\}$ 表示属性的有限非空集, $R = C \cup D$, $C = \{c_1,c_2,\cdots,c_n\}$ 是条件属性子集, $D = \{d\}$ 是决策属性子集; V 表示属性值域, $V = \bigcup_{p \in A} V_p$; $f: U \times A \rightarrow V$ 为信息函数,表示样本、属性和属性值之间的映射关系。在决策表中形式化表示如下: $f(x_i, c_i) = u_{j,i}, f(x_i,d) = v_i$。

(1)云平台中 Node 的监控数据(条件属性) NodeC = {cpu,memory,network,io…} 和 PodC = {cpu{…},memory{…},network{…},io{…}} 均为连续属性,Pod 部分经典监控参数见表 1。决策属性是系统健康度 D = {healthy,normal,unhealthy},为离散属性。

表 1　Pod 监控参数

监控类型	属性名称	序号
CPU	container_cpu_usage_seconds_total	1
	container_cpu_load_average_10s	2
内存	container_memory_usage_bytes	3
网络	http_request_duration_microseconds	4
	http_response_size_bytes	5
	http_request_size_bytes	6
	container_network_transmit_errors_total	7
	container_network_receive_errors_total	8
	container_network_receive_packets_dropped_total	9
	container_network_transmit_packets_dropped_total	10
	container_network_receive_bytes_total	11
	container_network_transmit_bytes_total	12
IO	container_fs_read_seconds_total	13
	container_fs_write_seconds_total	14
	container_fs_usage_bytes	15
	container_fs_io_time_seconds_total	…
…	…	…

决策表的一般形式见表 2。

表 2　决策表的一般形式

对象	条件属性			决策属性
U	c_1	…	c_m	d
x_1	$u_{1,1}$	…	$u_{m,1}$	v_1
x_2	$u_{1,2}$	…	$u_{m,2}$	v_2
…	…	…	…	…
x_n	$u_{1,n}$	…	$u_{m,3}$	v_n

(2) $\underline{RX}$, $\overline{RX}$ 分别表示上近似和下近似粗糙集,每一个子集 $X \subseteq U$,并有等价关系 $A \in IND(P)$, $\underline{RX} = \{x \in U \mid [x]_R \subseteq X\}$, $\overline{RX} = \{x \in U \mid [x]_R \cap X \neq \varnothing\}$, $\underline{RX} \neq \overline{RX}$。对应到系统中, $\underline{RX}$ 是系统中确定健康或不健康的最小定义集, $\overline{RX}$ 是系统中所有可能健康或不健康的最大定义集。

(3)决策表相容度定义为:

$$d_p(Q) = \frac{\sum_{k=1}^{s} |P(X_k)|}{|X|}$$

其中，$|X|$ 表示集合的个数，$P(X_k)$ 表示子集 X_k 在等价关系 P 下的下近似集。一般有 $0 \leqslant d_p(Q) \leqslant 1$，当 $d_p(Q)=1$ 时，表示所有的实例在相同条件属性下都有相同的决策属性值，此时称决策表是相容的。

(4)属性约简[14]。

定义 2：设 U 为一个论域，P 为定义在 U 上的一个等价关系簇，P 中所有绝对必要关系组成的集合称为关系簇 P 的绝对核，记作 CORE(P)。

定义 3：设 U 为一个论域，P 和 U 为定义在 U 上的两个等价关系簇，若 $\mathrm{POS}_P(Q)=\mathrm{POS}_{(P\setminus\{r\})}(Q)$，则称 r 为 P 中相对于 Q 可省略的(不必要的)，简称 P 中 Q 可省略的；否则称 r 为 P 相对于 Q 不可省略的(必要的)。

属性约简算法：

对于决策表中的每一个条件属性 c_i，进行如下流程，直至属性集合不再发生变化为止。

{

如果删除该属性 c_i 使 $\mathrm{POS}_{(P\setminus\{c_i\})}(Q)=\mathrm{POS}_P(Q)$，则说明属性 c_i 是相对于决策属性 d 不必要的，从决策表中删除属性 c_i 所在列并将重复的行进行合并；

否则，说明属性 c_i 是相对于决策属性 d 必要的，不能删除

}

对决策表条件属性进行约简，去掉不重要的条件属性，从而提取更准确和最少的判断系统监控度的决策规则。

(5)决策表一致性检查[15]。

定义 4：设 U 为一个论域，P 和 Q 为定义在 U 上的两个等价关系簇，如果 $\mathrm{POS}_p(Q)=U$，则称论域 U 是 P 上相当于 Q 一致的。

3.2 基于信息熵离散化连续属性

定义 5(信息熵)：设离散随机变量 $X=\{x_1,x_2,\cdots,x_n\}$，$X\subseteq U$，$|X|$ 表示实例个数。其中决策属性 $j(j=1,2,\cdots,r(d))$ 实例个数为 K_j，则 X 的信息熵定义为：

$$H(X)=-\sum_{j=1}^{r(d)} p_j \log_2 p_j, p_j=\frac{k_j}{|X|}$$

信息熵是信息论中用于度量信息量的一个概念。一个系统越是有序，信息熵就越低；反之，一个系统越是混乱，信息熵就越高。对云平台，如果基于监控指标计算的信息熵增加，表示集群有可能处于异常状态，需要动态弹性调整。因为监控数据是连续数据，建立粗糙集决策表需要将连续数据进行离散化。

具体的离散化算法步骤设计如下：

设决策表 $S=<U,R,V,F>$，$R=C\cup D$，条件属性集合 $C=\{c_1,c_2,\cdots,c_n\}$，P 为已经选取的断点集合，Q 是实例被断点集合 P 所划分成的等价类的集合，B 为候选的断点集合，H 为决策表信息熵，对连续的条件属性 $k\in C$，将属性值进行排序后：

$$l_k=v_0^k<v_1^k<\cdots<v_n^k$$

可选断点：

$$c_i^a=(v_{i-1}^a+v_i^a)/2(i=1,2,\cdots,n)$$

对于断点 C_j^a，针对集合 U 的信息熵计算方法参考文献[15]中方法。

离散化算法如下：

步骤一：$P\neq Q, Q=\{U\}, H=H(U)$，设置分类断点数 i；

步骤二：分别对各条件属性的属性值进行排序，取相邻两个属性均值加入到候选断点集合 B 中；

步骤三：计算每一个断点信息熵 $H(c,Q)$，$c\in B$，取 $c_{\min}=\min\{H(c,Q)\}$；

步骤四：若 $c_{\min}$ 是唯一断点，选择使得 $H(c,Q)$ 最小的断点 $c_{\min}$ 加入 P 中，$H=H(c,Q)$；$B=B-\{c\}$ 转步骤六；若 $c_{\min}$ 不是唯一断点，则转步骤五；

步骤五：计算信息熵相同的断点的条件属性的分类正确率，选择分类正确率大的条件属性的断点作为结果断点；

步骤六：对所有 $X\in Q$，若 $c_{\min}$ 将等价类 X 划分成 X_1 和 X_2，从 Q 中去掉 X，并把等价类 X_1 和 X_2 加入到 Q；

步骤七：计算离散后决策表的相容度，如果相容度在可接受范围且选择的 $c_{\min}$ 的数量等于 i 则结束，输出断点集合 P；若相容度不在接受范围则转步骤三计算下一个断点，若 i 个点不能计算出相容度接受范围结果，则 $i=i+1$。

根据离散后的决策表和粗糙集算法分别提取出决策规则。

4 模型验证

为了验证所建容器云系统健康度模型的有效性，实验选用 DELLR710 服务器，使用 Openstack Q 版本开启的 3 台虚拟机，系统采用 Centos7.6，软件版本为 Kubernetes 1.13.1，1 台 Master 节点(4G 内存 4 核 CPU)2 台 Slave 节点(每台 8G 内存 4 核 CPU)。实验系统健康度监控粒度为节点级别。文中在 Kubernetes 平台 Slave1 采集四组数据，第一组数据为采集筛选后的 60 个基本训练数据；第二组数据在第一组基础上再采集 10 次，在第 10 次插入 CPU 异常；第三组数据在第二组数据基础上再采集 10 次，在第 10 次插入内存异常；第四组数据在第三组数据基础上再采集 10 次，在第 10 次插入网络异常。本次实验只选取了三个条件属性代表系统特征，分别为 cpu，mem，net，生产环境应当选取更多监控指标和更细粒度的监控级别，可参考 3.1。初始决策属性判断依据服务负载占用量给

出判断系统健康与否的标准。除此之外,还可以选择新创建服务的响应时间进行判断。

为了模拟云平台负载,在容器中运行 Tensorflow 卷积神经网络 Demo 模拟 CPU 和内存负载,在容器里面使用 Tomcat 搭建网站,用 Apache Bench 模拟网站并发访问量。

第一组训练数据的 cpu,memory,network 选取断点及断点信息熵,如表 3 所示。

表 3　第一组数据基本指标选取断点及断点信息熵

条件属性	cpu	memory	network
断点值	28.5(0.67)	29.5(0.67)	30.5(0.67)
(信息熵)	68(0)	80(0.63)	77.5(0.65)

根据选择的信息熵对数据进行离散化,检查离散后的决策表是一致的,属性不可约简,可以导出下列决策规则:

r_1:(low,low,low) → (healthy)

r_2:(middle,middle,middle) → (normal)

r_3:(high,middle,high) → (unhealthy)

r_4:(high,high,middle) → (unhealthy)

r_5:(high,middle,high) → (unhealthy)

r_6:(high,high,high) → (unhealthy)

经计算,这些规则的确定性均为 1。

(1)cpu 异常注入。

第二组数据基本指标选取断点及断点信息熵见表 4。

表 4　第二组数据基本指标选取断点及断点信息熵

条件属性	cpu	memory	network
断点值	28.5(0.63)	29.5(0.63)	30.5(0.63)
(信息熵)	68(0)	80(0.62)	77.5(0.63)

相比第一组基础数据,断点未改变,提取的决策规则不变,能成功判断出 cpu 异常。

第二组数据离散后判断出的决策表见表 5。

表 5　第二组数据离散后判断出的决策表

序号	cpu	memory	network	health
1	low	low	low	healthy
2	low	low	low	healthy
3	low	low	low	healthy
4	low	low	low	healthy
5	low	low	low	healthy
6	middle	middle	middle	normal
7	middle	middle	middle	normal
8	middle	middle	middle	normal
9	middle	middle	middle	normal
10	high	middle	middle	unhealthy

(2)Memory 异常注入。

第三组数据基础指标选取断点及断点信息熵如表 6 所示。

表 6　第三组数据基础指标选取断点及断点信息熵

条件属性	cpu	memory	network
断点值	28.5(0.61)	29.5(0.61)	30.5(0.61)
(信息熵)	68.5(0)	80(0.64)	77.5(0.64)

相比第二组数据,断点未改变,提取的决策规则不变,能成功判断出 mem 异常。

第三组数据离散后判断出的决策表如表 7 所示。

表 7　第三组数据离散后判断出的决策表

序号	cpu	memory	network	health
1	low	low	low	healthy
2	low	low	low	healthy
3	low	low	low	healthy
4	low	low	low	healthy
5	low	low	low	healthy
6	middle	middle	middle	normal
7	middle	middle	middle	normal
8	middle	middle	middle	normal
9	middle	middle	middle	normal
10	middle	high	middle	unhealthy

(3)Network 异常注入。

第四组数据基础指标选取断点及断点信息熵如表 8 所示。

表 8　第四组数据基础指标选取断点及断点信息熵

条件属性	cpu	memory	network
断点值	28.5(0.59)	29.5(0.59)	30.5(0.59)
(信息熵)	74(0)	80(0.61)	77.5(0.65)

相比第三组数据,断点未改变,提取的决策规则不变,能成功判断出 net 异常。

第四组数据离散后的判断出的决策表如表 9 所示。

表 9　第四组数据离散后判断出的决策表

序号	cpu	memory	network	health
1	low	low	low	healthy
2	low	low	low	healthy
3	low	low	low	healthy
4	low	low	low	healthy
5	low	low	low	healthy
6	middle	middle	middle	normal
7	middle	middle	middle	normal
8	middle	middle	middle	normal
9	middle	middle	middle	normal
10	middle	middle	high	unhealthy

Slave1 实验过程资源负载变化如图 3 所示。

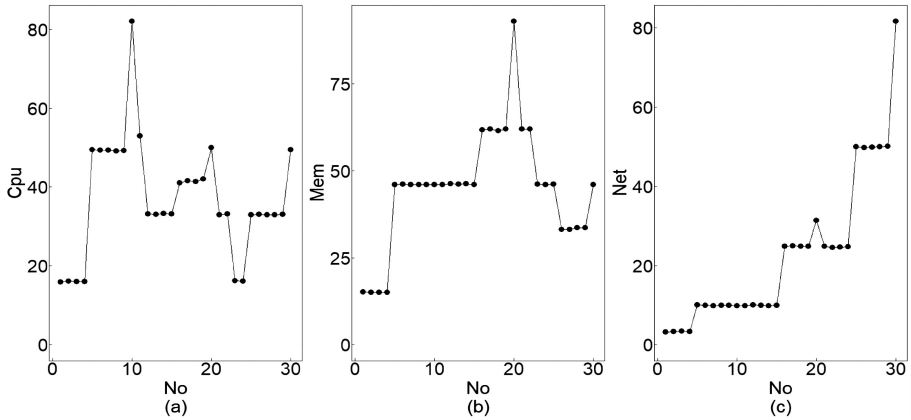

图 3 Slave1 实验过程资源负载变化

5 结束语

文中容器云系统健康度模型能够反映集群负载情况并检测异常,从而实时、动态地对整个云平台资源进行直观反映。该容器云系统健康度模型结合了粗糙集和信息熵的理论,基于 Kubernetes 容器云平台的实验验证了模型的有效性以及分析定位的准确性。对云平台系统健康度的分析建模有助于进一步对系统亚健康状态进行调度优化,从而在提高系统利用率的同时保证服务质量。

参考文献:

[1] PAREKH N, KURUNJI S, BECK A. Monitoring resources of machine learning engine in microservices architecture[C]// 2018 IEEE 9th annual information technology, electronics and mobile communication conference (IEMCON). Vancouver, BC: IEEE, 2018: 486-492.

[2] SEO S, KIM M, CUI Y, et al. SFA-based cloud federation monitoring system for integrating physical resources[C]// 2015 international conference on big data and smart computing (BIGCOMP). Jeju: IEEE, 2015: 55-58.

[3] MEDEL V, TOLOSANA-CALASANZ R, BAÑARES J, et al. Characterising resource management performance in Kubernetes[J]. Computers & Electrical Engineering, 2018, 68: 286-297.

[4] XIE X, WANG P, WANG Q. The performance analysis of Docker and rkt based on Kubernetes[C]//2017 13th international conference on natural computation, fuzzy systems and knowledge discovery. Guilin, China: IEEE, 2017: 2137-2141.

[5] 徐浩天,梁俸齐,罗 丽. 微服务架构与容器技术分析及应用[J]. 信息系统工程,2019(4):97-98.

[6] 于 洪,王国胤,姚一豫. 决策粗糙集理论研究现状与展望[J]. 计算机学报,2015,38(8):1628-1639.

[7] 王国胤,于 洪,杨大春. 基于条件信息熵的决策表约简[J]. 计算机学报,2002,25(7):759-766.

[8] 张 维,苗夺谦,高 灿,等. 基于粗糙集成学习的半监督属性约简[J]. 小型微型计算机系统,2016,37(12):2727-2732.

[9] LIA J, CHEN X, WANG P, et al. Local view based cost-sensitive attribute reduction[J]. Filomat, 2018, 32(5): 1817-1822.

[10] ZHANG C, LI D, ZHAI Y, et al. Multigranulation rough set model in hesitant fuzzy information systems and its application in person-job fit[J]. International Journal of Machine Learning and Cybernetics, 2017, 10: 717-729.

[11] YANG B, HU B. On some types of fuzzy covering-based rough sets[J]. Fuzzy Sets and Systems, 2017, 312: 36-65.

[12] SU T, WANG S, VU H, et al. An application of fuzzy theory to the power monitoring system in cloud environments[C]// 2016 international symposium on computer, consumer and control (IS3C). Xi'an: IEEE, 2016: 350-354.

[13] 张鹏飞,李本威,秦 明,等. 一种基于熵的连续属性离散方法[J]. 燃气涡轮试验与研究,2014,27(6):49-52.

[14] 周 炜,周创明,史朝辉,等. 粗糙集理论及应用[M]. 北京:清华大学出版社,2015.

[15] 谢 宏,程浩忠,牛东晓. 基于信息熵的粗糙集连续属性离散化算法[J]. 计算机学报,2005,28(9):1570-1574.