文章编号：1005-8451（2017）12-0001-04

一种基于改进FP-Growth算法的动车组故障预测研究

张　春，郭玉霞
（北京交通大学　计算机与信息技术学院，北京　100044）

摘　要：动车组的故障预测和健康管理是目前的研究热点，其中，故障预测的关键是寻找动车组故障信息和状态信息之间的关联关系。频繁模式增长（FP-Growth）算法是关联规则挖掘中的经典算法之一，用来挖掘频繁项集。针对动车组故障数据提出了一种改进的FP-Growth（IFP-Growth，Improved FP-Growth）算法，采用先序遍历FP-tree的方法产生条件模式基。实验结果表明，IFP-Growth算法能够有效提高动车组故障数据挖掘的效率，并且能够有效地挖掘动车组故障信息和状态信息之间的关联关系。

关键词：关联规则；FP-Growth算法；动车组；故障预测
中图分类号：U266.2：TP39　**文献标识码：**A

Fault prediction of EMU based on improved FP-Growth algorithm

ZHANG Chun, GUO Yuxia
(School of Computer and Information Technology, Beijing Jiaotong University, Beijing　100044, China)

Abstract: Prognostics and Health Management(PHM) of EMU is the hotspot of current research. The key of fault prediction is to find the relation between fault information and status information of EMU. The FP-Growth algorithm is one of the classical algorithms in association rule mining. It is used to excavate frequent item sets. This paper proposed an improved FP-Growth (IFP-Growth) algorithm for EMU fault data. It adopted pre-traversing FP-tree to generate conditional pattern bases. The experimental results showed that the IFP-Growth algorithm could effectively improve the efficiency of data mining of EMU faults and find the relation between fault information and status information of EMU.

Keywords: association rule; FP-Growth algorithm; EMU; fault prediction

动车组检修是保障列车安全运行的重要途径，要对维修策略进行优化就需要准确地把握动车组关键零部件的健康状态及其故障发生规律。故障预测与健康管理（PHM，Prognostics and Health Management）技术可实现动车组的故障预测和健康状态评估，并根据预测和评估结果给出维修决策，提高动车组的可用率，使维修方式更加具有主动性[1]。如今，随着先进的数据采集和计算机存储技术的飞速发展，产生和积累了大量的高速动车故障数据，如何利用这些宝贵的数据并从中挖掘出有效的知识，从而为动车组的故障预测及维修工作提供决策支持，是一个亟待解决的问题。

收稿日期：2017-07-28
基金项目：国家“863”计划项目（2015AA043701）；中国铁路总公司科技开发计划重点课题（2015J006-C）。
作者简介：张　春，高级工程师；郭玉霞，在读硕士研究生。

数据挖掘是从大量、有噪声、不完全和随机的实际应用数据中，提取隐含在其中不为人知的但却具有潜在利用价值的信息的过程[2]。作为数据挖掘的主要方法，关联规则可以发现同一事务中出现的不同项之间的相关性，即找出事务中频繁发生项的所有子集以及不同项之间的相互关联性。通过关联规则建立故障与状态信息之间的联系，对故障进行分析和预测是一种重要的故障检测手段[3]。本文针对动车的故障和运行状态进行关联关系挖掘，并用于动车组的故障预测。

1　FP-Growth算法概述

1.1　关联规则

关联规则挖掘是数据挖掘领域中最活跃的研究方法之一[4]，用来发现事务之间的联系。设

$I=\{a_1,a_2,\ldots,a_m\}$ 为项的集合，其中，a_1，a_2，…，a_m 为 m 个不同的项，包含 0 个或多个项的集合称为项集。$D=\{T_1,T_2,\ldots,T_n\}$ 是一个事务数据库，每个事务 T_i（$i \in \{1,2,\cdots,n\}$）是 I 的一个子集。每个事务有一个标识符，称为 TID。如果 I 的一个子集 A 满足 $A \subseteq T$，则称事务 T 包含项集 A。一个关联规则是形如 $A \Rightarrow B$ 的蕴含式，其中，$A \subseteq I$，$B \subseteq I$ 且 $A \cap B=\Phi$。

项集的支持度是包含此项集的事务个数，支持度大于用户指定的最小支持度的项集称为频繁项集或者频繁模式。长度为 k 的频繁项集称为频繁 k- 项集。规则 $A \Rightarrow B$ 的支持度是数据库中同时包含 A 和 B 的事务数目与所有事务数目之比，记为 support($A \Rightarrow B$)。规则 $A \Rightarrow B$ 的置信度是指同时包含 A 和 B 的事务数目与包含 A 的事务数目之比，记为 confidence($A \Rightarrow B$)。支持度和置信度是描述关联规则的两个重要概念。支持度用于衡量关联规则在整个事务数据集中的统计重要性，置信度用于衡量关联规则的可信程度。

1.2 FP-Growth算法

在关联规则分析中，频繁项集的挖掘最常用到的是 Apriori 算法。Apriori 算法是一种先产生候选项集再检验是否频繁的“产生 - 测试”的方法，该算法的缺点是为得到较长的频繁模式需要生成大量候选短频繁模式，而且当数据集很大的时候，需要不断扫描数据集造成运行效率很低 [5]。Jiawei Han[6] 等人提出的 FP-Growth 算法克服了 Apriori 算法的缺点，它不产生候选项集。

FP-Growth 算法采用存储有频繁模式相关信息的称为频繁模式树（FP-tree，Frequent Pattern tree）的树形数据结构，由频繁 1- 项集（又叫频繁项头表）、标记为 null 的根结点以及项前缀子树 3 部分组成。项前缀子树中的每个结点包括 3 个域 [7]：item_name，count 和 node_link。item_name 记录该结点所代表的项的名字；count 记录所在路径代表的事务中达到此结点的事务个数；node_link 指向下一个具有相同 item_name 的结点，如果没有这样一个结点，则其值被置为 null。频繁项头表包含 2 个域：item-name 和 head of node-link。head of node link 指向 FP-tree 中具有相同 item-name 的第一个结点。项头表的构建会直接影响到 FP-Growth 算法的性能。

FP-Growth 算法采用分而治之的思想 [5]，通过扫描两次事务数据库，将数据库中频繁项信息全部压缩在 FP-tree 中，对事务数据库的挖掘转化为对 FP-tree 的挖掘。对于大规模数据集，FP-Growth 算法在构建 FP-tree 时和搜索 FP-tree 过程中存在内存和计算的双重瓶颈 [8]。针对 FP-Growth 算法的缺点，文献 [9] 和文献 [11] 提出了基于 MapReduce 的 FP-Growth 算法的并行化改进算法，文献 [10] 提出了基于 Spark 框架的 SPFP 算法。

2 FP-Growth算法改进

动车组故障事务数据库中包含动车组故障信息和运行状态信息。故障信息包括牵引电机温度高、轴承磨损、牵引变流器、空调及通风系统等类型的故障。运行状态信息包括列车编号、故障时间、经度、纬度、列车速度、环境温度等。选取故障数据库中的 6 个事务及每个事务中的项如表 1 前两列所示，其中，F_t（t=1,2）表示故障信息，I_k（k=1,2,3,4,5,6）表示运行状态信息。设最小支持度计数为 2，根据 FP-Growth 算法步骤，先产生频繁 1- 项集，由于 I_6 的支持度计数 < 2，所以频繁 1- 项集中没有 I_6。按照频繁 1- 项集顺序对事务中的项进行降序排序，动车组故障数据事务，如表 1 所示；构建 FP-tree，如图 1 所示。

表1 动车组故障数据事务

TID	项	排序后的项
T_1	F_1,I_1,I_5,I_6	F_1,I_1,I_5
T_2	F_1,I_1,I_3,I_4	F_1,I_1,I_3,I_4
T_3	F_1,I_1,I_2,I_5	F_1,I_2,I_1,I_5
T_4	F_1,I_2,I_4	F_1,I_2,I_4
T_5	F_2,I_2,I_3	I_2,F_2,I_3
T_6	F_2,I_2,I_5	I_2,I_5,F_2

在产生条件模式基时，FP-Growth 算法是自下而上产生频繁 1- 项集的条件模式基，需要重复扫描公共路径。当构建的 FP-tree 非常庞大时，会消耗很大时间和空间。

经分析，公共路径重复的原因有：故障数据事务的压缩程度；频繁 1- 项集的长度。较大的事务压缩程度和较长的频繁 1- 项集都会使公共路径重复的可能性增大。为了避免重复搜索公共路径，减小时

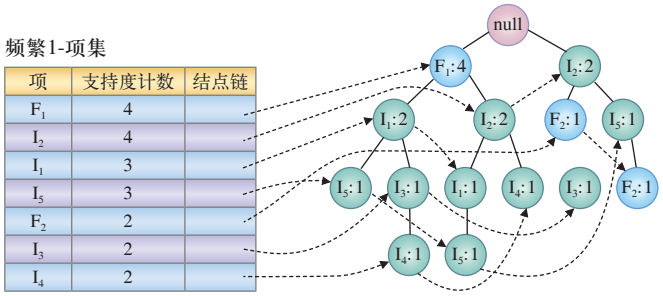

图1　根据FP-Growth算法构建的FP-tree

间复杂度和空间复杂度，改进 FP-Growth 算法，采用先序遍历 FP-tree 的思想，在一次遍历 FP-tree 后就能逐步取得频繁 1- 项集的全部条件模式基。

改进的 FP-Growth（IFP-Growth，Improved FP-Growth）算法产生条件模式基来挖掘 FP-tree 的过程如下。

（1）初始化共享路径 sharedPath 为空集，访问结点 F_1，此时 sharedPath 存放的是 F_1 的前缀路径，因为 sharedPath 现在为空，所以 F_1 的一个条件模式基是 null。

（2）将 F1 加入到 sharedPath，然后访问 I_1，此时 sharedPath 存放的是 I_1 的前缀路径，所以 F_1 是 I_1 的一个条件模式基，支持度是 2，记为 F_1:2。

（3）将 I_1 加入到 sharedPath，更新 sharedPath 内容为 F_1I_1。接着访问 I_5，此时 sharedPath 存放的是 I_5 的前缀路径，所以 F_1，I_1 是 I_5 的条件模式基，支持度是 1，记为 F_1，I_1:1。

（4）将 I_5 加入到 sharedPath，更新 sharedPath 内容为 $F_1I_1I_5$。由于 I_5 是叶结点，所以回退到最近的一个分支结点 I_1，同时 sharedPath 内容变为 F_1I_1。

（5）继续访问 I_1 的另外一个孩子结点 I_3，得到 I_3 的一个条件模式基 F_1，I_1:1，同时更新 sharedPath 内容为 $F_1I_1I_3$，然后访问 I_4，得到 I_4 的一个条件模式基，F_1，I_1，I_3:1。

（6）I_4 又是一个叶结点，回退到最近的且未被遍历的分支结点 F_4，用同样的方法访问其他的子结点，得到条件模式基。

整个搜索过程不需要项头表，相同方向的指针只需用 sharedPath 存储公共路径，最终可以根据条件模式基得到所有的频繁模式，如表 2 所示。

IFP-Growth 算法的搜索过程是先序遍历树的过程，只需对 FP-tree 所有结点遍历一次就能获得所有频繁 1- 项集的条件模式基，时间复杂度和空间复杂度均为 O(n)，n 为 FP-tree 的结点个数。

表2　通过条件模式基挖掘FP-tree

项	条件模式基	条件FP树	产生的频繁模式
I_4	{{F_1,I_1,I_3:1},{F_1,I_2:1}}	<F_1:2>	{F_1,I_4:2}
I_5	{{F_1,I_1:1},{F_1,I_2,I_1:1},{I_2:1}}	<F_1:2,I_1:2>	{F_1,I_5:2},{F_1,I_1,I_5:2}
I_1	{{F_1:2},{F_1,I_2:1}}	<F_1:3>	{F_1,I_1:3}
I_2	{{F_1:2}}	<F_1:2>	{F_1,I_2:2}

3　IFP-Growth算法应用分析

基于 2016 年某型动车组牵引系统的真实故障数据进行分析，每条故障数据包含故障编号、列车编号、故障名称、故障时间、车厢号、速度、经度、纬度、运行里程等属性。实验使用 4 台 x86_64 架构的 PC 机作为硬件设备，其中，1 台作为 Master 和 NameNode 结点，3 台作为 Slave 和 DataNode 结点，4 台计算机的软硬件环境配置相同，如表 3 所示，分别预装了 Centos 操作系统，搭建了 Hadoop 集群环境，包括 HDFS、Hive 以及 Mahout 等。Mahout 是 Apache 的一个开源机器学习库，提供了分类、聚类、协同过滤、FP-Growth 频繁项集挖掘等经典算法的 Java 实现，可在此基础上进行扩展和改进。

表3　实验软硬件环境

硬件环境	软件环境
x86_64架构的PC	操作系统：Centos 7
CPU：4核，3.20 GHz	Hadoop版本：Hadoop-2.5.0
内存：16 GB	Hive版本：Hive-0.12.0
硬盘：1 TB	Java开发环境：jdk1.8.0.77
网卡：千兆以太网卡	Mahout 版本：Mahout-0.9
	IDE：Eclipse Neon.3(4.6.3)

在数据量大小分别是 20 万条，40 万条，60 万条，80 万条，100 万条，120 万条时，比较 FP-Growth 算法和 IFP-Growth 算法的性能。当数据量达到 80 万条时，IFP-Growth 算法执行时间比 FP-Growth 少，效率明显提高，如图 2 所示。

选取 80 万条故障数据，设定最小支持度为 2.5%。挖掘出的部分关联规则，如表 4 所示。其中，规则“牵引电机温度高 3520，111° 8′ ~ 120° 52′，27° 15′ ~ 45° 2′”表示列车编组编号为 3520

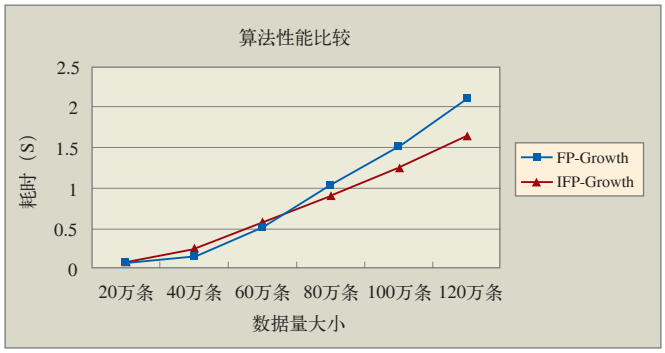

图2　算法性能比较

的牵引电机在东经 111° 8′ ～ 120° 52′ 、北纬 27° 15′ ～ 45° 2′ 的位置范围频繁发生温度高的故障。规则“牵引电机温度高 9 月，90 万 km”表示运行里程达到 90 万 km 的列车的牵引电机在 9 月份频繁发生温度高的故障。因此可在每年 9 月份在东经 111° 8′ ～ 120° 52′ 、北纬 27° 15′ ～ 45° 2′ 的地理范围对编号为 3520 的列车运行 90 万 km 及以上时对其牵引电机进行检修。

表4　部分关联规则

序号	规则	支持度计数	支持度
1	牵引电机温度高⇒3520，111° 8′～120° 52′，27° 15′～45° 2′	32 375	4.05%
2	牵引电机温度高⇒9月，90万km	29 726	3.72%
3	轴承轴温⇒3 601，8月，60万km	31 533	3.94%
4	牵引变流器传输不良⇒105° 76′～115° 8′，6月	26 421	3.31%

4　结束语

针对动车组的故障预测研究，本文提出了应用于动车组故障数据关联规则挖掘的改进 FP-Growth（IFP-Growth）算法，该算法提高了挖掘动车组故障与运行状态之间关联关系的效率。在有规则表明动车组关键零部件故障在某个时间段或地点频繁发生时对其进行维修，能够提高维修效率，增加动车组运行的安全可靠性。

参考文献：

[1] 常振臣，张海峰．动车组 PHM 技术应用现状及展望 [J]．电力机车与城轨车辆，2016, 39（1）：1-4.

[2] 钟　雁，马海漫，张　春．改进的 FP-tree 算法在动车组故障诊断中的应用研究 [J]．交通运输系统工程与信息，2013，13（6）：105-111.

[3] 吴宗翰．关联规则挖掘算法研究及其在交通事故分析中的应用 [D]．天津：南开大学，2015.

[4] 尹士闪，马增强，毛晚堆．基于频繁项目集链式存储方法的关联规则算法 [J]．计算机工程与设计，2012, 33（3）：1002-1007.

[5] 周诗慧．基于 Hadoop 的改进的并行 FP-Growth 算法 [D]．济南：山东大学，2013.

[6] Jiawei Han, Jian Pei, Yiwen Yin. Mining frequent pattems without candidate generation[J]. ACM SIGMOD Record, 2000, 29(2):1-12.

[7] 王新宇，杜孝平，谢昆青．FP-growth 算法的实现方法研究 [J]．计算机工程与应用，2004, 40（9）：174-176.

[8] Shandong Ji, Dengyin Zhang, Liu Zhang. Paths sharing based FP-growth data mining algorithms[C]. International Conference on Wireless Communications & Signal Processing (WCSP), 2016:1-4.

[9] 吕雪骥，李龙澍．FP-Growth 算法 MapReduce 化研究 [J]．计算机技术与发展，2012, 22（11）：123-126.

[10] L. Deng, Y. Lou. Improvement and Research of FP-Growth Algorithm Based on Distributed Spark[C]//International Conference on Cloud Computing and Big Data (CCBD), 2015:105-108.

[11] 章志刚，吉根林．一种基于 FP-Growth 的频繁项目集并行挖掘算法 [J]．计算机工程与应用，2014, 50（2）：103-106.

责任编辑　徐侃春