DOI:10.19652/j.cnki.femt.2204580

# 基于 RS_Hash 频繁项集的卫星载荷关联规则算法*

贾澎涛 温 滋
(西安科技大学计算机科学与技术学院 西安 710054)

**摘 要**:遥测数据是反映卫星健康状态的重要依据,对遥测载荷数据进行关联性分析,在一定程度上能反映出卫星的整体运行情况的好坏。针对传统关联规则算法存在效率低下、占用内存过多的问题,提出一种基于 RS_Hash 频繁项集的卫星载荷关联规则算法。首先对事务数据库使用动态随机抽样的方法获取样本数据,设计抽样误差和抽样停止规则来确定最优的样本容量;其次将抽取出的样本使用哈希桶来存储频繁项集,进而减少占用的内存,提高算法的运行效率;最后使用 3 个与载荷数据相似的公开数据集和卫星载荷数据集进行实验,结果表明,在公共数据集上取得了良好的效果,尤其是在具有大数据量级的卫星载荷数据集上效果明显,在不同事务长度和支持度的情况下,相较于 Apriori、PCY、SON、FP-Growth、RCM_Apriori 和 Hash_Cumulate 算法,RS_Hash 算法在平均时间效率上分别提高了 75.81%、49.10%、59.38%、50.22%、40.16%和 39.22%。

**关键词**:卫星载荷分析;关联规则;频繁项集;动态随机抽样算法;哈希桶
**中图分类号**: TP301.6 **文献标识码**:A **国家标准学科分类代码**: 520.10

# Algorithm of satellite payload association rules based on RS_Hash frequent item sets

Jia Pengtao Wen Zi
(College of Computer Science and Technology, Xi′an University of Science and Technology, Xi′an 710054, China)

**Abstract**: Telemetry data is an important basis to reflect the health status of satellites. The correlation analysis of telemetry load data can reflect the overall operation status of satellites to a certain extent. Aiming at the problems of low efficiency and excessive memory consumption of traditional association rule algorithm, a satellite load association rule algorithm based on RS_Hash frequent item set was proposed. Firstly, the dynamic random sampling method is used to obtain the sample data of the transaction database, and the sampling error and sampling stop rules are designed to determine the optimal sample size. Secondly, hash buckets are used to store frequent item sets in the extracted samples, thus reducing the occupied memory and improving the operation efficiency of the algorithm. Finally, three public data sets which are similar to the load data and satellite load data sets are used to carry out experiments. The results show that good results are achieved on the public data sets, especially on the satellite load data sets with large data magnitude. Under the condition of different transaction lengths and support degrees, compared with Apriori, PCY, SON, FP-Growth, RCM_Apriori and Hash_Cumulate algorithm, the average time efficiency of RS_Hash algorithm is improved by 75.81%, 49.10%, 59.38%, 50.22%, 40.16% and 39.22%.

**Keywords**: satellite load analysis; association rules; frequent item sets; random sampling algorithm; Hash buckets

## 0 引 言

卫星作为一个典型的复杂系统,其包含的有效载荷设备之间、载荷设备内部各零部件之间均存在着广泛的电性、数据接口以及复杂的系统交互,这些都决定了有效载荷数据之间存在必然的关联关系[1-2]。例如卫星的加热器打开,会产生其主电流增大、壳温升高、电源功率下降等影响,而传统航天器异常检测方法的知识和模型很难自动化构建,通常使用遥测数据的相关性方法找出指令数据之间的关联关系,进而判断航天器的运行规律,及时进行异常

**收稿日期**:2022-12-07
* **基金项目**:西安市科技计划(2020KJRC0069)项目资助

检测[3]。因此通过关联分析，找出数据之间的相关关系，能够提高分析效率，及早发现影响卫星特性的直接原因。

数据挖掘是从随机的、不完全的大量数据中，找出隐藏在其中的有价值信息的过程，而关联规则是数据挖掘中的重要方法[4]。关联规则通过分析寻找给定数据集中数据项之间的关联关系，来描述数据之间的密切程度。关联规则算法最早由 Agrawal 等[5]于 1993 年提出，并在 1994 年提出经典的 Apriori 算法，这是人们对关联规则数据挖掘研究的开始。通过文献分析和对比，Apriori 算法具有数据库消耗大、内存占用大以及算法效率低下的问题。在面对这些问题时，学者们纷纷从不同的角度和思想对其进行改进。

针对 Apriori 算法占用内存过大的问题，Agrawal 等[5]基于 Apriori 算法提出了其改良算法 AprioriTid 和 Apriori Hybird，在计算候选项集的支持度时提升了算法的效率。毕玉萍等[6]提出了基于排序树的 Node-Apriori 改进算法，减少了内存。胡世昌[7]提出了基于二进制编码的 Apriori 改进算法，对事务记录进行二进制编码后 加载到内存，有较好的性能。王伟等[8]提出了基于 MapReduce 的 Apriori 前后项约束关联规则改进算法，提高了对海量数据的处理能力和效率。程江洲等[9]针对电网故障难以快速挖掘有效信息的问题，将传统 Apriori 算法与自编码算法结合，有效地减少了运行时间，提升了故障检测的准确率。

针对 Apriori 算法需要多次扫描数据库，效率不高的问题，Han 等[10]提出 FP-Growth 算法，采用分而治之的策略，不产生候选项集，通过对数据库的一次扫描将其压缩到一棵频繁模式树中，减少了数据库的扫描次数。Li 等[11]基于深度剪枝策略提出了 MAR-DPS 算法，该算法利用原始 Apriori 算法压缩候选项集的规模，降低产生频繁项集的连接数。李涛等[12]将兴趣度关联规则应用在气象观测设备的一致性检验上，相较于同类算法在时间性能上更加优越。叶峰[13]提出了基于二分法改进的关联规则算法，能有效减少频繁项集的迭代过程和运算时间。刘彦戎等[14]提出了一种改进的矩阵关联规则提取算法，该算法具有较高的执行效率，并且降低了 I/O 成本和内存的使用率。张婷曼等[15]针对汽车流量时空分布不均引发交通拥堵的问题，提出一种基于不确定性关联规则的城市交通数据挖掘方法，考虑时空的相关性，提高了交通管理的效率。毛伊敏等[16]改进的并行关联规则增量挖掘算法，在大数据环境下具有相对较好的性能表现。

国外相关学者也就关联规则算法存在的上述问题进行改进。Le 等[17]针对产生的大量规则，提出了一种基于动物迁移优化的挖掘算法，减少了生成的规则数量。Krishna 等[18]针对冗余关联规则，提出了一种基于自适应二微分进化的非冗余关联规则算法，提高了算法的速率。Chen 等[19]结合 reason 模型和经典关联规则算法设计了多维关联规则，分析了各因素之间的相关性。Li 等[20]挖掘频繁项集确定各因素之间的关联规则，探讨了瓦斯爆炸事故的潜在规律特征。Florian 等[21]使用关联规则算法提出了一种数据驱动的故障风险评估方法，并在很多领域进行了应用。国外学者在研究关联规则算法的改进时，注重提高算法运行效率和算法的应用性，通常将关联规则算法应用在某个实际问题的解决中，但没有考虑提取关联规则时产生的频繁项集会占用大量内存并且会产生冗余的问题，因此论文针对这些缺点进行了改进。

国内外学者对关联规则算法的缺点进行了改进，但其不足之处在于大多数的方法只考虑了其中某一个问题，没有将两个方面结合起来，对其做最大程度的整合式改进。因此，本文从 Apriori 算法占用内存大和运行效率低两个方面着手，对其同时改进，提出了基于随机抽样和哈希(Hash)桶的 RS_Hash 算法，既能减少内存占用，又能提高算法的运行效率。

## 1 RS_Hash 算法

### 1.1 算法思想

为了提高算法的运行效率，考虑在保证算法准确率的前提上，减少其运行时间。在扫描数据库时，使用动态随机抽样的方法从数据中抽取样本，并根据实验确定其最优样本容量。在挖掘频繁项集时，抽取出的最优样本容量与整体数据库的支持度会有一定的误差，为了使得误差达到最小，设计并实现了两种随机抽样方法，确定最终抽取的样本。

针对 Apriori 算法生成大量频繁项集占用内存的问题，基于哈希算法的思想，通过哈希函数将频繁项集映射入哈希桶中，不同于传统哈希表的存储，在哈希桶中删去原有的频繁项集，只将项对的计数存放入桶中，哈希桶对应的整数值增加，能够减少内存的占用。

### 1.2 算法设计

设计动态随机抽样方法。扫描数据库 $D$ 并遍历所有的事务，得到数据库 $D=\{T_1,T_2,\cdots,T_n\}$，$T_i$ 表示每项事务集合，以及项目集 $I=\{i_1,i_2,\cdots,i_n\}$。基于样本近似代替总体的思想，对事务数据库使用随机抽样算法，计算样本与总体的误差。根据样本数据集，生成频繁 1 项集。每一个事务 $T_i(i=1,2,\cdots,n)$ 满足 $T_i\in I$，而所有的事务 $T_i$ 都分别是项目集合 $I$ 的一个子集；

构建 Hash 函数，挖掘频繁项集。扫描样本数据，在第 $k$ 次扫描时，每项事务产生自身的 $k+1$ 项集，将生成的 $k+1$ 项集使用 Hash 函数映射得到哈希值。根据其哈希值放入到该哈希值对应的桶中，并对桶中元素个数进行计数。若桶中元素个数小于最小支持度，则该哈希值所对应桶中元素一定不是频繁项集，将其从候选项集中删除。根据频繁项集，挖掘关联规则，找出大于最小置信度的强关联规则。

1)动态随机抽样算法

为减少寻找频繁项集所用的时间，首先使用动态抽样的方法，从小到大不断地从总体中抽出样本，直至获得最

优样本，代替总体进行关联分析。本文设计了两种抽样方法，将对比两种方法的适用性，选择最合适的样本参与实验。

(1)抽样误差 $\sigma$。

在进行随机抽样时，当样本容量不断增加，将所有频繁项集在样本上的支持度和它在总体上的支持度之差的均值定义为抽样误差 $\sigma$，当这个值不再显著变化时，即满足某个阈值时，便找到了可以代替总体进行分析的最优样本。

遍历数据库，得到所有的事务，随机抽取样本之后，计算样本中频繁项集与总体频繁项集的抽样误差。错误度量值的计算公式为：

$$Error = \frac{\sum (Sup\_count(D) - Sup\_count(sample))}{len(D)} \tag{1}$$

式中：$Sup\_count(sample)$ 为频繁项集在样本上的支持度；$Sup\_count(D)$ 为频繁项集在整体上的支持度；$len(D)$ 为数据库的长度。抽样误差为相邻抽取样本后错误度量值的变化为 $\sigma = \Delta Error$，当 $\sigma$ 小于某个阈值不再显著变化时，便找到了代替总体分析的最优样本。

(2)抽样停止阈值 $F(S_i)$

设 $S_i$ 为抽样序列，$i = 1,2,\cdots$，通过随机抽样从总体 $D$ 中抽取样本，相应的频繁性会产生 4 种情况：①在总体和样本中都频繁记为 $TP(S_i)$；② 在总体和样本中都不频繁记为 $TN(S_i)$；③ 在总体中频繁但在样本中不频繁记为 $FN(S_i)$；④ 在总体中不频繁但在样本中频繁记为 $FP(S_i)$。 由此设计一个抽样停止规则如式(2)所示。

$$F(S_i) = \frac{TP(S_i) + TN(S_i)}{X_i} - \frac{FP(S_i) + FN(S_i)}{X_i} \tag{2}$$

式中：$X_i$ 为每个 $S_i$ 中产生的 1 项集个数。为使抽样产生的误差最小，则该样本中的所有 1-项集都应与总体中 1-项集的频繁性相同，因此，当 $FP(S_i)$ 和 $FN(S_i)$ 都为 0，即 $F(S_i)$ 接近于 1 时，可以找到最优的样本。

由于方法(1)中的抽样误差需要设定阈值来判断样本的准确性，但阈值的设置容易受主观因素的影响，而方法(2)会出现因抽样样本过小导致 $F(S_i)$ 存在负值的问题，因此论文采用两种随机抽样算法共同评估样本的大小，选取最合适的样本容量。

2)哈希算法

由于关联规则算法会生成大量的频繁项集，加剧了存储的负担，因此，为减少存储频繁项集所用的空间，通过哈希算法将生成的候选项集映射到哈希表，统计哈希表中的数值来计算频繁项集的大小。将相同 Value 值放入同样的桶中，搜索时只需要其对应的 key 索引。本文在哈希桶中删去原有的频繁项集，只将项对的计数存放入桶中，哈希桶对应的整数值增加，减少了内存的占用。

哈希算法的计算过程如下。

频繁 1-项集通过自连接形成候选 2-项集，将候选 2-项集通过哈希函数映射到 Hash 表中，定义哈希函数如式(3)所示。

$$h(n_1, n_2) = (n_1 \hat{} n_2) \bmod (buckets) \tag{3}$$

式中：$n_1$ 为候选 2-项集中的首元素；$n_2$ 为第 2 位元素，$buckets$ 为哈希桶的个数。

统计 Hash 到桶中的项对个数，找出大于最小支持度(min_Support)的项对计数，则对应的桶为频繁桶。支持度计算方式如式(4)所示。

$$Support(X \rightarrow Y) = p(Y/X) = \frac{Sup\_count(X \cup Y)}{Sup\_count(D)} \tag{4}$$

当支持度大于最小支持度，即满足式(5)时，对应的桶为频繁桶，否则为非频繁桶。

$$Support(X \rightarrow Y) \geqslant \min\_Support \tag{5}$$

将哈希表压缩为 bitmap，每一个 bit 用作一个桶，如果哈希桶为频繁桶，则对应位置为 1，否则置 0，使用 0－1 值来代替事务数据。哈希桶内部存放整数值，项对被哈希到这些桶中，通过多步循环，创造所有的项对。

## 2 实验过程及结果分析

### 2.1 实验环境和数据集

实验基于 Windows 操作系统，实验环境配置如下：CPU 型号 R5 3550H；内存容量为 16 GB；编程语言 Python-3.6；编程平台 PyCharm-2020.2(Professional Edition)；集成环境管理 Anaconda Navigator-1.3.1。

实验使用大小不同的 4 个数据集，如表 1 所示，数据集 1～3 来源于 Kaggle 竞赛公开的商品数据集，数据集 4 来自某航天院提供的航天器平台有效载荷数据。数据集 1 为商品 ID 数据，数据大小为 6 000；数据集 2 为商品分类数据，大小为 100 000；数据集 3 为商品销售数据，大小为 208 084。数据集 4 为有效载荷数据，大小为 726 420，其中有效载荷工程参数数据由航天院传感器采集，共计 726 420 条数据，160 条指令。

**表 1 数据集**

| 数据集 | 事务数 | 事务最长长度 |
|---|---|---|
| 1 商品 ID 数据 | 6 000 | 28 |
| 2 商品分类数据 | 100 000 | 45 |
| 3 商品销售数据 | 208 084 | 127 |
| 4 有效载荷数据 | 726 420 | 160 |

### 2.2 实验结果分析

实验采用 4 个大小不同的数据集，使用 Python 语言对 Apriori 算法[5]、PCY 算法[4]、SON 算法[22]、FP-Growth 算法[10]、RCM_Apriori 算法[23]、Hash_Cumulate 算法[24]以及本文提出的 RS_Hash 算法进行编程实现，将产生频

繁项集的时间和占用内存的大小作为指标来衡量改进算法的效果。在不同样本长度、不同支持度的情况下验证论文改进算法的有效性。

1)动态随机抽样实验

使用随机抽样法对原始数据集进行抽样，以样本代替总体的思想，从数据集中抽取部分样本参与实验，既可以避免重复利用候选项集，又能提高算法的运行效率。

(1)抽样误差 $\sigma$

为了找到最合适的样本大小，假设当抽样误差阈值时 $\sigma<10$ 停止抽取。针对随机抽样的具体样本容量的选择，对每个数据集设计了实验。随着样本数目增多，各个数据集上抽样误差的变化情况如表 2 所示。

**表 2　样本大小和抽样误差**

| 数据集 | | 样本大小及误差 | | | | |
|---|---|---|---|---|---|---|
| 1 | 大小 | 2 000 | 2 500 | 3 000 | 3 500 | $\sigma$ |
| | Error | 73.05 | 60.77 | 45 | 43.26 | 1.74 |
| 2 | 大小 | 35 000 | 40 000 | 50 000 | 60 000 | $\sigma$ |
| | Error | 159.58 | 137.28 | 82.09 | 76.52 | 5.57 |
| 3 | 大小 | 40 000 | 50 000 | 60 000 | 70 000 | $\sigma$ |
| | Error | 53.71 | 41.97 | 27.35 | 23.73 | 3.62 |
| 4 | 大小 | 200 000 | 300 000 | 350 000 | 450 000 | $\sigma$ |
| | Error | 364.87 | 163.86 | 53.82 | 46.35 | 7.74 |

由表 2 可以看出，对数据集 1 进行随机抽样，当样本数量为 3 000 时，与样本 3 500 的随机抽样误差 $\sigma$ 仅为 1.74，误差趋于稳定，不再显著变化，因此，在数据集 1 上，拟选用最优样本数为 3 000；以此类推，在数据集 2 上 50 000 是可供选择的最佳样本数；同样地，在数据集 3 上样本数量为 60 000 时，抽样误差 $\sigma$ 为 3.62，因此，在数据集 3 上，拟选用的最优样本数为 60 000；数据集 4 上选用 450 000 为最优样本数量。

(2)抽样停止规则 $F(S_i)$

使用抽样停止规则 $F(S_i)$ 分别在 4 个数据集上进行实验，随机抽取样本的学习曲线如图 1 所示。

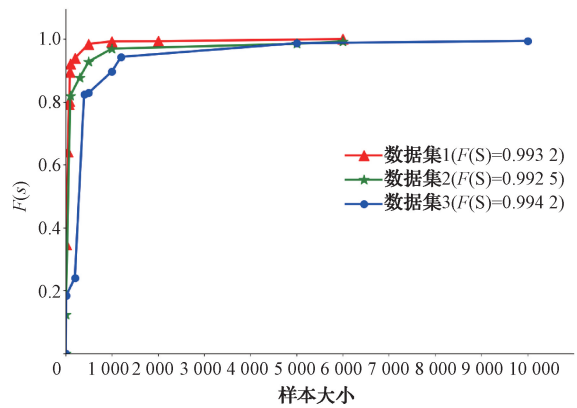



图 1　3 个公开数据集上不同样本的随机抽样学习曲线

各个数据集上 $F(S_i)$ 随样本容量的变化情况(此处只给出了特殊点样本)如表 3 所示。

**表 3　各数据集的部分样本大小和 $F(S_i)$**

| 数据集 | | 样本大小和 $F(S_i)$ | | | | |
|---|---|---|---|---|---|---|
| 1 | 大小 | 100 | 200 | 500 | 1 000 | 2 000 |
| | $F(S_i)$ | 0.922 0 | 0.939 3 | 0.984 7 | 0.992 8 | 0.993 2 |
| 2 | 大小 | 100 | 500 | 1 000 | 5 000 | 6 000 |
| | $F(S_i)$ | 0.819 3 | 0.928 5 | 0.969 6 | 0.984 9 | 0.992 5 |
| 3 | 大小 | 1 000 | 1 200 | 5 000 | 6 000 | 50 000 |
| | $F(S_i)$ | 0.896 5 | 0.943 4 | 0.986 9 | 0.994 2 | 0.999 |
| 4 | 大小 | 100 000 | 200 000 | 300 000 | 350 000 | 380 000 |
| | $F(S_i)$ | 0.632 3 | 0.732 9 | 0.893 2 | 0.964 5 | 0.997 5 |

从表 3 可知，在数据集 1 上，样本容量在[0，100]时，$F(S_i)$ 值从 0 急速上升到 0.922 0，说明由这个样本产生的关联规则与总体产生的关联规则的一致率约为 92%；当样本容量在[500，2 000]时，$F(S_i)$ 达到 0.993 2，即当选择样本为 2 000 时，准确率可高达 99.32%。理论上，样本容量越大，$F(S_i)$ 越近似的接近 1，为了提高算法效率的同时，最大程度的减少准确率的损失，可以选择 2 000 作为最优的样本容量值；数据集 2 上选择样本为 6 000 时，准确率可高达 99.25%，因此可以选择 6 000 为最优样本容量值；同样地，在数据集 3 上，在样本容量在[1 200，50 000]时，$F(S_i)$ 高达 0.999 0，即当选择样本为 50 000 时，准确率可高达 99.90%，因此，选择 50 000 为最优样本容量值；数据集 4 上选择 380 000 为最优样本。

综上所述，方法(1)和(2)在不同数据集上选择的样本数以及最终选择的样本数如表 4 所示。

**表 4　最终选择的样本大小**

| 样本容量 | 数据集 1 | 数据集 2 | 数据集 3 | 数据集 4 |
|---|---|---|---|---|
| 方法(1) | 3 000 | 50 000 | 60 000 | 450 000 |
| 方法(2) | 2 000 | 6 000 | 50 000 | 380 000 |
| 最终样本 | 2 500 | 50 000 | 55 000 | 415 000 |

为了将误差降到最低，本文使用两个方法选取样本的平均数作为最优样本进行实验，值得注意的是，在数据集 2 上，两种方法得到的样本数量差异巨大，由于在方法(2)中样本越大，$F(S_i)$ 值越接近 1，因此为保证最大程度的降低错误率，选择较大的样本数 50 000 进行实验。

2)RS_Hash 算法实验

(1)运行时间对比

在保证规则准确率的前提下，对样本进行随机抽取后使用最优样本容量进行实验。为了体现算法的有效性，分别对 Apriori 算法、PCY 算法、SON 算法、FP-Growth 算法、RCM_Apriori 算法、Hash_Cumulate 算法和所提出的 RS_Hash 算法在不同的数据集上进行对比实验，具体运行时间如表 5、6 所示。其中，加粗字体表示所用的最短时间。

**表 5　不同支持度下 4 种算法在数据集 1、2 上的运行时间**

| 算法 | 数据集 1 不同支持度运行时间 | | | | 数据集 2 不同支持度运行时间 | | | |
|---|---|---|---|---|---|---|---|---|
| | 0.001 | 0.002 | 0.003 | 0.004 | 0.002 | 0.003 | 0.004 | 0.005 |
| Apriori[5] | 35.108 4 | 35.399 6 | 35.639 3 | 35.579 8 | 473.864 | 505.509 | 524.000 | 514.298 |
| PCY[4] | 34.367 6 | 34.730 2 | 34.942 5 | 34.798 4 | 417.044 | 425.401 | 424.455 | 412.277 |
| SON[22] | 18.847 1 | 18.184 1 | 17.206 6 | 17.115 2 | 367.003 | 361.823 | 373.176 | 366.796 |
| FP-Growth[10] | 17.117 5 | 16.941 0 | 15.794 2 | 13.099 6 | 353.546 | 360.487 | 351.694 | 335.723 |
| RCM_Apriori[23] | 14.274 2 | 15.237 6 | 14.832 0 | 13.574 3 | 310.823 | 315.832 | 311.527 | 305.315 |
| Hash_Cumulate[24] | 16.764 3 | 16.520 4 | 15.964 2 | 13.654 0 | 316.264 | 314.653 | 310.637 | 300.075 |
| **RS_Hash** | **10.474 8** | **11.763 6** | **10.901 4** | **10.470 5** | **292.834** | **268.200** | **280.680** | **227.381** |

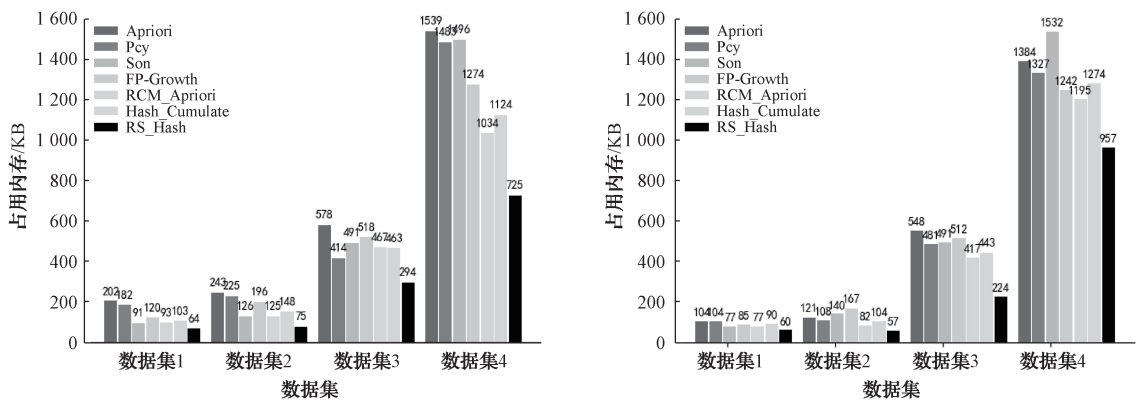
**表 6　不同支持度下 4 种算法在数据集 3、4 上的运行时间**

| 算法 | 数据集 3 不同支持度运行时间 | | | | 数据集 4 不同支持度运行时间 | | | |
|---|---|---|---|---|---|---|---|---|
| | 0.004 | 0.005 | 0.006 | 0.007 | 0.1 | 0.2 | 0.3 | 0.4 |
| Apriori[5] | 2 437.90 | 1 845.48 | 1 647.18 | 1 535.63 | 5 386.64 | 5 125.47 | 4 973.65 | 4 779.27 |
| PCY[4] | 650.683 | 619.388 | 592.165 | 511.352 | 2 547.82 | 2 765.86 | 2 381.60 | 2 006.34 |
| SON[22] | 1 189.59 | 1 138.48 | 729.252 | 653.596 | 3 718.53 | 3 485.84 | 2 733.17 | 2 366.96 |
| FP-Growth[10] | 825.945 | 817.927 | 771.825 | 721.249 | 2 893.47 | 2 556.86 | 2 338.90 | 2 106.68 |
| RCM_Apriori[23] | 624.831 | 585.826 | 592.416 | 613.781 | 2 294.67 | 2 045.83 | 1 984.85 | 1 872.22 |
| Hash_Cumulate[24] | 650.615 | 639.816 | 645.583 | 520.419 | 2 165.72 | 2 084.33 | 1 926.76 | 1 892.58 |
| **RS_Hash** | **452.298** | **451.179** | **439.914** | **433.880** | **1 398.45** | **1 233.92** | **1 216.85** | **1 063.27** |

由表 5、6 可以看出，在数据集 1 上，相较于 Apriori 算法、PCY 算法、SON 算法、FP-Growth 算法、RCM_Apriori 算法和 Hash_Cumulate 算法，RS_Hash 算法在平均时间效率上分别提高了 69.23%、68.59%、38.79%、30.10%、24.70%和 30.33%；在数据集 2 上，RS_Hash 算法在平均时间效率上分别提高了 46.84%、36.36%、23.31%、23.81%、14.07%和 14.01%；在数据集 3 上，RS_Hash 算法在不同支持度的情况下，所用的时间均比其他算法少，在平均时间效率上分别提高了 75.51%、24.63%、48.91%、43.23%、26.41%和 27.11%；在数据集 4 上，相较于其他 6 种算法，RS_Hash 算法在平均时间效率上分别提高了 75.81%、49.10%、59.38%、50.22%、40.16%和 39.22%。

(2)占用内存对比

7 种算法在 4 个数据集上挖掘频繁 1 项集时占用的内存如图 2(a)所示，频繁 2 项集占用内存如图 2(b)所示。随着事务数的增多，各算法占用的内存逐渐增大。由图 2 可知，在挖掘频繁项集的过程中，RS_Hash 算法占用的内存最小。




(a) 4个数据集上频发1项集占用内存对比　　(b) 4个数据集上频发1项集占用内存对比

图 2　4 个数据集上不同算法占用内存对比

(3)准确率对比

在相同的支持度阈值下比较了 RS_Hash 算法在卫星载荷数据集上不同样本数下挖掘出的关联规则总数，结果如表 7 所示。从表 7 可以看出，当样本数为 41 500 时，产

生的规则总数为13 925，与样本数为45 000时产生的规则总数非常接近，因此结果表明，使用样本代替总体的方式进行关联规则的提取时，通过抽样引起的关联分析的误差很小，用最优样本进行关联挖掘是可靠的。

**表7 RS_Hash算法在不同样本数的情况下产生的规则数**

| 样本数 | 规则数 |
|---|---|
| 10 000 | 6 264 |
| 20 000 | 8 527 |
| 30 000 | 11 364 |
| 41 500 | 13 925 |
| 45 000 | 13 936 |

由于海量的数据在挖掘过程中会产生大量冗余和相似的关联规则，而RS_Hash算法在进行频繁项集的挖掘过程中经过对项集频繁性的筛选，将非频繁的项集删去，因此在后续提取的规则中，减少了冗余。卫星载荷数据提取的规则需要前后项具有较高的关联性，弱相关的规则在实际应用中使用的价值不高，因此本实验使用置信度评价改进算法提取关联规则的准确率。

置信度表示关联规则前项出现时后项同时出现的概率，其表达公式如式(6)所示。

$$Confidence(X \rightarrow Y) = p(Y/X) = \frac{p(X,Y)}{p(X)} \tag{6}$$

将满足最小置信度的规则称为强关联规则，为了证明RS_Hash算法能够有效去除冗余，实验设定了不同的置信度阈值，对比在不同置信度下各算法提取的强关联规则总数。由于PCY、SON和FP-Growth算法在Apriori算法的基础上提升了算法的效率，并未去除冗余的规则，提取的强关联规则与Apriori算法相同，因此本实验对比了RS_Hash算法与Apriori、RCM_Apriori和Hash_Cumulate算法在不同置信度下产生的强关联规则总数，实验结果如图3所示。

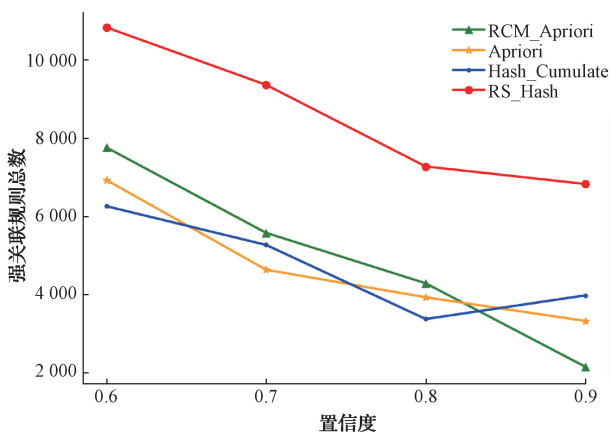



图3 不同置信度下强关联规则总数对比

实验证明，RS_Hash算法在不同置信度值下，产生的强关联规则数目多于其他算法，即挖掘出的关联规则前后项之间具有较高的相互关联性，规则更加可靠，提取的关联规则能够更加精准地满足实际的需要，提高了算法在应用中的准确率。因此，RS_Hash算法能迅速找到更有用的关联规则，有助于更好的对规则进行理解和分析。

## 3 结 论

本文通过对关联规则算法的原理进行分析，研究产生频繁项集的过程，针对Apriori算法的不足之处，从时间和空间两方面着手对其进行改进，提出了基于RS_Hash频繁项集的卫星载荷关联规则算法。对于时间复杂度，RS_Hash算法基于样本代替总体的思想，通过设置抽样误差阈值和停止抽样的规则，在保证了规则准确度的前提下，使用随机抽样的方式在数据集中抽取最优样本容量进行实验，在一定程度上减少了计算的时间。对于空间复杂度，在随机抽样方法抽取的样中结合哈希函数散列法将频繁项集映射入哈希桶中，对哈希桶进行计数，统计哈希桶的计数，频繁桶记为1，非频繁记为0，将事务数据库中的数值用0—1代替，将大量的数据转化为整数数值，所占用的空间远小于对原始数据的存储。通过实验验证，RS_Hash算法在对大数据的处理上有着足够的优势，在大数据时代非常适用。因此论文中提出的RS_Hash算法不仅减少了占用的内存，而且提高了算法的运行效率。实验选用的公共数据集与卫星载荷数据集具有相同的特性如数量大、维度高，因此，基于RS_Hash频繁项集的卫星载荷关联规则算法同样适用于商品选购、市场篮子分析和用户推荐系统等多个领域。

## 参 考 文 献

[1] 邹克旭，伊大伟，房红征. 卫星相关性分析方法及体系研究[C]. 第二十届全国测试与故障诊断技术研讨会，2011：420-424.

[2] 孙宇豪，李国通，张鸽. 一种基于相关概率模型的卫星异常检测方法[J]. 中国科学院大学学报，2021，38(3)：409-416.

[3] 杨甲森，孟新，陈托，等. 基于遥测数据相关性的航天器异常检测[J]. 仪器仪表学报，2018，39(8)：24-33.

[4] PARK J，CHEN M，YU P. An effective hash based algorithm for mining association rules[J]. Proceeding of the ACM SIGMOD International Conference on Management of Data. Newyork：ACM 1995：175-186.

[5] AGRAWAL R，IMIELINSKI T，SWAMI A. Mining association rules between sets of items in large databases [C]. Proceedings of ACM SIGMOD International Conference on Management of Data. Newyork：ACM Press，1993：207-216.

[6] 毕玉萍，胡世昌，李劲华. 基于排序树的Node-Apriori

改进算法[J]. 青岛大学学报(自然科学版),2020,33(3):50-56.
[7] 胡世昌. 基于二进制编码的 Apriori 改进算法[J]. 计算机应用研究,2020,37(2):398-400,423.
[8] 王伟,储泽楠,韩毅. 基于 MapReduce 的 Apriori 前后项约束关联规则改进算法[J]. 信阳师范学院学报(自然科学版),2020,33(3):448-453.
[9] 程江洲,闫冉阳,冯梦婷, 等. 基于 ACT-Apriori 算法的电网故障诊断方法研究[J]. 电子测量技术,2021,44(24):32-39.
[10] HAN J, PEI J, YIN Y, et al. Mining frequent patterns without candidate generation: A frequent-pattern tree approach[J]. Data Mining & Knowledge Discovery,2004,8(1):53-87.
[11] LI L, LI Q, WU Y, et al. Mining association rules based on deep pruning strategies [J]. Wireless Personal Communications, 2017(2):1-25.
[12] 李涛,郁美辰,陆正邦, 等. 基于关联规则挖掘的气象观测设备一致性检测算法[J]. 电子测量与仪器学报,2017,31(10):1568-1573.
[13] 叶峰. 基于二分法的改进 Apriori 关联算法研究[J]. 电子设计工程,2020,28(16):49-53.
[14] 刘彦戎,杨云. 一种矩阵和排序索引关联规则数据挖掘算法[J]. 计算机技术与发展. 2021,31(2):54-59.
[15] 张婷曼,丁凰. 依赖不确定性关联规则的城市交通流大数据挖掘[J]. 国外电子测量技术,2020,39(11):39-45.
[16] 毛伊敏,邓千虎,邓小鸿, 等. 改进的并行关联规则增量挖掘算法[J]. 计算机应用研究,2021,38(10):2974-2980.
[17] LE H, CHICLANA F. ARM-AMO: An efficient association rule mining algorithm based on animal migration optimization[J]. Knowledge-Based Systems, 2018,154:68-80.
[18] KRISHNA G J, RAVI V. Mining top high utility association rules using binary differential evolution[J]. Engineering Applications of Artificial Intelligence, 2020, 96: 103935.
[19] CHEN D, PEI Y, XIA Q. Research on human factors cause chain of ship accidents based on multidimensional association rules[J]. Ocean Engineering, 2020, 218(20):107717.
[20] LI L, GUO H, CHENG L, et al. Research on causes of coal mine gas explosion accidents based on association rule[J]. Journal of Loss Prevention in the Process Industries, 2022,80: 104879.
[21] FLORIAN P, KINS R, FREDERIK C, et al. Interpretable failure risk assessment for continuous production processes based on association rule mining[J]. Advances in Industrial and Manufacturing Engineering,2022(5): 100095.
[22] ANON. Efficient association rule mining based son algorithm for a bigdata platform [J]. Journal of Digital Contents Society, 2017, 18(8): 1593-1601.
[23] 廖纪勇,吴晟,刘爱莲. 基于布尔矩阵约简的 Apriori 算法改进研究[J]. 计算机工程与科学,2019,41(12):2231-2238.
[24] 郭倩,殷丽凤. 基于散列技术的多层关联规则算法的改进[J]. 计算机工程与设计,2021,42(9):2485-2491.

## 作 者 简 介

贾澎涛,博士,教授,主要研究方向为机器学习、人工智能、智慧矿山等。
E-mail:jiapengtao@xust.edu.cn

温滋,硕士研究生,主要研究方向为深度学习,数据挖掘。
E-mail:1518478860@qq.com