智能环境下基于边缘设备规则推理的数据预部署研究

汪成亮[1],赵 凯[1],刘嘉敏[2]
(1. 重庆大学计算机学院,重庆 400044; 2. 重庆大学光电工程学院,重庆 400044)

摘 要: 在现有的规则推理机制下,大量的传感器数据导致的过大规则匹配期间的实时特征计算量降低了推理实时性,同时边缘设备受限的内存资源难以应对如此庞大的数据量. 为此,本文设计了数据预部署方案(Data Pre-Deployment Scheme,DPDS). 利用规则解析与预处理模块解析规则集得到的规则网络和轻量级特征表(Light-weight Characteristic Table,LCT),该方案无需进行实时特征计算,使推理效率和实时性得到显著提高,并大大降低了规则匹配期间的内存占用量. 实验表明,即使在规则、数据规模很大的情况下,DPDS仍然具有较高的时间效率和空间效率.

关键词: 智能环境; 边缘设备; 规则推理; 规则匹配; 无线传感器网络
中图分类号: TP181 **文献标识码:** A **文章编号:** 0372-2112(2022)10-2347-14
电子学报 URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20200891

Study on Data Pre-Deployment Based on Inference and Computing of Edge Devices in Smart Environment

WANG Cheng-liang[1], ZHAO Kai[1], LIU Jia-min[2]
(1. *College of Computer Science and Technology, Chongqing University, Chongqing* 400044, *China*;
2. *College of Optoelectronic Engineering, Chongqing University, Chongqing* 400044, *China*)

Abstract: Under the existing rule inference mechanism, the amount of real-time feature calculation during rule matching caused by a large amount of sensor data reduces the inference real-time performance. At the same time, the limited memory resources of edge devices cannot cope with such a huge amount of data. For this reason, this thesis designs the Data Pre-Deployment Scheme(DPDS). By utilizing the rule network and Light-weight Characteristic Table(LCT) obtained by the rule analysis and preprocessing module, this scheme enables the rule network to directly reference the characteristic values in the LCT during inference without real-time feature calculations, which significantly improves efficiency and real-time and greatly reduces the memory usage of the inference process. The experimental results show that even in the case of a large amount of data and rules, DPDS still has high time efficiency and space efficiency.

Key words: smart environment; edge device; rule inference; rule matching; wireless sensor network

1 引言

因无线传感网络(Wireless Sensor Network,WSN)所展现出的巨大潜力,已出现大量关于智能环境系统的研究[1]. 智能环境可视为一个可交互式空间,其中部署了大量边缘设备,其相互关系如图1所示. 智能环境的目的是根据边缘设备中传感器采集的环境信息,来进行活动识别[2]、睡眠监测[3]等活动. 在智能环境所部署的大量边缘设备中,数据采集模块和规则推理模块均部署于各个设备中. 而规则推理模块在进行工作前,需要接收边缘设备上所搭载传感器所采集的环境信息为推理提供信息依据. 而在规则推理模块中,其存储于规则库中的规则通常由对应领域的专家知识转化而来,其形式可表示为“IF-THEN”,IF修饰条件前件,THEN修饰响应,分为规则前后件. 工作内存是规则推理模块中的重要组成,是储存了各种数据元素的全局数据库. 相关数据元素包括智能环境中所嵌入的传感器以设置周期所获取到的原始数据,以及在推理过程中生成的临时数据. 规则条件可访问所有的存放于工作内存中

收稿日期:2020-08-15;修回日期:2021-10-08;责任编辑:孙瑶
基金项目:国家自然科学基金(No.61672115);重庆市技术创新与应用发展专项重大主题专项(No.cstc2019jscx-zdztzxX0037)

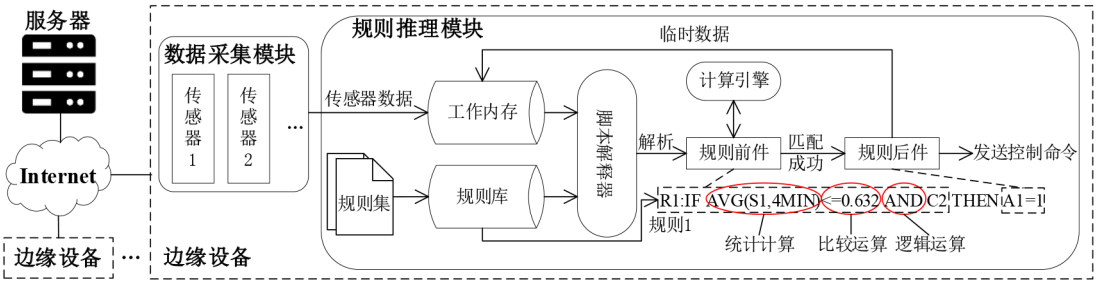


图1 智能环境系统与边缘设备

的数据. 而在运行过程中,系统会结合目前存放于工作内存中的数据,使其与规则库中所设定的规则对比. 在对比过程中,计算引擎将完成规则前件中的各种推理计算,其中包括比较、逻辑等类型的特征运算. 如果推理计算的结果为真,将激活规则. 随后,其对应的响应也将被触发,同时会生成一些临时数据,这些数据可能进一步应用于下一阶段的规则推理. 但是,已有的规则推理模块存在较低的匹配效率的明显缺点,整个系统在匹配任务中所用时间占比达到90%. 这个问题将导致实时性需求难以得到满足. 综合而言,有以下2个方面原因导致匹配效率下降[4].

(1)一些规则被重复匹配. 智能环境系统中,规则集所包含的不同规则之间常出现重叠或交叉的内容于规则前件或更细粒度的计算单元之中,这种现象存在于大部分应用当中. 另外,适应于发展需求,规则集的规模将随之快速扩大,规则和数据逐一匹配的策略,将使重叠规则部分的重复匹配的问题更加恶化. 匹配工作的耗时将因规则和数据规模的扩大,而表现出指数增长的趋势[4].

(2)过大的推理计算量. 由于无线传感网络的不断发展,智能环境系统所产生的数据量不断增加. 规则匹配过程中,规则和数据的大大增加,将导致原始数据的特征计算量难以接受. 当前推理系统的规则匹配需要对原始数据采取大量实时特征计算操作,导致匹配效率无法满足需求[4].

关于规则的重复匹配这一问题,Forgy所提出的RETE算法[5],取得了差强人意的解决效果. 作为一种以增量匹配为基本思路的推理算法,RETE算法利用所建立的规则网络,实现了规则间的模式共享的策略,从而达成了规则匹配的效率提升的效果. 当前,各种基于RETE算法的改进推理系统被实现(如CLIP[6],JESS[7],DROOLS[8],BizTalk[9]),并产生了实用价值. 尽管如此,笨重和复杂仍是此类系统难以避免的通病. 此外,RETE算法因没有针对推理计算量过大这一问题设计出可接受的解决方案,而无法在智能环境下的推理系统中得到理想的应用. 此外,原始数据需要在推理完成前一直保留,而边缘设备有限的内存资源,难以负担大量内存需求. 这一问题成为达到推理实时目的的关键瓶颈[4].

针对上述问题,本文设计了适用于无线传感器网络的基于边缘设备规则推理的数据预部署方案(Data Pre-Deployment Scheme,DPDS)[4]. 该方案对相关规则分析后,进行了对应的预运算. 由此所建立的规则网络,能够消除同样的规则匹配操作. 而规则前件中的被提取特征后计算,得到对应的统计单元. 根据统计单元中所构建的轻量级特征表(Light-Weight Characteristic Table,LCT),将特征值预先进行计算和存储操作. 规则网络可对轻量级特征表中的已经预先计算好的特征值进行引用,从而消除了规则匹配过程中对大量原始数据进行的实时特征计算的操作. 基于系统对轻量级特征表的内存使用量在一定程度上是静态可预测的特点,本文设计了一种LCT预部署策略,用以解决系统内存受限这一问题[4]. 本文主要做了如下3个方面的工作.

(1)对规则解析与预处理模块进行了设计[4],对规则进行解析和预运算操作,从而建立规则网络. 同时对规则前件中的特征进行提取,计算后作为统计单元,实现了推理过程中推理计算和规则匹配的分离[4].

(2)设计了有在线数据结构特性的一种轻量级特征表[4],它以统计单元作为输入,对推理过程中的特征进行预计算并提供对特征值进行存储的功能,使得推理时规则网络可以利用轻量级特征表中已经过预处理的特征值完成推理. 提出了LCT预部署策略,该策略在系统内存受限时,通过为轻量级特征表设定不同的优先等级并对一些轻量级特征表采取离线操作,对系统内存受限这一问题,提出了理想处理方案[4].

(3)进行仿真实验,模拟不同规模规则数目、不同数据流流速和不同最大滑动窗口长度下本文提出的方案与现有推理系统、采用模糊处理和B+树结构的RETE算法的推理系统以及采用RETE算法的推理系统的效果. 结果表明,本文提出方案在时间性能与空间性能上均有显著提升.

2 相关研究

在基于规则的系统领域,研究人员对规则处理方

案展开了广泛的研究[10,11]. 在推理系统领域,研究则主要包括两个方面:RETE算法和复杂事件处理机制(Complex Event Processing,CEP). RETE算法是模式匹配中的一种十分经典的算法,由Forgy于1989年首次提出[5]. RETE算法首先用于产生式系统[12]. 近年来,许多研究人员更加关注其在某些特定应用中的算法实现与改进. Chen等人[13]提出的模糊处理和B+树结构克服了RETE算法在大规模规则方面的局限性. 模糊处理使用模糊集将精确数据转换为模糊概念,便于用户自定义规则,而B+树结构加快了对非空节点的遍历. 根据工业环境的特点,Guo等人[14]提出了基于预匹配的分支过滤RETE算法. 为了在内存受限的平台上部署推理系统,文献[15]提出了一种自适应算法,该算法通过选择性地合并RETE网络的Alpha节点,更好地平衡了系统性能和内存使用量. 文献[16]通过在初始化RETE网络时令Alpha节点得到一个指向原始数据的指针,使得Alpha节点能够仅存储条件比较的结果而不是冗余的数据,从而使RETE网络的内存使用量得到进一步减小. Milhan等人[17]扩展了RETE算法以增强与复合上下文感知服务体系结构的系统,通过只搜索可匹配规则的子集,提高了规则匹配速度. 为了从RFID实时且不可靠的原始数据中检测出有价值的事件,Peng等人[18]建立了Alpha监测网络监测原始数据的所有属性,然后利用Beta监测网络将事件与业务规则组装成RFID复杂事件,最终得到相关的RFID复杂事件. 为解决RETE算法在性能和灵活性方面的问题,文献[19]使用了规则分解、Alpha节点索引和Beta节点索引这3种方法来改进RETE算法. 通过采用节点共享、类型预处理和基于索引的搜索优化机制,文献[20]提出了RETE算法的改进版本IRETE,该算法在多实体和多规则的环境下进行了测试并被证明为更高效的匹配算法. Xue[21]提出用SVM从工业数据中过滤出有意义的数据,使RETE算法不去匹配这些“垃圾”数据. Hubrechts等人[22]提出的Mete算法提供了一种基于Rete算法对分布式推理系统进行抽象和通用扩展的便捷方法,Mete算法无需对程序代码做任何修改. Araujo等人[23]针对RETE算法提出了一种知识库搜索算法,该算法利用现代计算机固有的并行结构,提高了推理引擎的性能. LIU等人[24]通过利用Bitmap事件对规则进行编码,提升了规则匹配效率. 传统的RETE算法在规则由很多重叠条件组成时才能达到较高性能,Kim等人[25]通过在条件的不同层级上生成多级索引树,在规则条件不重叠的情况下使算法也能有较好的性能表现.

随着无线传感器网络的发展,推理系统往往需要处理具有逻辑、时间限制和其他操作的复杂关联规则. Karen等人[26]提出了一种用于支持间隔时间语义的RETE算法的扩展,并指出了该扩展产生的问题. 另外,他们[27]还提出了一种用于检测相对时间约束的扩展,以及一种RETE算法中的对过期事件的垃圾回收机制. 为支持系统对时间敏感事件的解释,Bestel等人[28]提出了通过时间戳数据和报告数据之间的时间约束的概念的RETE算法的扩展. 此外,Wu等人[29]还介绍了一个可以对由RFID读取的实时数据流编码的事件进行复杂事件查询的系统.

上述所有的研究工作,通常都是通过集成事件处理与RETE算法来实现推理系统,然而,它们大多为特定应用设计,因此太过复杂和冗余. 更重要的是,当推理时有较大数据量时,RETE算法无法给出令人满意的解决方案,因此无法直接应用于基于无线传感器网络的智能环境系统中. 本文结合智能环境中的数据和规则特点,提出了一种适用于无线传感器网络的基于边缘设备规则推理的数据预部署方案.

3　数据预部署方案

数据预部署方案(Data Pre-Deployment Scheme,DPDS)的架构展示于图2中,依照规则集中规则所包含的推理周期,能够使规则集划分出推理周期各异的规则子集. 结合规则解析以及预处理模块,在所得到的规则子集内,能够生成规则网络和对应的轻量级特征表. 轻量级特征表中的特征值可被规则网络所引用,以此达成规则网络的过滤和发送,触发特征值匹配的组合条件的响应可完成相关推理工作[4].

智能环境的智能性由边缘设备所采集到的传感器序列数据流并由规则推理模块推理计算而达成. 关于规则推理,具有对时序区间内原始数据进行特征提取、阈值比较以及逻辑判别等计算内容,比较和逻辑运算比特征提取的计算量相对而言更小. 为解决减小实时特征提取所需的大量计算,所带来的严重时延问题,本文提出了一种能够完成规则解析与预处理的模块. 该模块对规则集的进行了计算分析,能够构建出相应的规则网络,以及获取到规则网络中各原子条件的统计单元. 由获取到的统计单元,本文采用一种类型为在线数据结构的轻量级特征表(Light-weight Characteristic Table,LCT)进行计算并且储存原子条件引用的特征值. 由于基于DPDS中系统对LCT的内存使用量能够进行一定程度上的静态预测,本文为解决系统内存受限的问题,提出了一种LCT预部署方案. 该方案采用的方式是为轻量级特征表给出优先级,并对剩余内存进行预估,在其不足时对优先级较低的轻量级特征表采取离线操作[4].

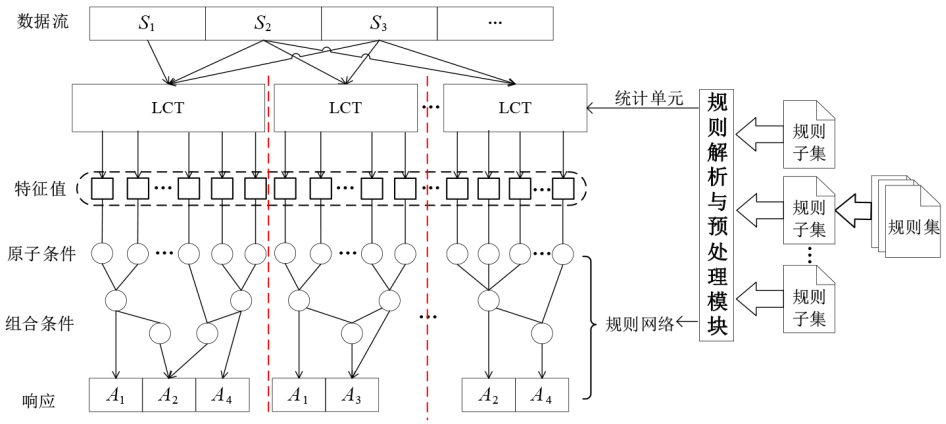


图2 数据预部署策略架构图

3.1 时间序列数据流

3.1.1 时间序列数据流

输入数据在智能环境中可整理为时序增序的三元组(streamID,timestamp,value). 同一数据流的streamID相同. 举例而言,streamID可在睡眠监测应用中表示震动[4].

定义1 (数据流) 一个数据流可表达为依照时序递增排列的元组序列 $S=\langle e_0,t_0\rangle,\langle e_1,t_1\rangle,\cdots,\langle e_n,t_n\rangle$,其中 e_i 是对应于时刻 t_i 的一个序列元素[4].

数据流中新的元组依据所设时间周期性地添加,这一时间段为数据流的增长周期,而数据流的流速与增长周期的乘积为一个常数. 在特殊情况下,如果在增长周期期间无新元组添加,则采用插值法填充. 如果增长周期包含一个以上的新元组,则计算多个元组,将其汇总结果进行添加. 在本文中,$S[t]$ 为在时刻 t 数据流 S 的值,S_k 表示streamID值取 k 时的数据流,$S_k[i:j]$ 则为从时刻 i 到 j 数据流 S_k 的一个子序列[4].

3.1.2 规则推理

智能环境系统当中所制定的规则是一种人类对求解问题的行为特征表示方法. 规则具有的2种属性为(RID,period),其中RID为规则的ID,period则是规则的推理周期. 例如 $C_1\wedge C_2\vee\left(C_3\wedge C_4\right)\rightarrow A_1$ 可作为一种规则结构的表示方式,其中 C_i 和 A_i 分别为原子条件以及对应响应. 而由范式存在定理可知,在所有命题公式中均有与之等值的析取范式(Disjunctive Normal Form,DNF)存在,举例而言,命题公式 $\left(x_1\vee y_1\right)\wedge\left(x_2\vee y_2\right)\wedge\cdots\wedge\left(x_n\vee y_n\right)$ 与 $\left(x_1\wedge x_2\wedge\cdots\wedge x_{n-1}\wedge x_n\right)\vee\left(x_1\wedge x_2\wedge\cdots\wedge x_{n-1}\wedge y_n\right)\vee\left(y_1\wedge y_2\wedge\cdots\wedge y_{n-1}\wedge y_n\right)$ 等值. 以该范式存在定理为基础,规则 $C_1\wedge C_2\vee\left(C_3\wedge C_4\right)\rightarrow A_1$ 可对应以下两条原子规则进行转化[4]:

$$C_1\wedge C_2\vee\left(C_3\wedge C_4\right)\rightarrow A_1$$
$$\Downarrow$$
$$C_1\wedge C_2\wedge C_3\rightarrow A_1$$
$$C_1\wedge C_2\wedge C_4\rightarrow A_1$$

定义2 (原子规则) 原子规则是其条件仅由逻辑与参与连接的规则[4].

所有原子规则可以由一种树形结构表示,其原子条件为叶结点,而响应表示为根结点. 规则网络则是由若干推理周期一致的规则所组成的树所具有的原子条件所构成的组合条件所构造. 所有的规则网络由包含一致推理周期的规则所构成的规则子集所构建. 图3所示为由规则 $C_1\wedge C_2\vee\left(C_3\wedge C_4\right)\rightarrow A_1$ 构建的规则网络. 图3(a)将规则 $C_1\wedge C_2\vee\left(C_3\wedge C_4\right)\rightarrow A_1$ 转化成为两个原子规则,并由此分解为两棵规则树. 图3(b)进一步通过组合原子条件形成的组合条件,构建出对应的规则网络. 组成与组合原子条件的谓词和运算符的常见示例如下[4].

(1)时间谓词:时间界限(例如滑动窗口)等.

(2)数据谓词:比较运算符,如<,>,≤,≥,=等;统计运算符,如MAX,MIN,AVG,SD(Standard Deviation),VAR(VARiance)等.

(3)逻辑运算符:逻辑与(∧),逻辑或(∨),逻辑非(~).

定义3 (原子条件) 无法被逻辑运算符分割的规则中的最小条件,作为原子条件[4].

定义4 (组合条件) 组合条件定义为由一个及以上的原子条件,或者由零个及以上的组合条件经逻辑运算符连接而构成[4].

现有推理系统中的推理过程,是向规则网络输入数据流,传播通过网络过滤后的原始数据,再触发与原始条件匹配的组合条件响应的过程. 所有在规则网络中具有与行为特征相关的计算操作的原子条件的行为特征,通常能够被各种统计特征和少许非统计特征的

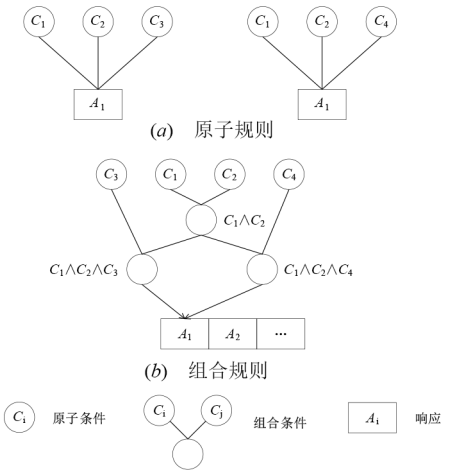


图3　将规则转化为规则网络

其他特征所归纳. 在对其他特征进行规则匹配时,本文不做任何特殊处理,按现有推理系统采取的文本扫描和对原始数据进行特征计算来进行. 后文将主要考虑行为特征为统计特征时的情况. 设时间长度 T 和当前时刻 t,则 $\mathrm{stat}(S_k, T)$ 表示在数据流 S_k 的子序列 $S_k[t-T: t]$ 上进行 stat 的运算符特征计算,则子序列 $S_k[t-T: t]$ 可表示为由时间间隔为 T 的滑动窗口所限定于在数据流 S_k 上所包含区间的 s 时间序列. 在智能环境系统中,其环境的近况更为人所关注,数据时序越靠后则越重要. 根据这一情况所设计的滑动窗口方法将不断删除最早数据,同时不断加入新接收到的数据[4].

定义5　(滑动窗口)　已知时间长度 T,t 为相对于 S 的初始观测时刻的经过时间,而 $t>T$ 且为可变时刻,因此 $\mathrm{SW}[t-T:t]$ 表示了一个 S 的滑动窗口,且时间长度为 T[4].

3.2　规则解析与预处理

假设 $\mathrm{RS}[i]$ 所表示的规则子集由任何推理周期为 period_i 的规则组成,利用规则解析与预处理模块,能够得到由这个规则子集转化而来的规则网络. 这个规则网络执行网络过滤,以及网络传播的周期,即该规则网络的推理周期也为 period_i. 对于特定规则网络,需要对每个原子条件赋予独有索引号,记作 CID. 原子条件大多具有在时段区间内对原始数据进行特征计算,以及使特征值和阈值比较这两种计算的操作. 其中,特征计算相比于阈值计算,计算代价更大. 利用规则解析与预处理模块,能够再对原子条件进行解析,并提取得到原子条件的特征计算中所得出的规则子集的统计单元序列 $\mathrm{SU}[0], \mathrm{SU}[1], \mathrm{SU}[2], \cdots, \mathrm{SU}[m]$[4],此时,统计单元可被定义为

<CID, stat, wSize, streamID, period, value>

其中,CID 表示其统计单元所对应的原子条件的索引号;stat 表示统计运算符,具有属性(isInc, ref),isInc 表示能否进行可增量式计算,ref 表示能够被引用的特征;wSize 表示特征计算操作中的滑动窗口的大小;streamID 表示特征计算操作中的数据流 ID;period 表示该特征计算的计算周期;value 表示特征值且默认值为 0. 对于可增量式计算,其含义能够表示为:对于一组给定的历史样本值 $h_1, h_2, \cdots, h_M$,一组增量样本值 $a_1, a_2, \cdots, a_N$,全量样本值 $h_1, h_2, \cdots, h_M, a_1, a_2, \cdots, a_N$ 的统计值能够通过历史样本和增量样本的指标以计算方式获取[4]. 举例而言,对于全量样本的均值的计算可以用以下公式表达:

$$\bar{X}=\frac{M\bar{H}+N\bar{A}}{M+N} \tag{1}$$

而全量样本的方差的计算为

$$\sigma_X^2=\frac{M\left[\sigma_H^2+\left(\bar{X}-\bar{H}\right)^2\right]+N\left[\sigma_A^2+\left(\bar{X}-\bar{A}\right)^2\right]}{M+N} \tag{2}$$

其中,$\bar{H}$ 和 σ_H^2 代表历史样本的均值和方差;$\bar{A}$ 和 σ_A^2 代表增量样本的均值和方差. 某一特征的可引用特征的含义是在该特征进行计算时,能够对样本中的其他特征进行引用[4]. 以样本峰度为例:

$$g_2=\frac{m_4}{\left(\sigma^2\right)^2}-3 \tag{3}$$

而样本峰度能够对该样本的方差进行引用,样本的方差可表示为

$$\sigma^2=\frac{1}{n}\sum_{i=1}^{n}\left(x_i-\bar{x}\right)^2 \tag{4}$$

此外,样本方差同样能够继续引用该样本的均值,其中 m_4 为四阶样本中心距. 显然,特征的引用关系,具有传递的性质,特征之间能够相互引用甚至循环引用. 而在智能环境系统的相关应用中,典型的统计特征如表1所示[4].

表1　典型统计特征

统计运算符	编号	可增量式计算	可引用特征
MAX	0	1	null
MIN	1	1	null
RANGE	2	0	MAX, MIN
AVG	3	1	null
VAR	4	1	AVG
STD	5	1	VAR
SKEWNESS	6	0	STD
KURTOSIS	7	0	VAR

若规则解析与预处理模块能够解析的一个规则子集而获得的统计单元序列的特征计算,可表示为 $\mathrm{stat}_1(S_1, T_1)$, $\mathrm{stat}_1(S_1, T_2)$, $\mathrm{stat}_2(S_1, T_1)$ 和 $\mathrm{stat}_2(S_1, T_3)$,而滑动窗口长度 $T_1 \neq T_2 \neq T_3$ 且 $T_1 < T_2$,stat_1 可增量式计算,且不具备可引用特征,stat_2 不可增量式计算,具有引用特征,且为 stat_1. 因 stat_1 可进行增量式计算,所以全量样本 $S_1[t-T_2: t]$ 的特征值能够在样本 $S_1[t-T_1: t]$ 的基

础上对样本$S_1[t-T_2:t-T_1]$进行增量计算而获得，由此就消除了对基于SU[0]与SU[1]中样本$S_1[t-T_1:t]$的重复计算过程. 对于$\text{stat}_2(S_1,T_1)$，由于stat_1为stat_2的可引用特征且$\text{stat}_2(S_1,T_1)$与$\text{stat}_1(S_1,T_1)$的操作的滑动窗口能够彼此对齐，所以在计算$\text{stat}_2(S_1,T_1)$时，可立即引用$\text{stat}_1(S_1,T_1)$的特征值，消除了对其可引用特征$\text{stat}_1(S_1,T_1)$的重复计算过程. 而对于$\text{stat}_2(S_1,T_3)$，其可引用特征表示为stat_1，因所关联的滑动窗口$\text{SW}[t-T_3:t]$不与$\text{stat}_1(S_1,T_1)$和$\text{stat}_1(S_1,T_2)$关联的滑动窗口$\text{SW}[t-T_1:t]$和$\text{SW}[t-T_2:t]$彼此对齐，所以它们的特征值无法被直接引用. 如果任一特征stat_1的可引用特征为stat_2，则stat_1为引用特征，此时stat_2相应地则为被引用特征. 如果包含某一特征的统计单元序列中具备与之相关的可引用特征，并且此被引用特征也是可增量式计算的特征，以及满足它们操作的滑动窗口为非对齐窗口，为尽可能将引用特征和增量式计算的长处发挥出来，需要对统计单元序列采取预运算[4]. 统计单元序列中涉及的预处理操作由算法1描述.

算法1 统计单元序列预处理算法

Input: Statistical Unit Sequence $\text{SU}[0],\text{SU}[1],\text{SU}[2],\cdots,\text{SU}[m]$.

Output: Statistical Unit Sequence.

```
1. for SU[i] in SU[0],SU[1],SU[2],…,SU[m] do
2.     if SU[i].stat.ref != NULL
3.         for SU[j] in SU[0],SU[1],SU[2],…,SU[m] do
4.             if SU[i].stat.ref == SU[j].stat && SU[j]
                   stat.isInc&& SU[i].wSize != SU[j].wSize
                   && SU[i].streamID == SU[j].streamID
5.                 insert SU(-1,SU[i].stat.ref, SU[i]
                   wSize, SU[i].streamID, SU[i].period,0)
                   into Statistical Unit Sequence;
6. return Statistical Unit Sequence.
```

整体而言，本算法对统计单元序列采取了预运算，为了达到让其中的被引用特征也能够进行可增量式计算，但与被引用特征操作的滑动窗口不对齐的目的，本算法创建了新的统计单元，该统计单元对应的特征为被引用特征. 新的统计单元不与任何原子条件对应，其CID被初始化为−1. 例如，对于与$\text{stat}_1(S_1,T_1)$，$\text{stat}_1(S_1,T_2)$，$\text{stat}_2(S_1,T_1)$和$\text{stat}_2(S_1,T_3)$对应的统计单元序列，预运算后的新序列所相应的特征计算则为$\text{stat}_1(S_1,T_1)$，$\text{stat}_1(S_1,T_2)$，$\text{stat}_1(S_1,T_2)$，$\text{stat}_2(S_1,T_1)$和$\text{stat}_2(S_1,T_3)$. 其中$\text{stat}_1(S_1,T_3)$可在$\text{stat}_1(S_1,T_2)$的基础上由对样本$S_1[t-T_3:t-T_2]$采取增量计算而来，而$\text{stat}_2(S_1,T_3)$可直接引用$\text{stat}_1(S_1,T_3)$的特征值. 相较于直接计算$\text{stat}_1(S_1[t-T_3:t])$，计算$\text{stat}_1(S_1[t-T_3:t-T_2])$的成本更低[4].

3.3 轻量级特征表

在DPDS中，为了实现规则网络在输入数据流的滑动窗口内选择特征值的功能，系统需要通过采取某种方法保留特征值在历史时刻的状态. 而在本文中，所采用的方法是利用轻量级特征表(Light-weight Characteristic Table，LCT)完成所需求的功能. 轻量级特征表能够计算并存储原子条件的统计单元特征值，轻量级特征表与规则网络一一对应，具体而言，轻量级特征表由一组相同滑动窗口长度分组的列所定义. 因情况类同，后文仅就规则子集和数据流数目为1的情况进行展开讨论[4].

令$\text{stat}_1(S_1,T_1),\text{stat}_2(S_1,T_2),\cdots,\text{stat}_n(S_1,T_n)$对应一个预运算后的从规则子集得到的统计单元特征计算. 其相关的各种特征计算，在数据流S上操作的滑动窗口得到，所计算得出的序列则可记作$\text{SW}[t-T_1:t],\text{SW}[t-T_2:t],\cdots,\text{SW}[t-T_m:t]$，其中$m<n$. 有关可增量式的特征计算，则必须对滑动窗口摘要进行相应维护[4]. 本文针对滑动窗口的问题，采取了均匀划分的策略. 将最长的滑动窗口划分成若干连续的基本窗口，而新的滑动窗口可通过数个所划分出的连续基本窗口拼接得到[4]. 图4显示了滑动窗口以及所对应的基本窗口的拓扑关系.

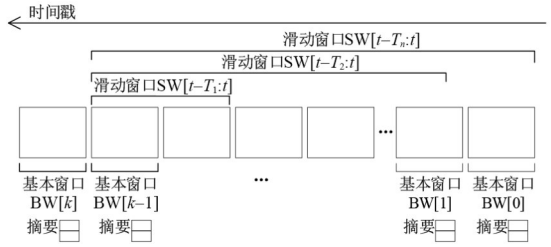


图4 滑动窗口与基本窗口

假设最长滑动窗口所对应的数据流子序列为$S[t-w:t]$. 令$w=kb$，其中k代表组成最长滑动窗口的基本窗口数量，b表示基本窗口长度. 令$\text{BW}[0],\text{BW}[1],\text{BW}[2],\cdots,\text{BW}[k-1]$表示最长滑动窗口SW中的基本窗口序列[4]，其中

$$\text{BW}[i]=S[t-w+ib:t-w+(i+1)b] \tag{5}$$

通过移动滑动窗口，新的基本窗口$\text{BW}[k]$不断得到，基本窗口$\text{BW}[0]$过期. 关键的一点在于如何设定基本窗口的长度，其原因在于在基本窗口$\text{BW}[i+1]$生成之前，需要引用$\text{BW}[i]$中的数据摘要[4]. 假设基本窗口$\text{BW}[i]$中第j个元素为$\text{BW}[i;j]$. 本文取规则子集当中，其推理周期，以及所有滑动窗口长度的最大公约数，用以设定基本窗口的长度，即基本时间单元(basic time unit)，用btu表示[4]为

$$\text{btu}=\text{GCD}(\text{period},T_1,T_2,\cdots,T_n) \tag{6}$$

基本窗口的摘要可用于计算出滑动窗口中的可增量式计算的特征值. 以计算滑动窗口的平均值为例,对于基本窗口BW[i],需要维护的摘要可表示[4]为

$$\text{digest}(\text{BW}[i])=\sum_{j=1}^{b}\frac{\text{BW}[i;j]}{k} \tag{7}$$

通过汇总基本窗口中维护的摘要,计算出滑动窗口SW[$t-T_j:t$]的摘要[4]为

$$\text{digest}(\text{SW}[t-T_j:t])=\sum_{l=1}^{\frac{T_j}{\text{btn}}-1}\text{digest}(\text{BW}[i])\left(\frac{T_j}{\text{btn}}\right) \tag{8}$$

当新的基本窗口BW[k]生成时,滑动窗口的摘要将被更新[4]为

$$\begin{aligned}\text{digest}_{\text{new}}(\text{SW})=&\text{digest}_{\text{old}}(\text{SW})\times\frac{T_j}{\text{btn}}\\&+\text{digest}(\text{BW}[k])\\&-\text{digest}(\text{BW}[0])\Big/\left(\frac{T_j}{\text{btn}}\right)\end{aligned} \tag{9}$$

令统计单元的滑动窗口长度组成的集合[4]为

$$\text{TS}=\left\{\text{ts}_i\middle|\,\text{ts}_i\subset\left\{T_0,T_1,T_2,\cdots,T_m\right\}\right\} \tag{10}$$

TSA$\subseteq$TS作为一个顺序集合,由统计运算符为可增量式计算的统计单元的滑动窗口长度所构成,TSB$\subseteq$TS为另一顺序几何,构成其的统计运算符为不可增量式计算的统计单元[4], TSA$\cup$TSB=TS,假设由可增量式计算的统计单元在数据流S上的组成的集合[4]为

$$SS=\left\{\text{stat}_i\middle|\,\text{stat}_i\subset\left\{\text{stat}_0,\text{stat}_1,\text{stat}_2,\cdots,\text{stat}_r\right\}\right\} \tag{11}$$

对于可进行增量式计算的统计单元,假设由TSA中相邻两元素的差组成的集合[4]为

$$\text{ND}=\left\{\text{diff}_i\middle|\,\text{diff}_i=\begin{cases}\text{ts}_0, & \text{if } i=0\\ \text{ts}_i-\text{ts}_{i-1}, & \text{if } i>1\end{cases}\right\} \tag{12}$$

则在滑动窗口SW[$t-\text{ts}_i:t$]上的特征值能够由在SW[$t-\text{ts}_{i-1}:t$]的基础上增量式计算SW[$t-\text{ts}_{i-1}-\text{diff}_i:t-\text{ts}_{i-1}$]的特征值求出,其中$\text{ts}_i,\text{ts}_{i-1}\subset$TSA. 有关滑动窗口SW[$t-\text{ts}_i:t$],如果$\text{stat}_j\subset$SS且$\text{stat}_j$操作的任意滑动窗口长度均未超过$\text{ts}_i$,则需要维护该窗口的摘要$\text{stat}_j(\text{SW}[t-\text{ts}_i:t])$. 对于不可增量式计算的统计单元,则采用直接计算的方式. 在对相同滑动窗口长度的所有统计单元进行操作的情况下,若一个特征为另一个的可引用特征,则引用特征在计算过程中引用被引用特征[4].

轻量级特征表由滑动窗口长度为TS$\cup$ND$\cup$btu中各元素分组的列组成. 其中按窗口长度为btu分组的列能够直接计算原始数据中的特征值,用以存储基本窗口的摘要;按窗口长度为ND各元素大小分组的列通过汇总基本窗口的摘要得到用于增量式计算的中间摘要;按窗口长度为TSA[i]分组的列通过汇总窗口长度分别为TSA[$i-1$]与ND[i]分组的列的摘要得到可增量式计算的特征值;按窗口长度为TSB各元素大小分组的列采用直接计算原始数据的方式得到不可增量式计算的特征值[4]. 轻量级特征表只在内存中缓存部分原始数据用于不可增量式计算特征计算,对于基本窗口,只保留摘要,丢弃原始数据[4]. 轻量级特征表的详细构造方法由算法2和算法3给出.

算法2　LCT构造算法(1)

Input: Statistical Unit Sequence SU[0], SU[1], SU[2], ⋯, SU[n], btu, TSA, TSB, ND, SS, t.

Output: LCT.

```
1. Columns=TSA ∪ TSB ∪ ND ∪ btu;
2. for i=0 to sizeof(SS)−1 do
3.     if SS[i].ref exist in SS
4.         SS[i] refer SS[i].ref; //SS[i] refers to the value of SS[i].ref
5.     insert SS[i](S[t-btu:t]) into digest;
6. insert digest into LCT[Columns.index(btu)][t]; //update btn columns
7. SS=NULL;
8. tmp=0;
9. for i=1 to sizeof(ND)−1 do
10.    digest=NULL;
11.    for SU[j] in SU[0], SU[1], SU[2], ⋯, SU[n] do
12.        if ND[i]<=SU[j].wSize && SU[j].stat.isInc
13.            insert SU[j].stat into SS;
14.    for j=0 to sizeof(SS)−1 do
15.        for k=0 to ND[i]/btu−1 do
16.            tmp=SS[j](tmp, LCT[0][t−k]);
17.        insert tmp into digest;
18.    insert digest into LCT[Columns.index(ND[i])][t]; //update ND
       columns
19.    SS=NULL;
20. for i=0 to sizeof(Columns−TSA−TSB)−1 do
21.    delete expired entries;
22. return LCT.
```

在算法2中,构造轻量级特征表采用的是滑动窗口长度TSA,TSB,btu和ND元素大小分组的列作为算法的输入. 其中算法的第2~6行,为一个循环结构,遍历了滑动窗口SS的各个元素. 经过存在性检验后,该循环以插入和引用的方式对所要构造的LCT部分不断更新,构造了由滑动窗口长度为btu分组的列. 9~18行则是另一个双重循环结构,在插入更新的过程中,对滑动窗口ND的元素的值和SU的值不断比较后再进行添加,其作用为构造滑动窗口长度为ND的,每个元素大小分组的列. 算法2在每个基本窗口被创建时被调用,其中在一系列的取值较为特殊,作为中间列. 这些列的

元素分组所在的滑动窗口长度为btu和ND[4].

算法3 LCT构造算法(2)

Input: Statistical Unit Sequence SU[0], SU[1], SU[2], …, SU[n], btu, TSA, TSB, ND, SS, t.

Output: Light-Weight Characteristic Table LCT.

Initialization: SS=NULL.

1. Columns=TSA ∪ TSB ∪ ND ∪ btu;
2. insert LCT[Columns.index(ND[0])][t] into LCT[Columns.index(TSA[0])][t]; //update ND columns
3. **for** i=1 to sizeof(TSA)−1 **do**
4. **for** SU[j] in SU[0], SU[1], SU[2], …, SU[n] **do**
5. **if** TSA[i]<=SU[j].wSize && SU[j].stat.isInc
6. insert SU[j].stat into SS;
7. **for** j=0 to sizeof(SS)−1 **do**
8. insert SS[j](LCT[TSA[i−1]][t], LCT[ND[i]][t−TSA[i−1]] into digest; // update TSA columns
9. insert digest into LCT[Columns.index(TSA[i])][t]
10. SS=NULL
11. **for** i=0 to sizeof(TSB)−1 **do**
12. **for** SU[j] in SU[0], SU[1], SU[2], …, SU[n] **do**
13. **if** SU[j].wSize==TSB[i] && !SU[j].stat.isInc
14. insert SU[j].stat into *SS*;
15. **for** j=0 to sizeof(*SS*)−1 **do**
16. **if** SS[j].ref exist in SS
17. SS[j] ref SS[i].ref;
18. insert SS[j](*S*[t−TSB[i]:t]) into digest;
19. insert digest into LCT[Columns.index(TSB[i])][t]; //update TSB columns
20. **for** i=0 to sizeof(Columns−btu−ND)−1 **do**
21. delete expired entries;
22. return LCT.

算法3的输入与算法2相同. 其目标是构造LCT中,滑动窗口长度是TSA以及TSB中各元素分组的列,这些列存储了对应原子条件的统计单元的特征值. 其中2~9行构造了由窗口长度为TSA的各元素大小分组的列,10~19行构造由窗口长度分别为TSB的各元素分组的列,19~21行删除每列中的过期表项. 算法3的调用周期为规则子集的推理周期,称算法3构造的包含由滑动窗口长度为TSA和TSB各元素分组的列为LCT的目标列,原子条件所需引用的特征值存储于目标列中. 对于轻量级特征表的形象描述,图5所示的是某老年人睡眠监测应用中的LCT. 需要说明的是,该表所使用生成规则子集较为简单,其推理周期设定为5分钟. 对睡眠期间的身体运动和姿势所采取的监测仪器为地震检波器[4]. 在应用中,所涉及的某个规则子集展示如下:

R1: IF AVG(S1,4MIN)≤0.632 AND VAR(S1,11MIN)≤0.221 THEN A1=1

R2: IF MAX(S1,7MIN)>1.112 AND AVG(S1,9MIN)>0.436 AND KURTOSIS(S1,9)>0 THEN A2=1

该规则子集所对应的TSA={4MIN, 7MIN, 9MIN, 11MIN},ND={4MIN,3MIN,2MIN,2MIN},TSB={11MIN},SS={MAX,AVG,VAR},btu=1MIN.

图5所示的LCT,其形状类似左下三角形. 这是因为该LCT对当前时刻到最大滑动窗口长度为止的所有历史时刻的基本窗口摘要都进行了存储. 而将基本窗口摘要进行汇总后,则可以获得由滑动窗口长度为ND的各元素分组的列的摘要. 由此能够再进行增量式计算,一步步算出每一个统计单元对应的特征值[4].

3.4 LCT预部署策略

如果采取尽可能地存储特征值的轻量级特征表于内存中的做法,则可大量节省系统I/O. 而在实际情况

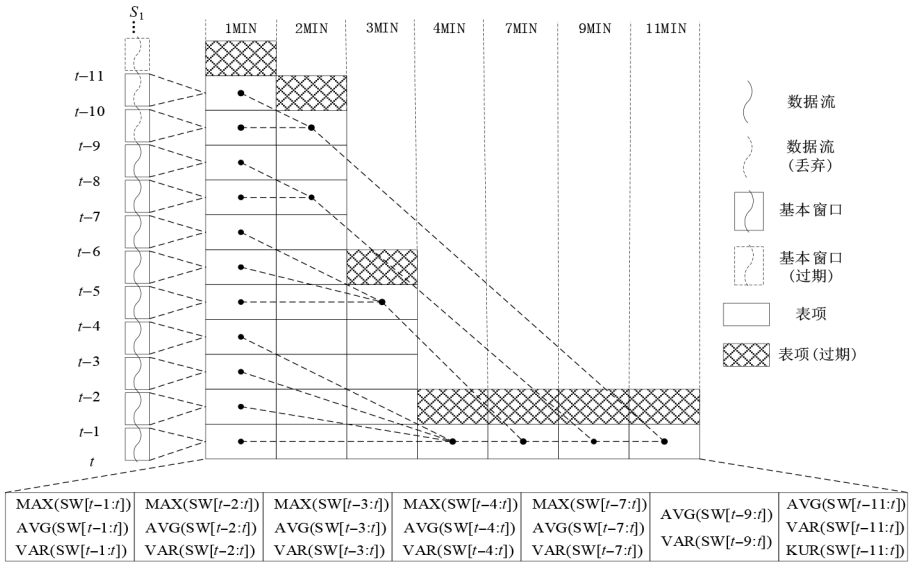


图5 轻量级特征表

下,边缘设备的在智能环境系统中资源可用量通常有限.一旦有智能环境的规模扩大的需求,特征量的存储就会因为规则规模的扩展而扩展.而这带来的结果则是,轻量级特征表的数量,以及与之相关内存占用量会不断大量增长,进而使数据在内存和磁盘之间的交换次数成倍增加,也同样会对系统I/O造成巨大负担.由此可知,数据交换的时机、策略的取舍对于适应智能环境系统中内存被限制的情景至关重要.系统的内存占用在智能环境系统中主要由规则网络、轻量级特征表和缓存的原始数据组成.内存占用量对于基于系统的部分,拥有一定的静态可预测性.这是一种在规则集确定后,能够由规则解析并预处理的模块,进而明确获取LCT的数量和内存占用量的特性.以此本文设计了一种基于规则集预分析和预处理的LCT预部署策略[4].

若某一规则子集中,其由统计单元当中的滑动窗口长度组成的集合分别为TS,TSA,TSB,其表示的是由可增量式计算以及不可增量式计算的统计单元的滑动窗口长度组成的集合,TSA, TSB $\subseteq$ TS所对应的元素个数分别为α和β,而集合SS的元素个数为γ,基本时间单元为btu.该规则子集构建的轻量级特征表中,中间列的表项数量最多[4]可通过如下公式计算得出:

$$\frac{\mathrm{TSA}[\alpha]}{\mathrm{btu}}+\sum_{i=0}^{\alpha-2}\frac{\mathrm{TSA}[i]+1}{\mathrm{btu}} \tag{13}$$

其中由基本窗口长度分组的列的表项数量为TSA[α]/btu,由滑动窗口长度为ND各元素分组的列的表项数量[4]不超过

$$\sum_{i=0}^{\alpha-2}\frac{\mathrm{TSA}[i]+1}{\mathrm{btu}} \tag{14}$$

目标列中各个列表项数量为1,则LCT表项数量最大[4]为

$$\frac{\mathrm{TSA}[\alpha]}{\mathrm{btu}}+\sum_{i=0}^{\alpha-2}\frac{\mathrm{TSA}[i]+1}{\mathrm{btu}}+\alpha+\beta \tag{15}$$

LCT中各表项占用的内存空间不大于$\gamma\times A$,则轻量级特征表占用的内存空间[4]不大于

$$\left(\frac{\mathrm{TSA}[\alpha]}{\mathrm{btu}}+\sum_{i=0}^{\alpha-2}\frac{\mathrm{TSA}[i]+1}{\mathrm{btu}}+\alpha+\beta\right)\times\left(\gamma\times A\right) \tag{16}$$

对于流速为λ的数据流,缓存的原始数据大小为不可增量式计算的运算符操作的滑动窗口最大者[4]为

$$\mathrm{TSB}\left[\beta\right]\times\lambda\times A \tag{17}$$

对于N表示的规则集中的规则数目,n表示原子条件的数量,由于规则网络是由树组织起来的网络结构,所得出的规则网络总的内存占用量[4]为

$$(2n-1+N)\times A \tag{18}$$

其中,A代表每个元素(摘要、原子条件、响应)的内存占用常量.因为需要考虑系统内存受限的情况,应该尽量将更新频率高的轻量级特征表存储在内存中.所以给更新频率高的轻量级特征表设置高优先级,给更新频率低的轻量级特征表设置低优先级.如果系统中当前可用的内存资源为M,当所有规则子集的规则网络、轻量级特征表和原始数据的内存占用量超过M时,将轻量级特征表和原始数据按优先级由低到高进行离线操作,并在内存中设置一个缓存块.重复这一过程直到内存占用总和低于M[4].

有关离线的轻量级特征表,需要在内存中为它们保存TSA,TSB和SS等信息,并且当每个基本窗口创建完成时,将基本窗口所包含的原始数据以及摘要写入磁盘.各基本窗口的索引信息[4]可表示为<streamID, BWstart, BWend>.而在离线轻量级特征表对应的规则网络的推理时刻出现前的时间内,预取$t-\mathrm{TSA}(\alpha)$至t之间的所有基本窗口摘要和$t-\mathrm{TSB}(\beta)$至t之间的所有原始数据,并在缓存块中完成离线轻量级特征表的生成构造.离线轻量级特征表的构造算法由算法4给出[4].

算法4　离线LCT构造算法

Input: Statistical Unit Sequence SU[0], SU[1], SU[2], ⋯, SU[n], btu, TSA, TSB, SS, t.

Output: offline LCT.

Initialization: SS=NULL.

```
1. Columns=TSA ∪ TSB ∪ btu;
2. insert digest into LCT[Columns.index(btu)];
3. tmp=0;
4. for i=0 to sizeof(TSA)−1 do
5.     digest=NULL;
6.     for SU[j] in SU[0], SU[1], SU[2], ⋯, SU[n] do
7.         if TSA[i]<=SU[j].wSize && SU[j].stat.isInc
8.             insert SU[j].stat into SS;
9.     for j=0 to sizeof(SS)−1 do
10.        for k= TSA[i−1]/btu to TSA[i]/btu−1 do
11.            tmp=SS[j](tmp, LCT[0][t−k]);
12.        if i=0
13.            insert tmp into digest;
14.        else
15.            insert SS[j](tmp, LCT[Columns.index(TSA[i−1])][t])
       into digest; //update TSA columns
16.        tmp=0;
17.    insert digest into LCT[Columns.index(TSA[i])][t];
18. for i=0 to sizeof(TSB)−1 do
19.    for SU[j] in SU[0], SU[1], SU[2], ⋯, SU[n] do
20.        if SU[j].wSize==TSB[i] && !SU[j].stat.isInc
21.            insert SU[j].stat into SS;
22.    for j=0 to sizeof(SS)−1 do
23.        if SS[j].ref exist in SS
24.            SS[j] ref SS[i].ref;
25.        insert SS[j](S[t−TSB[i]:t]) into digest;
26.    insert digest into LCT[Columns.index(TSB[i])][t];
27. return offline LCT.
```

离线LCT不构造中间列，而是直接采用增量式计算的方式构造目标列. 算法第2行用预取的基本窗口摘要构造了离线轻量级特征表由窗口长度为btu分组的列，第4~17行构造由滑动窗口长度为TSA各元素分组的列，第18~26行构造由滑动窗口长度为TSB各元素分组的列. 以3.3节中老年人睡眠监测应用中所使用的规则子集为例，图6阐述了离线轻量级特征表的构造过程[4].

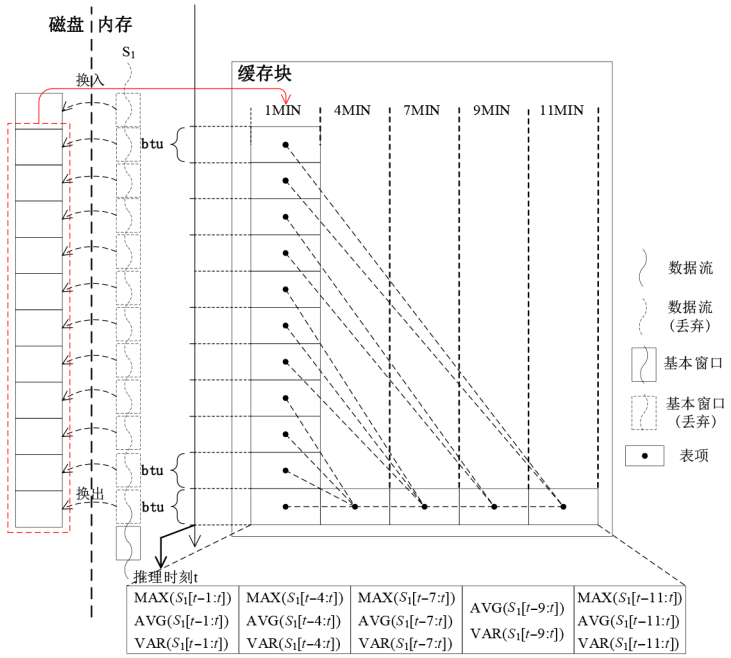


图6 轻量级特征表预部署方案

4 实验验证与结果分析

在这一节，对DPDS的性能进行测试评估，采用模拟的规则集来评估DPDS的表现. 为了评估DPDS的普适性和泛化能力，本文设置两组规则集分别进行实验测试. 其中一组规则集中条件重复率设为75%，另一组规则集中条件重复率设为25%. 与条件重复率较低的规则集相比，在规则数目相同的前提下条件重复率较高的规则集在规则或更细粒度的计算单元上存在更多地交叉重叠的部分，相应地可增量式计算的计算单元也更多. 具体来说，条件重复率为75%的规则集比条件重复率为25%的规则集中的重复条件和可增量式计算的计算单元多约3倍. 通过对比条件重复率为75%和条件重复率为25%的规则集在性能表现上的差异可以对DPDS的普适性和泛化能力进行评估. 若无说明，本文在其他对比实验中默认采用条件重复率为75%的规则集进行实验测试. 为了灵活设置数据流流速，本文利用了大量的历史数据来模拟到达的数据流对DPDS的性能进行测试，并与现有推理系统、文献[13]所提出的采用模糊处理和B+树结构的RETE算法和使用RETE算法的推理系统对比进行分析. 文献[13]所提出的采用模糊处理和B+树结构的RETE算法在RETE算法基础上将精确数据转换为模糊概念以方便用户自定义规则，使用B+树加快了非空节点的遍历速度，相比于RETE算法具有非常好的性能优势. 实验中，算法使用Visual C++实现，实验的硬件环境为CPU主频1 GHz、主存512 MB的嵌入式开发板. 本节的实验测试了现有推理系统、采用RETE算法的推理系统、采用模糊处理和B+树结构的RETE算法和采用DPDS的推理系统的时间以及算法的内存开销. 本节设置3组实验，分别考察了3种情况下算法的性能：

(1)固定规则数目和最大窗口长度，变化流速的情况；

(2)固定规则数目和流速，变化最大窗口长度的情况；

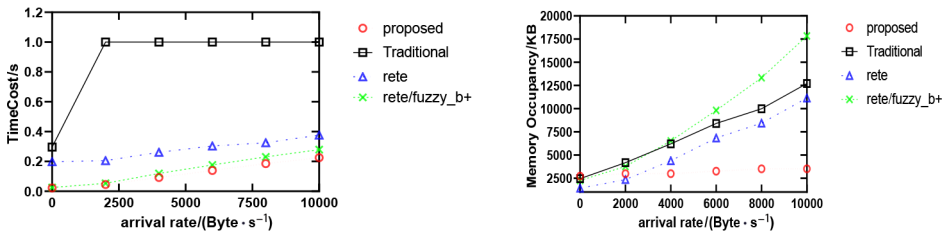
(3)固定流速和最大窗口长度，变化规则数目的情况.

在规则网络的匹配过程中，不同匹配路径对算法的时间性能有较大影响. 为了确保RETE算法、采用模糊处理和B+树结构的RETE算法和DPDS在匹配时具有相同匹配路径，本文在实验中使用了相同的历史数据模拟了输入至推理系统的数据流，同时，在第三种情况中规则的增加是通过重复增加原有规则实现的. 通过控制匹配过程中的规则和数据，本文使不同对比实验在大致相同的匹配深度和匹配成功率下进行，其中

匹配成功率约为50%.

4.1 固定规则数目和最大窗口长度

在规则数目为5 000,最大窗口长度为8 min,数据流流速变化的情况下,4种系统的时间开销如图7(*a*)所示. 现有推理系统在数据流流速为200 Byte/s时的时间消耗约为0.3 s. 当数据流流速接近2 000 Byte/s时,现有推理系统的时间消耗约为1 s,这意味着现有推理系统无法应对当前的数据流流速(2 000 Byte/s). 对于采用RETE算法和采用模糊处理和B+树结构的RETE算法的推理系统,时间消耗随着数据流流速的增长而增长. 采用DPDS的推理系统优于采用RETE算法和采用模糊处理和B+树结构的RETE算法的推理系统,这是因为DPDS使用轻量级特征表通过基本窗口对计算任务进行了分割,且采用增量式的方式计算可增量式计算的统计单元,以及通过引用可引用特征大大减少了计算量. 图7(*b*)给出了4种推理系统的内存使用情况. 对于采用DPDS的推理系统,系统使用的内存主要由规则网络、轻量级特征表以及对少量原始数据的缓存组成. 当数据流流速增长时,采用DPDS的推理系统仅增加少量增量原始数据的缓存. 采用RETE算法的推理系统使用更多的内存用于缓存完整原始数据用于规则匹配. 而采用模糊处理和B+树结构的RETE算法的推理系统由于在RETE网络中引入了B+树结构加速规则匹配效率,导致它与采用RETE算法的推理系统相比具有更大的内存占用量. 现有推理系统的内存使用情况与采用RETE算法的推理系统相似,但使用更多内存用于存储规则集中的规则.



(*a*) 时间消耗　　(*b*) 内存占用量

图7 不同数据流流速下数据预部署策略的性能

4.2 固定规则数目和流速

在本节的实验中,规则数目为5 000,数据流流速为1 000 Byte/s,设置了最大窗口长度逐渐增长的6组实验,实验编号及对应规则数目如表2所示. 图8(*a*)和图8(*b*)分别展示了在不同最大窗口长度下4种系统的时间和内存消耗对比.

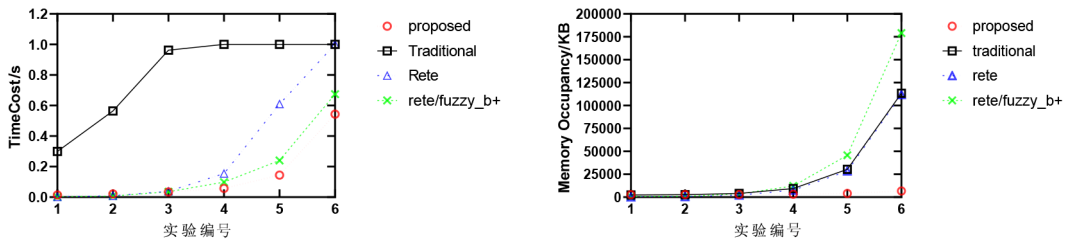
表2 不同最大窗口长度下的实验参数

实验编号	1	2	3	4	5	6
最大窗口长度	8	16	32	64	128	256

如图8(*a*)所示,现有推理系统在最大滑动窗口长度为8 min时,系统的时间消耗约为0.3 s. 当最大滑动窗口接近32 min时,系统时间消耗接近1 s. 这意味着现有推理系统无法应对最大窗口长度大于32 min的情况. 采用RETE算法的推理系统的时间消耗随着最大窗口长度的增长急剧增长,无法应对最大窗口长度大于256 min的情况. 采用DPDS的推理系统构建的轻量级特征表随着最大窗口长度的增长变得更加复杂,但实时性仍然优于现有推理系统和采用RETE算法的推理系统. 采用模糊处理和B+树结构的RETE算法的推理系统的时间消耗与采用DPDS的推理系统非常接近. 在图8(*b*)中,现有推理系统使用的内存主要由用于规则匹配的完整原始数据组成,因此内存消耗随着最大窗口长度的增长急剧增长. 采用RETE算法的推理系统消耗内存的情况与现有推理系统相似. 采用DPDS的推理系统的内存消耗几乎不受最大窗口长度的影响,这是因为它使用轻量级特征表将最大窗口分解成若干基本窗口,且对于可增量式计算的统计单元只保存基本窗口的摘要而丢弃原始数据.

4.3 固定流速和最大窗口长度

本节实验测试了在固定数据流流速为1 000 Byte/s和最大窗口长度为64 min,变化规则数目的情况下DPDS的性能,以及DPDS在不同条件重复率的规则集下的性能表现. 当系统更换规则集时,DPDS将根据新的规则集重新构建规则网络和轻量级特征表. 本节测试了当规则数目为n时,重新构建规则网络和轻量级特征表的时间消耗(T_{LCT}和T_{NET}). 如表3所示,当规则数目设置为100 000时,重新构建规则网络和轻量级特征表的时间消耗分别为0.637 s和0.72 s. 而当规则数目为1 600万时,重新构建规则网络和轻量级特征表的时间消耗为63.7 s和246.3 s. 尽管看起来时间消耗很大,但实际应用中的规则数目通常小于100万,且规则集更换不频繁.



(a) 时间消耗 (b) 内存占用量

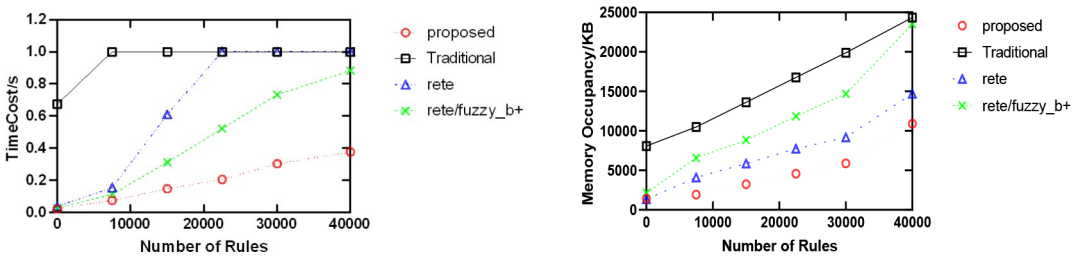
图8 不同窗口长度下数据预部署策略的性能

表3 不同规则数目下的代价分析

n(million)	0.01	0.1	1	2	4	8	16
T_{LCT}/s	0.074	0.72	7.48	16.4	39.6	100.1	246.3
T_{NET}/s	0.006	0.637	0.762	1.42	12.8	37.6	63.7

4.3.1 条件重复率为75%

在这一小节中,本文测试了4种系统在条件重复率为75%的规则集下的时间和内存开销. 图9(a)给出了4种系统的时间消耗. 当规则数目约为7 500时,现有推理系统无法及时对所有规则进行匹配. 采用RETE算法的推理系统与现有推理系统相似,当规则数目约为22 500时,系统无法及时完成网络筛选和网络传播. 相较于现有推理系统和采用RETE算法的推理系统,采用模糊处理和B+树结构的RETE算法的推理系统和采用DPDS的推理系统受规则数目的影响非常小. 如图9(b)所示,采用DPDS的推理系统的内存消耗与采用RETE算法和采用模糊处理和B+树结构的RETE算法的推理系统相似,它们的规则网络随着规则数目的增加变得越来越复杂,而B+树的引入导致采用模糊处理和B+树结构的RETE算法更多地占用系统内存. 现有推理系统由于要在内存中缓存规则集的完整规则信息,因此耗费更多的内存.

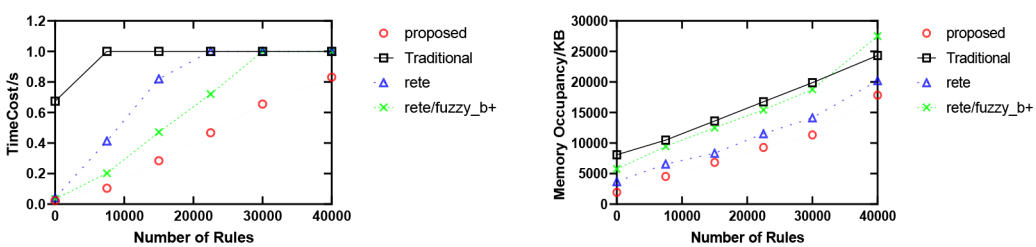


(a) 时间消耗 (b) 内存占用量

图9 不同规则数目下数据预部署策略的性能(规则重复率为75%)

4.3.2 条件重复率为25%

本节实验测试了4种系统在条件重复率为25%的规则集下的时间和内存开销. 如图10所示,随着条件重复率的降低,采用RETE算法、采用模糊处理和B+树结构的RETE算法和采用DPDS的推理系统在时间和空间性能表现上均有所下滑. 采用RETE算法和采用模糊处理和B+树结构的RETE算法的推理系统相似,随



(a) 时间消耗 (b) 内存占用量

图10 不同规则数目下数据预部署策略的性能(规则重复率为25%)

着条件重复率的下降,需要使用更多内存和时间来用于构建更多节点和更复杂的规则网络. 同样地,随着条件重复率的下降,DPDS也必须投入更多内存来扩展轻量级特征表的规模和构建更大的规则网络. 而规模更大的轻量级特征表进一步增加了完成增量式计算所花费的时间,导致DPDS的时间性能下滑较为明显. 然而,在与其他方法的对比实验中,DPDS仍然保持着不错的性能优势,由此可以说明DPDS具有不错的普适性和泛化能力.

5 结论

针对在基于无线传感器网络的智能环境中,规则匹配期间实时特征计算量过大和边缘设备内存资源受限的问题,本文设计了一种基于边缘设备规则推理的数据预部署方案. 该方案主要包含3个部分:规则的解析及预处理、轻量级特征表以及LCT预部署策略. 其中规则即系及预处理部分通过对规则进行解析完成了规则网络的构建和统计单元的生成;通过分析统计单元,生成了用于存储特征值的轻量级特征表,利用轻量级特征表,可以完成实时特征的预计算,从而避免了规则匹配期间的实时特征计算,加快了规则匹配速度;LCT预部署策略通过对占用内存量的预分析和轻量级特征表的IO策略解决了系统内存受限的问题. 仿真实验结果显示,在大规模的规则数量、数据数量等条件下,该方案都表现出了较好性能.

本文主要考虑对单数据流的特征计算,如计算原始数据均值、方差、峰度等. 多数据流的特征计算,如计算两条数据流间的协方差、相关系数等在金融、医疗等领域有非常广阔的应用前景,这将是今后研究需要关注的问题.

参考文献

[1] COOK D. Smart Environments: Technology, Protocols and Applications[M]. Hoboken, NJ: Wiley, 2005.

[2] 汪成亮, 王小均. 基于三轴传感器的老年人日常活动识别[J]. 电子学报, 2017, 45(3): 570-576.
WANG C L, WANG X J. Daily activity recognition based on triaxial accelerometer of elderly people[J]. Acta Electronica Sinica, 2017, 45(3): 570-576. (in Chinese)

[3] LONG X, FONSECA P, HAAKMA R, et al. Actigraphy-based sleep/wake detection for insomniacs[C]//2017 IEEE 14th International Conference on Wearable and Implantable Body Sensor Networks. Eindhoven: IEEE, 2017: 1-4.

[4] 汪成亮, 赵凯. 一种智能环境下基于边缘设备规则推理的数据预部署方法: CN112651506A[P]. 2021-04-13.
WANG C L, ZHAO K. Data Pre-Deployment Method Based on Edge Device Rule Reasoning in Intelligent Environment: CN112651506A[P]. 2021-04-13. (in Chinese)

[5] FORGY C L. Rete: A fast algorithm for the many pattern/many object pattern match problem[J]. Artificial Intelligence, 1982, 19(1): 17-37.

[6] ZHOU R L, PAN J F, TAN X B, et al. Application of CLIPS expert system to malware detection system[C]//2008 International Conference on Computational Intelligence and Security. Suzhou: IEEE, 2008: 309-314.

[7] CHEN W, OUYANG D T, YE Y X. RIF2Jess: inferencing RIF rules *via* translation to jess rules[C]//2009 International Conference on Computational Intelligence and Software Engineering. Wuhan: IEEE, 2009: 1-6.

[8] PROCTOR M. Relational declarative programming with JBoss drools[C]//Ninth International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2007). Timisoara: IEEE, 2007: 5.

[9] HERRING C, MILOSEVIC Z. Implementing B2B contracts using Bi$_z$Talk[C]//Proceedings of the 34th Annual Hawaii International Conference on System Sciences. Maui: IEEE, 2001: 6859651.1-6859651.10.

[10] ROTTENSTREICH O, COHEN R, RAZ D, et al. Exact worst case TCAM rule expansion[J]. IEEE Transactions on Computers, 2013, 62(6): 1127-1140.

[11] WANG Y M, YANG L H, FU Y G, et al. Dynamic rule adjustment approach for optimizing belief rule-base expert system[J]. Knowledge-Based Systems, 2016, 96: 40-60.

[12] BATORY D. The LEAPS algorithm[C]//Computer Science, Austin: University of Texas at Austin, 1994: 60693236.

[13] CHEN J F, LI Y H. Improved rule engine based on RETE algorithm for smart-home environment[C]//2019 IEEE 19th International Conference on Communication Technology. Xi' an: IEEE, 2019: 344-349.

[14] GUO C, XIONG W, HAO L Y. An improved rete algorithm with branch filtration[J]. Procedia Engineering, 2017, 174: 767-772.

[15] WOENSEL W VAN, ABIDI S S R. Optimizing semantic reasoning on memory-constrained platforms using the RETE algorithm[C]//European Semantic Web Conference. Cham: Springer, 2018: 682-696.

[16] YAY E, MADRID N M, RAMÍREZ J A O. Using an improved rule match algorithm in an expert system to detect broken driving rules for an energy-efficiency and safety relevant driving system[J]. Procedia Computer Science,

2014, 35: 127-136.
[17] KIM M, LEE K, KIM Y, et al. RETE-ADH: An improvement to RETE for composite context-aware service[J]. International Journal of Distributed Sensor Networks, 2014, 10(4): 507160.
[18] PENG X L, ZHENG L J, LIAO T. RCEDM: A rete-based RFID complex event detection method[C]//International Conference on Data Mining and Big Data. Cham: Springer, 2017: 137-147.
[19] LIU D, GU T, XUE J P. Rule Engine based on improvement Rete algorithm[C]//The 2010 International Conference on Apperceiving Computing and Intelligence Analysis Proceeding. Chengdu: IEEE, 2010: 346-349.
[20] YANG P L, YANG Y L, WANG N. IRETE: An improved RETE multi-entity match algorithm[C]//2011 International Conference on Electronics, Communications and Control(ICECC). Ningbo: IEEE, 2011: 4363-4366.
[21] XUE L L, LIU Y, TONG X, et al. A two-level reasoning method based on SVM_RETE algorithm in industrial environments[C]//2017 IEEE 17th International Conference on Communication Technology. Chengdu: IEEE, 2017: 1920-1925.
[22] HUBRECHTS M, KAMBONA K, RENAUX T, et al. Mete: A meta rete interface for distributed rule-based systems[C]//Proceedings of the 17th Belgium-Netherlands Software Evolution Workshop. Delft: CEUR, 2018: 29-32.
[23] ARAUJO D A, HENTGES A R, RIGO S J, et al. Applying parallelization strategies for inference mechanisms performance improvement[J]. IEEE Latin America Transactions, 2018, 16(12): 2881-2887.
[24] LIU Y, ZHANG T S, LI S C, et al. A real-time reasoning engine for injection-production optimization of water and oil wells on account of bitmap[J]. Acta Automatica Sinica, 2017, 43(6): 1007-1016.
[25] KIM K, KIM B W, LIM S, et al. Two-step filtering for rules without overlapping conditions[C]//2019 34th International Technical Conference on Circuits/Systems, Computers and Communications(ITC-CSCC). JeJu: IEEE, 2019: 1-4.
[26] WALZER K, SCHILL A, LÖSER A. Temporal constraints for rule-based event processing[C]//Proceedings of the ACM First Ph.D. Workshop in CIKM. New York: ACM, 2007: 93-100.
[27] WALZER K, BREDDIN T, GROCH M. Relative temporal constraints in the Rete algorithm for complex event detection[C]//Proceedings of the Second International Conference on Distributed Event-Based Systems. New York: ACM, 2008: 147-155.
[28] BERSTEL B. Extending the RETE algorithm for event management[C]//Proceedings of the Ninth International Symposium on Temporal Representation and Reasoning (TIME´02). New York: ACM, 2002: 49-51.
[29] WU E, DIAO Y L, RIZVI S. High-performance complex event processing over streams[C]//Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data. New York: ACM, 2006: 407-418.

作者简介

汪成亮 男，1975 年 5 月出生，四川资阳人. 博士. 现为重庆大学计算机学院教授，博士生导师. 主要研究方向为复杂系统智能控制、无线网络及 RFID 研究与应用等.
E-mail: wangcl@cqu.edu.cn

赵 凯 男，1995 年 10 月出生，湖北潜江人. 重庆大学在读硕士研究生. 主要研究方向为智能环境、物联网.
E-mail: 20144732@cqu.edu.cn