分类号 C93/64　　密级
U D C 0004258　　编号 10741

兰州财经大学
LANZHOU UNIVERSITY OF FINANCE AND ECONOMICS

硕 士 学 位 论 文

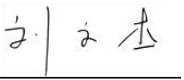
论文题目 闭频繁项集挖掘算法在 ABC 库存管理优化问题上的研究与应用

研 究 生 姓 名：刘文杰
指导教师姓名、职称：杨海军 教授
学 科、专 业 名 称：管理科学与工程
研 究 方 向：物流信息系统分析与应用

提 交 日 期：

独创性声明

本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文作者签名：  签字日期：2022 年 5 月 29 日

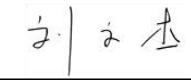
导师签名：  签字日期：2022 年 5 月 29 日

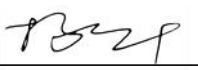
关于论文使用授权的说明

本人完全了解学校关于保留、使用学位论文的各项规定，同意（选择“同意”/“不同意”）以下事项：

1.学校有权保留本论文的复印件和磁盘，允许论文被查阅和借阅，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文；

2.学校有权将本人的学位论文提交至清华大学“中国学术期刊（光盘版）电子杂志社”用于出版和编入 CNKI《中国知识资源总库》或其他同类数据库，传播本学位论文的全部或部分内容。

学位论文作者签名：  签字日期：2022 年 5 月 29 日

导师签名：  签字日期：2022 年 5 月 29 日

Research and Application of Closed Frequent Itemset Mining Algorithm in ABC Inventory Management Optimization Problem

Candidate : Liu Wenjie

Supervisor : Yang Haijun

摘 要

库存物资的合理分类对企业库存管理活动至关重要，对于每类物资根据其性质有针对性地制定相应的库存控制策略，可以降低库存成本，优化库存管理活动，其中ABC库存分类法是最基础的方法。现有ABC分类研究中大多数仅考虑独立需求物资，没有考虑物资之间内在的关联性。频繁项集挖掘算法产生的频繁模式包含着物资间的内在联系，可以用来优化ABC库存分类，但是挖掘出的频繁模式数量巨大，会导致某些项重复出现，在库存分类调整时可能会出现分类矛盾的问题，而利用闭频繁项集压缩无损的特点就能很好地解决这一问题。闭频繁项集是相关频繁项集的精简表示方式，闭频繁项集挖掘极大减少了挖掘结果中的频繁项集数量，成为近年来数据挖掘领域的一个重要研究课题。当前多个闭频繁项集挖掘算法已经提出，均可应用于ABC库存分类优化问题。DCI_Closed算法是一个经典的闭频繁项集挖掘算法，经过分析发现，其剪枝策略等方面仍有改进空间，算法效率有待提升。因此本文提出新剪枝策略来优化DCI_Closed算法的搜索空间，并据此提出改进算法DCI_ESCS，其次利用DCI_ESCS算法对ABC库存分类问题进行优化。研究内容如下：

（1）本文将储存所有2-项集支持度信息的ESCS结构（Estimated Support Co-occurrence Structure）应用到经典闭频繁项集挖掘算法DCI_Closed上，提出针对2-项集的ESCS剪枝策略，最终得到改进的DCI_ESCS算法。并在SPMF公开资源库中connect、pumsb、chess、pumsb_star、accidents五个数据集上、不同最小支持度阈值下进行实验，对比分析算法改进前后的时间性能。实验结果表明，改进的DCI_ESCS算法在事务和项集较长、较稠密的数据集上表现良好，时间效率均有一定程度的提高。

（2）本文将DCI_ESCS算法应用到物资关联性ABC库存管理优化问题上。首先对原始数据进行初步分类作为后期分类调整的依据，然后用改进的DCI_ESCS算法挖掘出闭频繁模式并筛选出其中有效的模式，最后利用有效模式各物资的关联性调整部分初始分类。通过进一步分析可以得出，调整后的库存分类能够优化库存管理，降低库存成本，而且也能提高服务水平。

关键词：ABC库存分类 库存管理 闭频繁项集 剪枝策略

Abstract

The reasonable classification of inventory materials is very important to the inventory management activities of enterprises. For each kind of materials, it can reduce the inventory cost and optimize the inventory management activities by formulating the corresponding inventory control strategy according to its nature. Among them, ABC inventory classification method is the most basic method. Most of the existing ABC classification studies only consider the independent demand for materials, without considering the internal correlation between materials. Frequent itemsets mining algorithm of frequent patterns contain the inner link between the material and can be used to optimize the ABC inventory classification, but a huge number of mining the frequent patterns can lead to some repeated, contradictions in the adjustment may occur when the inventory classification problem, and the use of the characteristics of frequent closed itemsets compression condition can solve this problem well. Closed frequent itemsets are a simplified representation of related frequent itemsets. Mining closed frequent itemsets greatly reduces the number of frequent itemsets in mining results, which has become an important research topic in data mining in recent years. Several closed frequent item set mining algorithms have been proposed, which can be applied to the ABC

inventory classification optimization problem. DCI_Closed algorithm is a classic closed frequent item set mining algorithm. Through analysis, it is found that there is still room for improvement in pruning strategy and the efficiency of the algorithm needs to be improved. Therefore, this paper proposes a new pruning strategy to optimize the search space of DCI_Closed algorithm, and then proposes an improved algorithm DCI_ESCS. Then, the DCI_ESCS algorithm is used to optimize the ABC inventory classification problem. The research contents are as follows:

(1) In this paper, the Estimated Support co-occurrence Structure (ESCS), which stores all the Support information of 2-item sets, is applied to the classical closed frequent item set mining algorithm DCI_Closed, and an ESCS pruning strategy for 2-item sets is proposed. Finally, the improved DCI_ESCS algorithm is obtained. Experiments were carried out on five data sets of CONNECT, PUMSB, Chess, PUMSB_STAR and Accidents in SPMF open resource database under different minimum support thresholds to compare and analyze the time performance of the algorithm before and after improvement. Experimental results show that the improved DCI_ESCS algorithm performs well on long and dense data sets, and its time efficiency is improved to some extent.

(2) In this paper, DCI_ESCS algorithm is applied to the optimization problem of material-related ABC inventory management. First, the

original data are preliminally classified as the basis for classification adjustment in the later stage. Then, the closed frequent patterns are mined and the effective patterns are screened by the improved DCI_ESCS algorithm. Finally, some initial classifications are adjusted by the correlation of the materials in the effective patterns. Through further analysis, it can be concluded that the adjusted inventory classification can optimize inventory management, reduce inventory cost, and improve service level.

Keywords： ABC inventory classification; Inventory management; Closed frequent itemsets; Pruning strategy

目　录

1 引言

1.1 研究背景与意义

对库存物资进行分类是库存管理中非常重要的一项工作。针对每类库存物资制定相应库存管理策略优化管理活动，不仅可以减少库存空间占用，降低企业经营成本，而且也能提高订货完成率（Order Fill Rate）。传统的ABC库存分类方法由Dickie等[1]引入，并成功地应用于通用电气公司的库存处理。传统ABC库存分类方法将库存物资-也称为库存单位 (Stock Keeping Unit，SKU)按年货币使用量的多少分为A、B、C三类，即其重要性逐次递减，同时对每一类SKU采用不同的库存管理策略。虽然其分类标准单一，实施简单，但对许多库存管理重要指标没有考虑，导致其分类结果具有较大局限性。因此，Flores等[2]提出了多准则ABC库存分类方法(Multi-criteria ABC Inventory Classification，MCIC)，综合考虑提前期、成本、价格和关键性等多种要素的影响。在实践中，Ramanathan等[3]认识到还有其他重要的定量和定性准则需要考虑，例如“通用性、陈旧性、可替代性、每年的请求数量、稀缺性、耐用性、可修复性、订单规模、可储存性和缺货惩罚成本”。目前已有多种研究模型用以解决MCIC问题，具体方法涉及层次分析法、数学规划、群智能算法、人工智能及多种方法融合算法等方面。其中无监督机器学习方法在研究MCIC问题上取得不错的进展，其最大优点在于不涉及主观性。

以上模型仅从要素角度定性和定量地决定库存，没有考虑到库存物资之间的关联性。而在实际的订货过程中，供货方往往要考虑在一个订单中用户同时订了哪些货。比如在多个订单中A、B物资同时出现，说明A与B具有很高的关联度，如果在采购时只采购了A物资，而B物资库存不足，那么会影响用户体验，从而导致订单数减少。因此，找出物资之间的关联性，合理调整库存物资分类，对一些关键物资进行重点控制管理对于企业具有十分重大的意义。文献[4]、[5]利用频繁项集挖掘算法来发现库存关联性，进而优化库存分类，但是存在着挖掘出的频繁模式数量过多导致在改进库存分类时部分药品分类时可能出现交叉的问题。比如现有I_1、I_2、I_4三种药品，分别属于A、B、C类。应用频繁项集挖掘算法，产生部分关联模式和支持度如下：$\{I_1,I_4:4\}$、$\{I_2,I_4:4\}$、$\{I_1,I_2,I_4:4\}$。由于这三个项

集支持度都为4，其分类调整的重要性优先级是相同的。对于药品I_4来说，如果按照第一个和第三个药品项集，I_4与I_1具有很强的关联性，在库存管理中应采用与I_1相同的库存控制水平，因此将其从C类调整到A类；同理，按照第二个药品项集，I_4的库存控制水平应与I_2相同，因此将其从C类调整到B类，这样就会出现分类矛盾的情况。

模式挖掘是数据挖掘领域中非常关键的一个分支，能够挖掘出各项间隐藏的重要的关联性信息。频繁项集挖掘(Frequent Item Mining，FIM)是模式挖掘中最基础的内容，旨在挖掘频繁地同时出现在数据库中的项，假定事务中每个项的价值都相同并且仅考虑项集在交易事务中出现的总次数。然而，FIM的结果通常是很大的集合，尤其是当数据集很密集或者阈值很小时，因此衍生出闭项集这一概念，并引申到频繁项集的研究中，发展成为闭频繁项集挖掘算法，生成的闭频繁项集中的元素明显少于频繁项集，但是不会损失任何信息，可以从发现的闭频繁项集恢复到所有频繁项集。因此，使用闭频繁项集挖掘算法可以在很大程度上减少存储空间和内存使用。在现有的闭频繁项集挖掘算法研究中，剪枝策略相对单一，大都是针对1-项集进行剪枝，对于2-项集和n-项集（$n\geq3$）的剪枝策略相对匮乏，而有效的剪枝策略可以提前发现并剪掉大量没有希望的项集，因此改进闭频繁项集的剪枝策略对于此类算法效率的提升具有很大的帮助。

针对上述提到的两个问题：频繁模式在优化库存分类时缺少可操作性和现有的闭频繁项集挖掘算法剪枝策略单一，本文试图从以下两个方面开展研究：

（1）在一个经典的闭频繁项集挖掘算法DCI_Closed融入ESCS结构（Estimated Support Co-occurrence Structure），提出ESCS剪枝策略对2-项集进行剪枝，提出改进的DCI_ESCS算法，减少了原算法挖掘时间，提升算法效率。

（2）利用闭频繁项集挖掘算法来解决库存分类矛盾的问题。在闭频繁项集挖掘算法中，上述$\{I_1,I_4\}$、$\{I_2,I_4\}$和$\{I_1,I_2,I_4\}$属于同一个等价类，只有最大项集$\{I_1,I_2,I_4\}$作为闭频繁模式被挖掘出，这样I_4就可以合理地从C类物资调整为A类物资。根据这些闭频繁模式之间的关联性对原库存分类进行调整，优化了库存管理方式，降低了库存成本，更好地满足客户需求。

1.2 国内外研究现状

1.2.1 频繁项集挖掘算法

频繁项集挖掘算法主要基于水平层级、基于模式增长和基于垂直数据格式三大类。

Agrawal 等[6]提出了 Apriori 算法，是首次采用水平层级机制的算法，该算法的核心思想是 n 项集两两组合生成(n+1)候选项集以及利用向下闭包属性作为剪枝策略筛去候选项集中大量没有希望的项集，以达到提高挖掘效率的效果。但是该算法存在不足之处：候选项集数量多、读取数据库次数多。Agrawal 等[7]针对 Apriori 算法的不足之处进一步研究，提出了 AprioriTid 的新算法，该算法使用了 Tid 结构来储存所有候选项集的相关信息，因此不需要重复扫描。Ashoka等[8]采用了Apriori 算法的基本思想,提出了Partition 算法。Partition 算法采用划分思想，首先把数据库分成若干子区，进行第一次读取挖掘出每个子区的频繁项集合成候选项集。接着对数据库进行第二次读取来发现候选项集中的所有满足最小支持度阈值的项集。

以上基于水平层级机制的算法存在着一些局限性：一方面会产生大量的候选项集，另一方面会对数据库重复扫描。为此，Han 等[9]提出了 FP-growth 算法，改变了以往算法的水平层级机制，采用了模式增长机制。FP-growth 算法构建了 FP-tree 的数据结构，将原始数据库的所有信息完整储存到 FP-tree 中，这样事务项的相关信息就不用通过反复读取数据库来获取，而是可以通过 FP-tree 结构直接获得。FP-growth 算法的步骤为：构造条件模式基、构建条件 FP-tree、挖掘频繁项集，并且该算法不产生候选项集。研究证明，FP-growth 算法与 Apriori 算法相比，运行时间大大降低，挖掘效率得到显著提升。但是该算法不适用于事务差别较大的数据库，因为事务差别较大会产生大量条件模式基，失去了压缩性的优势。

之前两大类算法的数据库格式都是水平的，即{事务:事务中的项}，表示一条事务中出现了哪些项。J.Zaki 等[10]在 2000 年提出的 Eclat 算法，采用了与以上算法完全不同的数据结构，它采用的是垂直数据格式，即 {项:事务集合}，表示一个项在哪些事务中出现过。该算法首先对数据库进行一次读取，把数据从

{事务:事务中的项}转换成{项:事务集合}的形式，通过所有 1-项集事务集合的长度计算它们的支持度，比如对于项 i 有 TID=[1,2]，说明该项在两条事务中出现过，则该项的支持度计数为 2。接下来的挖掘过程及剪枝策略和 Apriori 算法相同，不同的是在使用频繁 K-项集构造候选（K+1）-项集时采用交运算的方法，即候选（K+1）-项集的 TID_set 是其 K-项子集的 TID_set 的交集。因此，Eclat 算法采用的垂直数据格式只在转换格式的时候扫描数据库，各个项集的支持度信息储存在 TID_set 中。但是 Eclat 算法也存在不足之处：在 TID_set 很长的情况下，可能会消耗较多的储存空间，构造候选（K+1）-项集时交运算比较复杂，会增加算法的运行时间。为了解决以上问题，王红梅等[11]在 Eclat 算法的基础上进行改进，提出了有效的改进算法，该改进算法能够在不同段上分别挖掘频繁项集，并在分段求交集的过程中预测支持度的大小，能够非常有效地减少交运算时消耗的时间，从而提高了算法挖掘效率，并且对于项集长度较长、较稀疏的数据集效率提高更加明显。

1.2.2 高效用项集挖掘算法

现有的高效用项集挖掘算法从阈值角度出发可以划分成单一阈值、多阈值以及不设阈值的高效用项集挖掘算法。

单一最小效用阈值高效用项集挖掘算法是为数据库中的每一项都设定一个固定的阈值，判断一个项集是否具有高效用即判断该项集的效用值是否大于或等于设定的固定阈值。Two-Phase 算法[12]首先采用层级搜索的方式由 n-项集两两组合生成(n+1)-候选项集，然后对生成的(n+1)-候选项集进行筛选，保留所有满足阈值的项集。Two-Phase 算法为了获取效用值信息必须反复读取数据库，并且组合而成的候选项集数目非常大，针对以上局限性，IHUP 算法[13]被提出。该算法也使用了 FP-tree 的数据结构，提出 IHUP-tree 的新结构，将原始数据库各项的支持度和事务加权效用 TWU 值保存到每个节点上，一定程度上节约了占用内存。以上算法都是两阶段算法，会生成大量候选项集，而一阶段算法则直接计算项集的效用值来挖掘高效用项集，不产生候选。FHM 算法[14]通过构建了估计效用共现结构 EUCS 用来保存所有 2-项集的事务加权效用值，针对 2-项集进行剪枝，剪去大量没有希望的 2-项集及扩展项集，减少了连接次数和算法的内存占用，提

高了算法效率。为了解决算法运行和内存消耗时的计算成本，一种新的EFIM算法[15]被提出，该算法依赖于两个新的上界：修正子树效用(Revised Sub-tree Utility)和局部效用(Local Utility)，这两个效用值作为剪枝条件更加具有约束力，利用它们进行剪枝可以剪去更多非高效用的项集。EFIM算法还引入了一种新的基于数组的效用计算技术，即快速效用计算，以在线性时间和空间中计算这些上界。此外，为了降低数据库扫描的成本，EFIM算法还提出了高效的数据库投影和事务合并技术，即HDP (High-utility Database Projection)和HTM (High-utility Transaction Merge)，这些技术也是在线性时间内完成的。实验研究表明，EFIM在密集数据集上通常比目前最先进的算法快2到3个数量级，同时在稀疏数据集上表现得也相当好。此外，EFIM的一个关键优势是它的内存消耗很低。

在实际生活中。每种物品自身性质不同，市场价值也不同，用相同的最小效用阈值来衡量是否为高效用物品不符合实际。因此，综合考虑每个项的性质、价值等因素，为其设置一个合理的最小效用阈值来判断是否具有高效用是十分有必要的。Lin 等[16]采用了HUIM-MMU(HUIM with Multiple Minimum Utility Thresholds)框架来克服以上局限。根据这个框架，用户可以为每个项指定不同的阈值，以发现高效用项集。为了实现HUIM-MMU框架，首先提出了一个名为HUI-MMU的算法，它依赖于一个新的排序向下闭包(Sorted Down Closure, SDC)属性和最小效用阈值(Least Minimum Utility Threshold, LMU)。此外，还提出了一种基于TID-index策略的改进算法HUI-MMUTID，以提高挖掘性能。在现实生活数据集和合成数据集上的大量实验表明，两种算法在考虑多个最小效用阈值的情况下，可以有效地发现事务数据库中高效用项集的完整集合。该作者在文献[16]原先工作的基础上进行扩展，提出改进的HUI-MMUTE算法[17]，该算法融入了FHM 算法中的EUCP(Estimated Utility Co-occurrence Pruning)策略，对于EUCS估计效用共现结构中的2-项集利用 SDC 属性进行剪枝，删除大量不满足阈值条件的低效用项集，算法效率得到显著提升。对于以上提到的 HUI-MMU 、HUI-MMUTID、HUI-MMUTE三种算法[16][17]，会生成大量的候选项集，影响了算法效率。对此，文献[18]提出基于深度优先搜索方式的HIMU 算法和扩展HIMU-EUCP算法。该文章采用一个MIU-tree数据结构来储存数据库的效用信息，并且提出

GDC 属性(Global Downward Closure) 和 CDC 属性(Conditional Downward Closure)来高效剪枝。HIMU-EUCP 算法采用 HUI-MMUTE 算法相同的思想，在 HIMU 算法的基础上进行扩展，能够非常有效地提高算法挖掘效率。文献[19]提出一个基于多效用阈值的 MHUI 算法，该算法是一阶段的算法，不需要进行大量的中间候选生成过程并且引入了后缀最小效用的概念，提出了有效挖掘高效用项集的四个剪枝策略：TWU-M-Prune、U-M-Prune、EUCS-M-Prune、LA-M-Prune。最后在 8 个标准数据集上对该算法的性能进行了评价，并与 HUI-MMUTE 算法和 HIMU-EUCP 算法进行了比较。实验结果表明，该算法的时间效率比 HUI-MMUTE 算法提高了 2-3 个数量级，比 HIMU-EUCP 算法提高了 1-2 个数量级，特别是在中等长度和密集的标准数据集上。此外，该算法的内存占用也显著降低。上述 MHUI 算法在求各项集及其扩展项的最小效用阈值时需要重新比较、重复计算，因此文献[20]提出了一种快速挖掘算法 FMHUI 算法。该算法进一步改进了最小效用阈值的计算模式，即利用前一次的结果，不需要多次、大量的计算过程。

高效用项集挖掘算法通过每个项集出现次数和效用值来获得高效用项集，许多算法通过预先设置最小效用阈值来计算高效用项集。然而，确定最小效用阈值并不容易，阈值过高或过低都可能导致挖掘结果的不准确。基于此，一系列不需要预先假定最小效用阈值的高效用项集挖掘算法被提出，如 Top-k 高效用项集挖掘算法和基于二进制粒子群优化的算法。Top-k 高效用项集[21]挖掘算法的中心思想是设置一个数目 k 来限制挖掘出的高效用项集的数量，比如设定 k=10，则效用值从高到低排在前 10 位的项集被挖掘出。因此，该类算法可以精确地控制挖掘结果的大小，并帮助用户更容易找到能够产生最高利润的有用项集，成为近年来一个新的研究热点。Tseng 等[22]提出了 TKO 算法，该算法采用垂直数据格式的效用列表结构来保存数据库相关数据。当一个项集出现时，可以通过其效用列表得到效用值。TKO 算法提出一种新的 RUC（Raising the threshold by the Utilities of Candidates）策略，快速提高了最小效用阈值。此外。该算法使用了 RUZ（Reducing Estimated Utility Values by using Z-elements）和 EPB（Exploring the Most Promising Branches First）策略，有效增加了初始的效用阈值，减小了项集的预估效用值，其性能得到了提升。Duong 等[23]提出了基于效用列表的 kHMC 算法，该算法使用了 EUCPT （Estimated Utility

Co-occurrence Pruning Strategy with Threshold）和 TEP（Transitive Extension Pruning strategy）剪枝策略来减小搜索空间。EUCPT 策略使用项集共现信息，避免了大量的连接操作。TEP 策略则利用项集效用的新上限来减小搜索空间。此外，kHMC 算法使用一种早期放弃策略，取消了对应项集不是 Top-k 高效用项集的效用列表的构建。Srikumar 等[24]认为，提高最小效用阈值的策略对密集数据库无效，特别是在算法进行的初期过程，因此提出了 THUI 算法。THUI 算法使用了一种新的 LIU（Leaf Itemset Utility）结构，该结构采用了三角矩阵的形式，能够更紧缩地储存数据库的相关数据，在增加最小效用阈值上具有更好的效果。针对以上算法只能处理中小规模的数据，在海量数据上性能明显下降的问题，Han 等[25]提出了一种新的 PTM 算法。 PTM 算法把原始数据库划分为一组带有前缀模式的子区，所有子区都能够读取到内存中，同时将数据用同样的前缀模式保存。PTM 的效用值可以通过对应子区中的数据得到，并且按照平均事务效用的顺序处理基于前缀的分区，以更快地提高效用阈值。此外， PTM 使用一种简洁的辅助数据结构，可以直接跳过大部分子区，大大提升了效率。对于要处理的分区，PTM 在有希望的项目的集合枚举树上利用深度优先搜索来找到所需的结果。该算法还设计了基于全后缀效用的子树剪枝规则，有效地减少了集合枚举树的探索空间。大量实验结果表明，PTM 能够有效地发现海量数据上的前 k 个高效用项集。Gunawan 等[26]提出了一种基于二进制粒子群优化的 HUIM-BPSO-nomut 算法来优化高效用项集的搜索，不需要预先设置最小效用阈值。该算法包括四个阶段:预处理、初始化和编码粒子、适应度评估和更新阶段。在预处理阶段，分别根据交易和利润数据计算所有交易中每个项目的效用和每个交易的总效用，并根据总效用将交易按降序排序。第二阶段，通过形成粒子群优化算法的种群来初始化粒子群优化算法。总体由 pop_size 数量的粒子组成，每个粒子代表数据库中的一个事务。一个粒子在形式上被定义为维 d 的向量 $\vec{p}$，其中 d 是所有交易中不同项目的数量。对于一个交易 T，如果 $i \in T$，$p_i = 1$；如果 $i \notin T$，$p_i = 0$。

除了从阈值角度划分之外，还有一些其他类型的高效用项集挖掘算法陆续被提出，比如近似的、含负值的、局部和峰值的高效用项集挖掘算法等等。传统高效用算法是在假设数据库中存储的数据无缺陷的情况下执行的。如果在给定的数据库中存在未知的错误，如噪声，那么就会影响算法性能。针对此问题，Baek

等[27]研究了一种在有噪声的数据库中挖掘近似高效用项集的 AHUPM 算法。AHUPM 算法提出一种效用公差因子的概念，来计算项集的有效效用范围。此外，AHUPM 算法在挖掘过程中引入了 AHUP-tree（Approximate High Utility Pattern-tree）结构和 atwu（Approximate Transaction Weighted Utilization）剪枝策略，以便更紧凑地维护必要信息以及大大减少了模式的搜索空间。实验结果表明，算法在执行时间、内存使用和可扩展性等方面都具有较好的性能。但是该算法的局限性在于寻找最优效用公差因子。针对于传统高效用项集挖掘算法局限于只有正效用的数据集忽略了现实生活负效用重要性的问题，Singh 等[28]提出了一个有效解决该问题的算法：EHNL 算法。在效率提高方面，该算法采用子树剪枝策略、数据集投影和事务合并技术，减少搜索空间和数据集扫描成本。在效用计算方面，该算法使用基于数组的效用计数技术来计算线性时间和可忽略内存空间的效用值。此外，为了将子树剪枝策略和具有长度约束的负效用项集挖掘结合起来，该算法根据事务的效用值对事务中的项进行非递增排序，并利用偏移指针将负的项保留在已排序项的末尾。通过实验证明，EHNL 算法能有效地发现真实数据集的负效用项集，且内存消耗低。针对传统的高效用项集挖掘算法的没有考虑到项集的效用会随着时间的变化而变化，因此无法找到在考虑整个数据库时不能产生高效用价值，但在某个特定时间段具有高效用价值的项集，Fournier-Viger 等[29]首次定义了局部高效用项集的概念，并在此基础上定义了其扩展概念：峰值高效用项集，该扩展是指寻找项集产生比通常高得多的效用(峰值)的时间段。文献[29]提出了 LHUI-Miner 和 PHUI-Miner 两种算法来挖掘这些模式。此外，由于峰值高效用项集的集合可能很大，而且峰值高效用项集中的某些项对其峰值贡献不大，因此提出了第三种名为 NPHUI-Miner 的算法来发现一组更小的模式，称为非冗余峰值高实用项集(Non-redundant Peak High Utility itemset, NPHUIs)。实验结果表明，该算法可以发现传统算法无法发现的有用模式，并且可以减少运行时间和内存消耗。

1.2.3 闭项集挖掘算法

（1）闭频繁项集挖掘算法

现有的经典的闭频繁项集挖掘算法和频繁项集挖掘算法一样，大致分成基于

水平层级机制、基于模式增长机制和基于垂直数据格式机制三种类型。

A-CLOSE[30]延续了 Apriori 算法的水平层级机制，首先通过 Apriori 策略逐层浏览频繁项集格，挖掘每个等价类的最小元素。在第二步中，A-CLOSE 计算之前找到的所有最小生成器的闭包。由于单个等价类可能有多个最小项集，因此可能会计算冗余闭包。此外，A-CLOSE 性能受到离线闭包计算高成本和大量子集搜索的影响。

为了解决水平层级机制算法项集连接成本昂贵的问题，一些闭频繁项集挖掘算法也采用了模式增长的机制。CLOSET[31]采用与 FP-growth 相同的 FP-tree 数据结构将原始数据库中的相关信息储存到对应的路径结点中，并且采用了单前缀路径压缩技术来快速识别闭频繁项集。此外，该算法为了实现在大型数据库中挖掘的可扩展性，构建了一个分区投影机制。实验证明，CLOSET 算法性能优于 A-CLOSE 算法和 CHARM 算法。但在稀疏数据集或支持度阈值较低时，其性能仍然会受到影响。CLOSET+[32]也使用了 FP-tree 结构，但与 CLOSET 算法不同之处体现在以下几方面：①采用混合树投影方法，对不同类型不同性质的数据集采用不同的物理树投影机制，有效提高了空间效率。②使用项集跳过技术来修剪搜索空间。③使用高效的子集检查方法确保新发现的项集是闭项集。研究结果证明，就运行时间、内存使用和可扩展性而言，CLOSE+相对于现有挖掘算法具有一定优势。FPClose 算法[33]使用 FP-tree 结构的另一种变体—CFI-tree，用于检查频繁项集的闭合性。此外，采用一种新的 FP 阵列技术，该技术能够大大减少遍历次数，尤其适用于稀疏数据集。实验结果表明，FPClose 闭项集检测方法比 CLOSET+方法更有效。

以上算法的数据均为{事务：项}的形式，一些算法将其进行转换，转换为{项：事务集}的形式。CHARM[34]使用了一些创造性的思想：①不同于之前算法只探索项目集空间，CHARM 通过项-事务搜索树 IT-tree 结构同时探索项目集空间和事务空间。②采用了一种新的搜索技术，能够忽略项-事务搜索树的很多层级，不必枚举一些非闭合的子集，能够提高搜索效率。③使用纵向数据表示 diffsets 技术减少 TID 交集计算的内存占用。④使用一种快速的基于散列的方法来移除在计算过程中发现的任何“非封闭”集合，显著压缩候选项集。在大量真实和合成数据库上进行的广泛实验评估表明，CHARM 在算法效率上比现有算法提高了 1-2 个数量级，在事务数量上也是可线性扩展的。DCI-Closed[35]引入两个变量：PRE_SET

和 POST_SET。其中 POST_SET 用于构建所有可能的生成器，PRE_SET 用于进行生成器重复检查。实验证明，DCI-Closed 算法优于 CLOSE+和 FPClose。DCI_Closed 算法在 3.1 节做了详细描述，在这里不再过多说明。

表 1.1 对以上几个经典的有代表性的闭频繁项集挖掘算法进行了总结归纳。

表 1.1 经典闭频繁项集挖掘算法归纳

算法名称	采用机制	搜索顺序	数据结构	特点
A-CLOSE	水平层级	广度优先	—	项集连接操作计算成本昂贵
CLOSET	模式增长	深度优先	FP-tree	性能优于 A-CLOSE 算法和 CHARM 算法，不适用于稀疏数据集或支持度阈值较低的情况
CLOSET+	模式增长	深度优先	FP-tree	优于现有 CHARM、CLOSET 算法
FPClose	模式增长	深度优先	CFI-tree	在闭项集检测方法上比 CLOSET+方法更有效
CHARM	垂直数据格式	深度优先	IT-tree	能够显著压缩候选项集，事务数量上可线性扩展
DCI-Closed	垂直数据格式	深度优先	IT-tree	在任何情况下都优于 CLOSET+和 FPClose

近几年，国内外学者在闭频繁项集挖掘算法问题上积极探索创新，取得不错的研究成果。党红恩等[36]提出一种基于数据变换与并行运算的 DTPC 算法，该算法利用质数对数运算的方法，将复杂的数据转换成简单的数字，在 Spark 平台上进行闭频繁项集的挖掘。实验证明，该算法能够有效减少算法的运行时间，并且节约了计算资源成本。Aryabarzan 等[37]提出一种快速挖掘的 NECLATCLOSED 算法，该算法使用项目集搜索树来表示搜索空间。对数据库扫描以识别包含 1-项集的 TSets，基于 TSets 识别出所有的频繁 1-项集作为根目录的子目录。此外，算法还提出一种快速包容检查的技术，使用一个 hashmap 结构将闭频繁项集的有序列表与支持度关联起来进行快速检查。实验证明 NECLATCLOSED 算法在大多数情况下都优于以上主流算法，尤其是在运行时间上。

（2）闭高效用项集挖掘算法

现有的闭高效用项集挖掘算法可以分成一阶段和两阶段两大类。

两阶段的经典闭高效用项集挖掘算法有 CHUD[38]、 LHUCI-Miner[39]、EFIM-Closed[40]。CHUD[38]采用 REG（Removing the Exact utilities of items from the Global TU-Table）、RML（Removing the Mius of items from Local TU-Tables）、DCM（Discarding Candidates with a MAU that is less than the minimum utility threshold）三种剪枝策略来进行剪枝。和 DCI-Closed 算法一样，CHUD 算法采用 IT-tree (Itemset-Tidset Pair Tree)的数据结构进行挖掘，在 IT-tree 中，每一个节点 N(X)包含项集 X、X 所在的 TID 集合、两个有序集合：PrevSet(X)和 PostSet(X)。与 DCI-Closed 不同的是，每个节点 N(X)与一个估计效用值（TWU）相关联。此外，该算法使用了 DAHU 技术，能够从挖掘出的闭高效用项集还原到所有的高效用项集。Mai 等[39]提出了一种 LHUCI-Miner 算法，利用效用值置信度以及闭高效用项集格来发现生成闭高效用项集。实验证明，该算法的挖掘效率得到显著提升。EFIM-Closed[40]使用两个新的剪枝上界：Sub-tree Utility 和 Local Utility，以及一个基于数组的效用值计算方法-快速效用计算。此外，为了降低数据库读取的成本，EFIM-Closed 提出了两种高效的技术：高效用数据库投影和高效用事务合并。最后，为了发现闭高效用项集，提出了 3 种机制:前向闭合检查、后向闭合检查以及闭合跳跃。实验结果表明，与最先进的 CHUD 算法相比，EFIM-Closed 算法的速度提升可达 71 倍，内存消耗减少可达 18 倍。

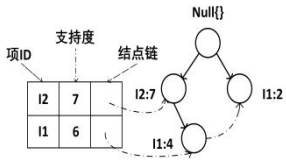
一阶段的经典闭高效用项集挖掘算法有 CHUI-Miner[41]、CLS-Miner[42]、IncCHUI[43]。CHUI-Miner[41]是第一个不产生候选的闭高效用项集挖掘算法，该算法引入了 EU-List（Extended Utility-list）的数据结构来发现闭高效用项集，以项 X 为例，X 的 EU-List 由 X 的效用列表、支持度计数、效用单元数组以及 X 的前置集 PrevSet(X)和后置集 PostSet(X)两个有序集合组成。在一些真实数据集和合成数据集上进行实验来评估提出算法的性能，实验结果表明，与之前两阶段的算法相比，CHUI-Miner 算法省略了候选项集的生成过程，挖掘效率得到显著提升，特别是对于密集数据集的挖掘。CLS-Miner[42]利用效用列表结构直接计算项集效用而不产生候选，采用 Chain-EUCP(Chain-Estimated Utility Co-occurrence Pruning)、LBP(Lower Branch Pruning)和 Coverage(Pruning by

Coverage)三种剪枝策略，并且采用了子集验证的技术，能够有效地提升算法的效率。实验证明，CLS-Miner 算法比现有算法相比在挖掘时间上得到显著减低。以上闭高效用项集挖掘算法假设数据库是静态的，这使得在增量数据的情况下挖掘非常昂贵，因为对每批新事务都需要处理整个数据集。因此 IncCHUI 算法[43]被提出，提出了一种增量效用列表结构，可以从增量数据库中有效地挖掘闭高效用项集。此外，还采用了有效的剪枝策略来快速构建增量效用列表，并消除未更新的候选项。最后，提出了一种高效的基于哈希表的方法来更新或插入挖掘的项集。实验证明，该算法比之前算法效率更高，并且能够有效处理动态数据。

（3）闭项集挖掘算法研究性质归纳

通过以上大量文献对频繁项集、高效用项集、闭项集包括闭频繁项集和闭高效用项集的研究分析，总结出制约算法效率的变量有：事务数据库读取次数、算法使用的数据结构、是否产生候选项集（挖掘过程是一阶段还是二阶段）、k-项集的生成方法（组合连接或计算）及方向（广度遍历或深度遍历）、剪枝策略等等。为了更好地描述各类闭项集挖掘算法的性质，本小节对几种常用的数据结构、搜索空间及剪枝策略进行详细描述，如表 1.2、表 1.3、表 1.4 所示。

表 1.2 常用数据结构及应用算法

名称	结构图	结构描述	应用算法
FP-tree		FP-tree 从树的根结点出发，将事务数据库中的所有项及支持度信息压缩进去，由于该结构在创建过程中是按每一条事务逐次增加分支，因此保留了项集的关联信息。左边的项头表由项、支持度和结点链，通过结点链使每一项指向右侧树中对应的位置。	CLOSET CLOSET+

续表 1.2 常用数据结构及应用算法

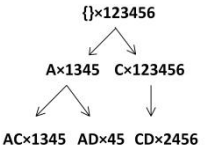
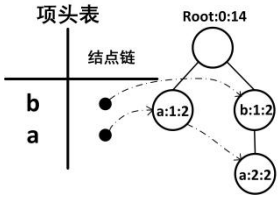
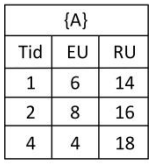
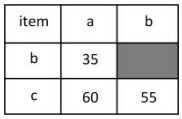
名称	结构图	结构描述	应用算法
IT-tree		IT-tree 保留了事务数据库的所有信息，其中每一个节点 N(X)包含着项集 X 以及 X 所在的 TID 集合，TID 的大小可以表示项集支持度的大小。	CHARM DCI-Closed CHUD
CFI-tree		CFI-tree 储存所有已发现的闭频繁项集以及它们的支持度计数。每一个结点由项集、支持度计数、结点链和层数构成。以上述结构图中 a:1:2 为例，表示项集 d 是处于 CFI 树中的第一层，其支持度为 2。与 FP-tree 不同的是，FP-tree 构建过程中支持度是采用增量累加的方式，而 CFI-tree 则直接用最大支持度来替换更新之前的支持度。	FPClose
Utility -list		效用列表由三部分组成：项集、项集效用和项集的剩余效用。假设事务 T1={A(2)，B(3)，E(2)},A、B、E 的效用值分别为 3、4、1。则 A 项集在 T1 中的效用=A 在 T1 中出现次数×A 的效用值=2×3=6；A 项集在 T1 内的剩余效用=B 项集在 T1 中的效用+C 项集在 T1 中的效用=3×4+2×1=14。	CHUI-Miner CLS-Miner IncCHUI
EUCS		EUCS(估计效用共现矩阵)是一个以三元组 (a,b,c)的形式存储 2-项集的 TWU 的集合，其中{a,b}是一个 2-项集,c 是它的 TWU 值,即 TWU({a,b})=c，该结构一般以三角矩阵形式实现。	CLS-Miner

表 1.3 常用搜索空间及应用算法

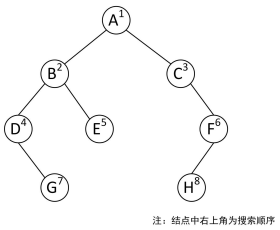
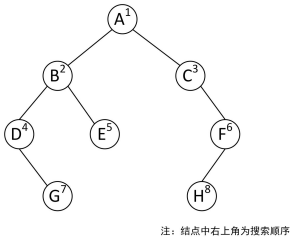
方式	示例图	搜索路径	应用算法
广度优先		根据距起点的距离，按照从近到远的顺序逐层对各节点进行搜索	A-CLOSE
深度优先		沿一条路径不断往下搜索直到找不到结点为止，然后返回，对下一条路径进行搜索	CHARM、CLOSET、CLOSET+、FPClose、DCI-Closed、CHUD、EFIM-Closed、CHUI-Miner、CLS-Miner、IncCHUI

表 1.4 常用剪枝策略及应用范围

剪枝策略	描述	应用范围	应用算法
SC 剪枝	对 1-项集 x，如果满足支持度 SC(x)＜minsup，则项集 x 不是频繁项集，并且 x 的所有扩展项集也都不是频繁项集	1-项集	闭频繁项集挖掘算法
TWU 剪枝	对 1-项集 x，如果满足 TWU(x)＜mutil，则项集 x 不是高效用项集，并且 x 的所有扩展项集也都不是高效用项集。	1-项集	闭高效用项集挖掘算法
EUCS 剪枝	对 2-项集(xy)，在 EUCS 结构中如果满足 TWU(xy)＜mutil，则项集 xy 不是高效用项集，并且 xy 的所有扩展项集也都不是高效用项集。	2-项集	CLS-Miner

1.2.4 数据流挖掘算法

（1）数据流闭频繁项集挖掘算法

在数据流上挖掘闭频繁项集的算法有 Moment[44]、CloStream+[45]、TMoment[46]等。

Moment[44]的基本思想是通过事务滑动窗口和提出一种新的数据结构—闭合枚举树 CET 来发现数据流中的闭频繁项集。通过闭合枚举树结构储存所有闭频繁项集和临界闭频繁项集，可以大量减少树节点的生成，减少插入和删除项集占用的时间和空间。CloStream+[45]构建了 Closed Table 和 Cid List 两个新的数据结构来存储数据流中闭项集的相关信息。利用 Cid List 和 SET 函数查找与添加或删除的事务有共同项的闭项集。与之前的方法不同，CloStream+算法只需要在从 SET 函数返回的闭项集和添加或删除的事务上进行交集运算，并且可以根据用户指定的最小支持度阈值在线获取所有频繁闭项集。在大量数据集上的实验结果表明，CloStream+算法挖掘性能在一定程度上有所提升。TMoment[46]提出一种更紧凑的数据结构 TCET(Transaction Translate Closed Enumeration Tree)存储和更新滑动窗口的闭频繁项集，可以有效地储存窗口的相关信息。当一个最新的事务到达窗口，最旧的事务离开窗口时，项集的支持度信息被更新。研究实验证明，与之前的算法相比，该算法内存更低，运行时间更少，并且适用于高速、无界的事务性数据流。

（2）数据流闭高效用项集挖掘算法

在数据流上挖掘闭高效用项集的算法有 DS_CRWF[47]、WCSPMPD-Stream[48]等。

DS_CRWF[47]提出了 CRWF_tree 的数据结构，利用 CRWF_tree 在滑动窗口中动态记录候选闭加权频繁模式，随着新窗口的到来和旧窗口的删除，CRWF_tree 不断更新，通过对 CRWF_tree 的一次扫描，可以得到当前滑动窗口的闭加权频繁模式。WCSPMPD-Stream[48]的核心思想是通过时间权重衰减和无更新衰减的方法来限制衰减程度，并使用一种新的数据结构 WFPS-tree 来存储数据流上的闭高效用项集。实验结果证明，与之前算法相比，该算法运行时间在一定程度上有所减少。

1.2.5 多准则ABC库存分类法

有大量文献对MCIC问题进行研究。Flores等[49]首次提出将层次分析法(Analytic Hierarchy Process，AHP) 模型用于MCIC ，以各库存要素的加权值来反映每个库存单位的相对重要性。Partovi等[50]使用一家大型制药公司维修部门的库存数据，基于层次分析法加入相关标准，用于对SKU分类。不少国内学者也对应用 AHP 来解决MCIC问题展开研究。严婷婷等[51]从制造商和分销商两个角度分别建立独立需求和从属需求库存物资 AHP 评价指标体系。徐向宇等[52]选取关键系数、缺货成本等六个因素作为分类指标，使用ABC-DEA分类法，对某企业备件库存进行分类优化。尽管AHP简单易行，但其判断矩阵的确定易受人为因素影响。李付伟等[53]将模糊层次综合评价法（Fuzzy Analytic Hierarchy Process，FAHP）应用到包装材料库存分类中，在FAHP模型中采用层次分析的方法，对评价因子权重分配进行改进。实验证明该方法一定程度上减少了主观性且库存分类结果更加合理。

Bhattacharya等[54]提出一个基于TOPSIS(Technique for Order Preference by Similarity to Ideal Solution)概念和距离的多准则共识框架进行ABC库存分类，其中考虑了单位成本、交付周期、消耗率、物品的易腐性和原材料的储存成本等因素。Chen等[55]使用两个虚拟SKU:正理想SKU(Positive Ideal Item,PII）-其在全部库存要素上得最高分值和负理想SKU（Negative Ideal Item,NII）-其在全部库存要素上得最低分值，每个实际SKU都与这两个虚拟SKU作比较，同时运用 TOPSIS 概念建立一个综合绩效指数，将每个实际SKU到PII和NII的距离合成一个综合得分，用于SKU排名分类。

数学规划(Mathematical Programming，MP）方法是研究MCIC问题的常用方法。通常针对每个SKU，数学规划分类模型能够将多个库存要素值计算成一个评分，实现多准则ABC库存分类。Ramanathan等[3]开发了一个简单的加权线性优化模型，称为 R 模型，使用加权加法函数将不同库存要素的取值为每个SKU合成为一个评分。Zhou 等[56]对 R 模型进行了扩展，提出 ZF 模型，解决了 R 模型的主要缺点：把在次要库存要素上赋值高的SKU不恰当地划分为A类。该模型每个SKU使用最有利和最不利的两组权重组合构建综合指数。Ng 等[57]在 R 模型的基础上提出了一个简单的替代模型— Ng 模型，该模型将库存项的全部要素值

变换为标量得分，再应用 ABC 原理计算分数来进行分类。Ng 模型在一定程度上进行转换，可以在没有线性优化器的情况下得到库存单位的得分。Hadi-Vencheh 等[58]在 Ng 模型上进行改进优化，提出了 H 模型，其中考虑了多准则 ABC 库存分类的权重值。国内学者丁斌等[59]提出 R 模型的一般通用模式—S 模型。当 k=1 时，S 模型就是 R 模型。S 模型利用最优库存值 S 值大小对库存进行分类，体现多个分类标准的权重影响。朱双凯等[60]提出组合赋权-ABC 分类法，耦合主客观权重，提高了库存分类的合理性。

决策树技术也应用到 SKU 多准则分类上。Porras 等[61]考虑 SKU 的关键性、需求量及价格进行组合分类，采用一次一个特征，逐步进行分类方式。对于每种组合分类都制定了具体的库存管理策略。

遗传算法、粒子群优化算法、支持向量机及人工神经网络等人工智能技术也应用到 MCIC 问题研究中。Guvenir 等[62]提出了一种利用遗传算法(Genetic Algorithm，GA)进行多准则分类的新方法。为了将遗传算法应用到权重学习问题中，文中提出一种新的交叉算子，该算子保证了生成的子代是权重向量的有效表示，实验证明新方法的分类结果更接近决策者的分类。Partovi 等[63]基于人工神经网络(Artificial Neural Network,ANN)对某制药公司库存单位进行 ABC 分类，该人工神经网络模型采用了两种学习方法：反向传播(Back Propagation，BP)和遗传算法(Genetic Algorithm，GA)。在两个数据集上对计算结果比较表明，两种 ANN 预测精度均高于多元判别(Multiple Discriminate Analysis，MDA)模型，并且这两种学习方法之间没有显著差异。 Yu 等[64]使用了支持向量机(SVMs)、反向传播网络(BPNs)和 k 近邻(k-NN)这些人工智能算法。经实验验证，这些算法表现出优于 MDA 的性能，其中 SVM 分类准确性最高。于俊甫等[65]提出一种合成少数类过采样技术-支持向量机(SMOTE-SVM)算法，在使用 SVM 分类之前先使用 SMOTE 技术进行预处理，解决了数据不平衡的问题。Tsai 等[66]使用粒子群优化(Particle Swarm Optimization，PSO)技术找到分类类别的最佳数量，在库存分类时能够参考特定的标准，简化了在分类之前必须决定有多少类别的决策过程。

最近，无监督聚类模型也被用于处理 MCIC 问题。Keskin 等[67]开发了模糊 C 均值(Fuzzy C-Means，FCM)聚类模型，使决策者能够更好地处理在多属性决策中评价属性值为模糊信息的情况。陈希等[68]提出 Canopy-FCM 模糊聚类模型，该

模型是在FCM模型的基础上引入Canopy算法，有效减小了在初始中心选取上的随机性的影响。刘晔等[69]选取单价、销量、维修周期、关键程度和订货提前期五个分类标准，将K-means 算法应用到汽车备件分类中，并采用分类调整方法对结果进行调整。Lolli 等[70]开发了一个由 AHP 和 K-means 聚类算法组成的混合模型来改进 MCIC 问题研究。该混合模型先从 AHP 获得排名得分，按以上得分再用 K-Means 进行聚类分析。吴龙涛等[71]对灰色聚类模型进行改进，引入信息熵和基于中心点的三角白化权函数进行计算分类。Zowid 等[72]提出了将高斯混合模型(Gaussian Mixture Model ,GMM)应用到MCIC问题中。GMM是一种简单的优化模型，不涉及主观性且计算时间较短。文献[72]对GMM模型的成本-服务绩效进行了计算研究，也与R模型、ZF模型、Ng模型和H模型四种加权线性和非线性数学规划 MCIC 模型进行了比较，研究表明GMM模型分类结果表现更好。

1.3 主要研究内容与研究结构

1.3.1 主要研究内容

基于上节研究现状，一方面，在现有的研究中针对于DCI_Closed算法的剪枝策略仍存在局限性，大多只检查1-项集，删掉没有希望的1-项集及以该项集为结点的子树，而缺少针对2-项集或者n-项集（n≥3）的剪枝策略。另一方面，在ABC库存分类中，大都是从物资本身出发，很少考虑到物资之间的关联性，而部分考虑到关联性的研究中大多采用频繁项集挖掘算法来改进库存分类，产生的大量频繁项集可能会出现分类矛盾的情况。

因此本文针对以上问题做的研究工作如下：

（1）优化DCI_Closed算法剪枝策略。将ESCS支持度共现结构应用到DCI_Closed算法中，提出一种改进的闭频繁项集挖掘算法DCI_ESCS。该算法扩展了DCI_Closed算法的剪枝策略，针对2-项集采用了ESCS剪枝策略，首先在扫描数据库的时候构建ESCS结构，储存所有2-项集的支持度信息，然后剪掉所有小于最小支持度阈值的2-项集以及超集，从而减小搜索空间提高挖掘效率。

（2）优化库存分类和库存管理。在库存物资分类时考虑物资之间的关联性，即顾客对于产品需求的关联性。首先用多准则分类方法将库存药品分为A、B、C

三类，然后用改进的 DCI_ESCS 算法在销售数据上进行挖掘，挖掘出具有关联性的有效闭频繁项集模式，根据这些模式对库存分类进行调整，在能满足顾客需要保证企业正常运营的同时也能减少结余库存数量节约库存成本。

1.3.2 研究组织结构

本文分为五个章节，如图 1.1 所示。

第一章为引言。介绍本文的研究背景及意义，总结了频繁项集挖掘算法、高效用项集挖掘算法、闭项集挖掘算法、数据流挖掘算法及多准则 ABC 库存分类法的相关研究现状，还介绍了论文的主要研究内容、论文的组织结构以及研究的创新点。

第二章为相关理论研究。对相关领域的理论展开阐述，主要围绕着关联规则、频繁项集和闭频繁项集、独立需求和相关需求及库存控制方法、ABC 库存管理法几部分展开。

第三章为改进的闭频繁项集挖掘算法。简单介绍了原算法 DCI_Closed 算法的核心思想和实现过程，对改进的 DCI_ESCS 算法的搜索空间、剪枝策略、具体实现过程展开叙述，最后在不同数据集上对 DCI_Closed 算法和 DCI_ESCS 算法的运行时间进行对比分析来验证新改进的 DCI_ESCS 算法的有效性。

第四章为闭频繁项集挖掘算法在 ABC 库存优化问题上的应用。分为问题定义、初始 ABC 库存分类、ABC 库存分类调整优化几个部分。

第五章为总结与展望。对本文的研究内容进行总结归纳，针对仍存在的一些不足之处展开进一步展望。

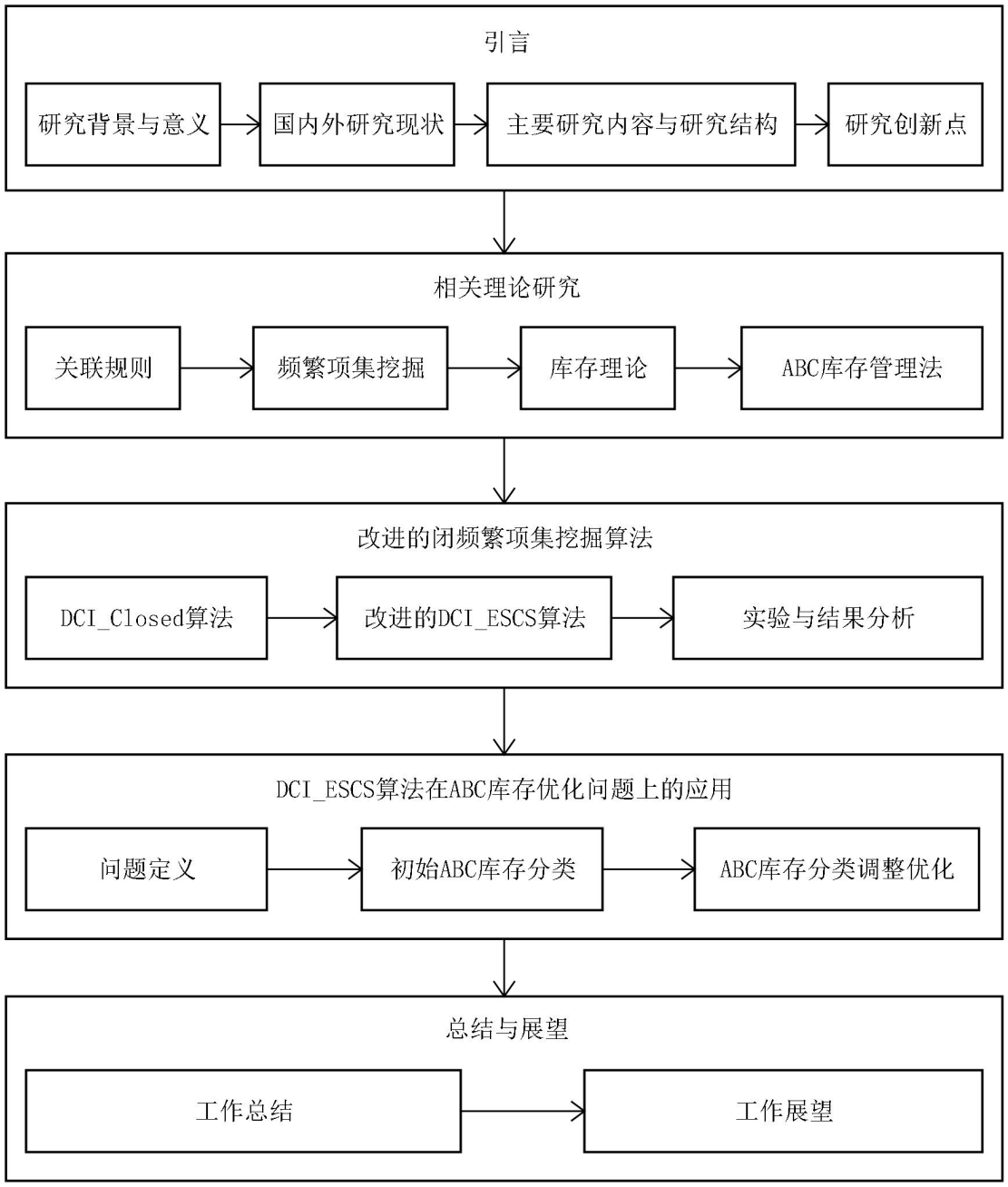


图 1.1 研究框架图

1.4 研究创新点

（1）将 ESCS 结构应用到闭频繁项集挖掘算法领域中，针对于 2-项集提出一种新的 ESCS 剪枝策略，在一个经典的闭频繁项集挖掘算法 DCI_Closed 上进行改进，提出改进的 DCI_ESCS 算法。

（2）将闭频繁项集挖掘算法应用到物资关联性 ABC 库存分类优化问题中，

利用闭项集精简压缩无损的性质删除大量影响 ABC 库存分类调整的冗杂的频繁模式，保证分类调整的唯一性。

1.5 本章小结

本章分别介绍了本文的研究背景与意义，国内外研究现状包括频繁项集挖掘算法、高效用项集挖掘算法、闭项集挖掘算法、数据流挖掘算法、多准则 ABC 库存分类法，主要研究内容与研究结构以及研究的创新点，为本文下一步研究奠定基础。

2 相关理论研究

本文利用闭频繁项集挖掘模式的关联性对库存分类和管理做出优化，因此本章主要介绍本文的理论基础，包括关联规则、频繁项集挖掘、库存理论、ABC 库存管理法和密度峰值聚类模型。其中频繁项集挖掘部分包括频繁项集挖掘和闭频繁项集挖掘，库存理论部分包括独立需求和相关需求、独立需求的库存控制、相关需求的库存控制。

2.1 关联规则

关联规则旨在挖掘出大型数据库中各项之间的关系。设数据库 D 由一组事务$T_j(j=1,2,\cdots,m)$组成，即 D=$\{T_1,T_2,\cdots,T_m\}$，每个 T_j 都是一个项集 I，有且仅有一个事务标识符 TID，由 n 个项组成，即 $I=\{i_1,i_2,\cdots,i_n\}$，长度为 k 的项集被称为 k 项集。关联规则的形式为 $r: i_1 \rightarrow i_2$，其中$i_1 \subset I$，$i_2 \subset I$，$i_1 \neq \emptyset$，$i_2 \neq \emptyset$，$i_1 \cap i_2 = \emptyset$，每个关联规则 r 的确定涉及到两个定义：支持度 support 和置信度 confidence，支持度是指所有事务集中同时包含项 i_1和项 i_2的事务的百分比，置信度是一个条件概率，指在所有事务集中包含项 i_1的同时也包含项 i_2的事务的百分比，公式如下：

$$\text{support}(i_1 \rightarrow i_2) = P(i_1 \cup i_2) \quad (2\text{-}1)$$

$$\text{confidence}(i_1 \rightarrow i_2) = P(i_2|i_1) \quad (2\text{-}2)$$

可以根据具体实例自行设定最小支持度阈值和最小置信度阈值，当一个规则满足支持度不小于最小支持度阈值且置信度不小于最小置信度阈值时，则称这个规则为强关联规则。根据公式（2-2）我们可以推出：

$$\text{confidence}(i_1 \rightarrow i_2) = P(i_2|i_1) = \frac{\text{support}(i_1 U i_2)}{\text{support}(i_1)} = \frac{\text{support\_count}(i_1 U i_2)}{\text{support\_count}(i_1)} \quad (2\text{-}3)$$

通过公式（2-3）得出，规则$i_1 \rightarrow i_2$的置信度可以通过项i_1、项 $i_1 \cup i_2$ 的支持度计数计算得到，同时，支持度可以直接用支持度计数来表示。而通过支持度计数大小与最小支持度阈值的比较直接可以确定一个项集是否为频繁项集，因此关联规则挖掘问题可以简化为频繁项集挖掘问题，具体挖掘过程由以下两步来实

现：

（1）挖掘出数据库中所有的频繁项集。有关频繁项集挖掘的理论部分见章节 2.2。

（2）由频繁项集生成强关联规则。对于每一个频繁项集I_1，生成关联规则$I_2 \rightarrow （I_1 - I_2）$，其中$I_2 \subset I_1$，且该规则的置信度大于等于最小置信度阈值。例如，现有一个频繁项集$\{i_1，i_2，i_4\}$，它的非空子集有$\{i_1，i_2\}$、$\{i_1，i_4\}$、$\{i_2，i_4\}$、$\{i_1\}$、$\{i_2\}$、$\{i_4\}$，由这些非空子集可以得到如下规则：$\{i_1，i_2\} \rightarrow i_4$、$\{i_1，i_4\} \rightarrow i_2$、$\{i_2，i_4\} \rightarrow i_1$、$i_1 \rightarrow \{i_2，i_4\}$、$i_2 \rightarrow \{i_1，i_4\}$、$i_4 \rightarrow \{i_1，i_2\}$。根据公式（2-3）分别计算这些规则的置信度，满足最小置信度阈值条件的即为强关联规则。

2.2 频繁项集挖掘

频繁项集挖掘是数据挖掘中关联规则技术的重要组成部分，能够发现一组事务中常常同时出现的项集，这些项集间具有很高的关联性，这些关联性信息具有隐秘、高价值的特点。在频繁项集挖掘的基础上，又进一步衍生出闭频繁项集的概念。本节针对频繁项集挖掘和闭频繁项集挖掘的基本概念和相关定义性质展开阐述。

2.1.1 频繁项集挖掘

表 2.1 展示的是一个数据库实例，它由 4 条事务(T_1, T_2, T_3, T_4)和 4 个项(A,B,C,D)组成。下面将详细介绍在频繁项集挖掘问题中会用到的定义和性质。

表 2.1 数据库实例

TID	Transaction
T_1	A C D
T_2	A B C D
T_3	B D
T_4	B C D

定义 1 支持度(Support Count，SC)。项集在事务数据库出现的频度，也称作项集的支持度计数。例如，在表 2.1 中，项 A 在 T_1、T_2 中各出现一次，因此 $SC(A) = 2$，同理，项 B 在 T_2、T_3、T_4 中各出现一次，$SC(B) = 3$。

定义 2 频繁项集。当一个项集的支持度不小于最小支持度阈值，则称此项集为频繁项集。例如，设最小支持度阈值为 2，，因此项集 AD 和项集 BD 都是频繁项集。

2.1.2 闭频繁项集挖掘

闭项集的概念是基于以下两个函数提出来的：

$$f(T) = \{i \in I | \forall t \in T, i \in t\} \quad (2\text{-}4)$$

$$g(I) = \{t \in D | \forall i \in I, i \in t\} \quad (2\text{-}5)$$

其中函数 f 返回所有事务中共同包含的项集，函数 g 返回包含项集 i 的所有事务

定义 3 闭项集。当一个项集称之为闭项集，当且仅当满足：

$$c(I) = f(g(I)) = f \circ g(I) = I \quad (2\text{-}6)$$

其中混合函数 $f \circ g(I)$ 也被称作伽罗瓦操作或者闭包操作。

例如，在表 2.1 中的事务集中，$c(ACD) = f \circ g(ACD) = f(g(ACD)) = f(T_1, T_2) = ACD$，因此项集 ACD 是一个闭项集。

定理 1 对项集 X 和项集 Y，如果满足 $X \subset Y$ 以及 $SC(X) = SC(Y)(|g(X)| = |g(Y)|)$，则 X 的闭包和 Y 的闭包相同，即 $c(X) = c(Y)$。

证明：如果 $X \subset Y$，则有 $g(Y) \subseteq g(X)$。又已知 $SC(X) = SC(Y)$，即可以推

出$|g(Y)| = |g(X)| \Rightarrow g(X) = g(Y) \Rightarrow f(g(X)) = f(g(Y)) \Rightarrow c(X) = c(Y)$

因此，给定一个生成器 X，如果找到一个包含 X 的超集 Y，并且 Y 和 X 的支持度是相同的，可以得出 $c(X) = c(Y)$。对此，可以删除生成器 X，无需实际计算它的闭包。然而，这种生成器重复检查策略在时间和空间上都很昂贵。在时间上，因为可能需要在大量的闭项集中搜索每个生成器的组成，会耗费大量时间；在空间上，为了能够有效运行算法，需要将所有的闭项集保存在主内存中，空间占用也很大。为了降低这种成本，可以将闭项集存储在紧凑的前缀树结构中，通过一个或多个散列项进行索引。

定理 2 对于一个项集 X 和一个项 i，如果满足 $g(X) \subseteq g(i)$，则项 i 是 X 的一个闭包，即 $i \in c(X)$。

证明：已知 $g(X \cup i) = g(X) \cap g(i)$，由 $g(X) \subseteq g(i)$可以推出 $g(X \cup i) = g(X) \cap g(i) = g(X)$。则有 $f(g(X \cup i)) = f(g(X)) \Rightarrow c(X \cup i) = c(X) \Rightarrow i \in c(X)$。

定义 4 闭频繁项集。如果项集 X 同时满足在数据库中不存在 X 的超集 Y 且与 X 的支持度相同且 $SC(X) \geq minsup$，则称 X 为闭频繁项集。

定义 5 ESCS(Estimated Support Co-occurrence Structure，支持度共现结构)。类似于 EUCS，ESCS 中的数据也是以三元组(a,b,c)的形式存储的，并有 SC({a,b})=c，即 ESCS 是用来存储数据库中所有 2-项集的支持度计数的，以表 2.1 为例，建立的 ESCS 见表 2.2。

表 2.2 ESCS

Item	A	B	C
B	1		
C	2	2	
D	2	3	3

定义 6 等价类。闭包运算在频繁项集格上定义了一系列的等价类:如果两个项集具有相同的闭包，即它们有相同的支持度，则它们属于同一个等价类。

由定义 4 可知，对于一个项集 i,如果不存在具有和 i 具有相同支持的超集，则项集 i 是闭合的。因此，一个闭项集是同一个等价类的最大项集，挖掘每个等

价类的所有最大元素等同于挖掘所有的闭项集。从图 2.1 中我们可以看到，具有相同闭包的项集被分组在同一个等价类中，每个等价类包含着具有相同支持度的项集，而闭项集是其中的最大项集。例如项集 A、AC、AD、ACD 属于同一个等价类，右上角显示每个项集的支持度，它们具有相同的闭包和支持度，其中的最大项集 ACD 则是这个等价类中的闭项集。同理 ABCD、BCD、BD、CD、D 是它们所在等价类中的最大项集，作为闭项集被挖掘出来。此外，假设最小支持度阈值 minsup=1,闭频繁项集的数量(六个)明显低于频繁项集的数量(十五个)，而那些非闭合的频繁项集能完整地保存在频繁项集格结构中。这也验证了闭项集挖掘的有效性和高效性，比起单纯挖掘频繁项集，挖掘闭频繁项集可以大大缩短挖掘时间，节省占用空间，并且不会损失任何信息。因此闭频繁项集是频繁项集的一种精简压缩模式。

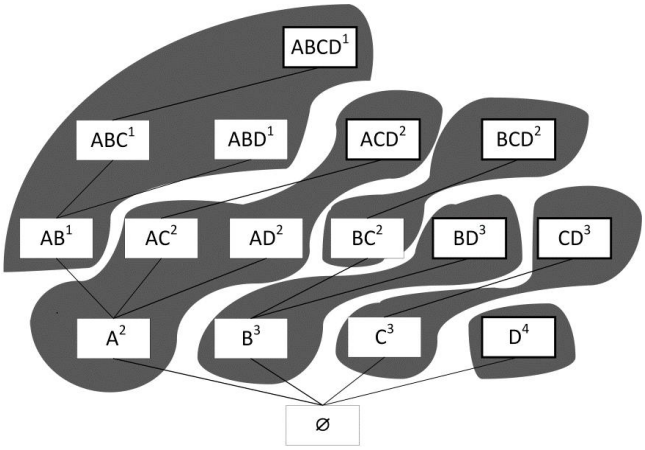


图 2.1 包含闭项集的频繁项集格

2.3 库存理论

库存设置的初衷是由于人们无法对外部市场未来的需求变化做出预测，为了保证生产的连续性和企业的正常运转不得已采取的一种对策，企业会提前采购用于未来交易的一些物资暂时存储在企业库存中，当库存不足时也会采取一系列订货策略及时地对库存进行补充。同时也正因为外部市场需求的变化性和不确定性，企业很难对库存物资数量有精准的把控，因此会造成一些库存物资的冗余，库存

管理理论由此应运而生。库存管理理论是由库存概念形成的一套管理体系，不仅对企业管理有着重要的作用，并且还可以加强库存管理，优化订货策略和库存控制方法。一方面可以提高企业库存流动的效率，避免库存积压过多造成高昂的库存成本；另一方面能够第一时间依据用户的需求提供相应的物资，减少用户需求订单与企业供应环节的对接时间。一般来说，一个企业库存管理质量的高低由以下几个因素来决定：

（1）平均库存价值。指库存内所有物资在一定时间范围内占用的金额，比如以一个月为时间周期：

$$年平均库存价值 = \frac{一年内全部库存物品的价值之和}{12} \tag{2-7}$$

（2）可供给时间。指现有的库存能够满足外部市场多长时间的需要。

（3）库存周转率。

$$库存周转率 = \frac{年销售额}{年平均库存价值} \times 100\% \tag{2-8}$$

库存周转率又分为成品库存周转率、在制品库存周转率和原材料库存周转率。其中：

$$成品库存周转率 = \frac{年销售额}{年平均库存价值} \times 100\% \tag{2-9}$$

$$在制品库存周转率 = \frac{生产产值}{在制品平均库存价值} \times 100\% \tag{2-10}$$

$$原材料库存周转率 = \frac{原材料消耗额}{原材料库存价值} \times 100\% \tag{2-11}$$

（4）库存服务水平。

$$库存服务水平 = \frac{供应量或销售量}{供应量或销售量+短缺量} \times 100\% \tag{2-12}$$

2.3.1 独立需求与相关需求

基于已有的理论研究发现，单（多）周期需求通常是依据库存中物品的性质、人们对物品需求依赖程度与反复性等指标加以定义区分。单周期需求，人们通常称作一次性订货，该需求物资具有时效性、需求循环期限短的特点，人们对于这一类物品只有在某一特定时期才有需求，过了这段时间这类物品就会失去应有的价值，因此在现实生活中通常不会反复购买，订货周期比较长。比如像期刊、元宵、生冷海鲜易腐烂的物品等等，这些物品都属于单周期需求的物品。多周期需

求在我们日常生活中更为常见一些，定义为在一段相当长的时期内需要对某一类物资进行反复的、不间断的购买补充，比如人们日常生活中米、面等食品，消费完之后要及时采购，因此需要及时补充这类物品的库存。

多周期需求按需求的性质分成了独立需求和相关需求两大类。对于企业来说，如果一类物资的需求信息是完全受制于外部市场条件，其他物资不会对其产生任何影响，是完全独立的，则称这类需求为独立需求（Independent Demand）。独立需求物资通常是指一个完整的制成品或最终产品，比如一辆汽车、一台电脑等，其在当前时间周期内的需求信息是通过上一时间周期的需求信息预测得出的。如果企业物资的需求受到其他物资需求影响或由其他物资决定，则称这类需求为非独立需求或相关需求（Dependent Demand），比如组成一个完整产品的部分零部件、半成品等，其需求信息是根据数学公式计算出来的。因此，对于一个相对比较独立的企业的负责人来说，其一个完整产品需求的数量和时间很难预先精准确定，只能通过以往的销售数据进行预测，而对于一个完整产品的部分原料，通常是依据产品的组成关系和一定的生产比例关系来精准计算得到的。

以上两种需求的定义、性质有所不同，其库存控制方法同样也存在一定的差异性。然而无论是对于哪种需求来说，其库存控制都要以满足用户订单需求、避免库存囤积、降低库存成本为目标来考虑，需要进行以下几个环节：①确定订货数量；②确定订货点，即订货的时间；③确定库存的检查周期。接下来针对于两种不同需求的常用的库存控制方法进行简单表述。

2.3.2 独立需求的库存控制

对于独立需求物资来说，往往使用订货点控制策略，常见的通常有如下几种：

（1）(Q,R)策略

(Q,R)策略即连续性检查的固定订货量、固定订货点策略，其核心内容通常是持续性对库存进行核查，一旦发现库存量 Q 小于订货点水平 R 的时候，立即订货来补充库存，并且确保每次下单数量 Q 为一个固定量。此策略通常用于市场需求旺盛、需求量变化大、缺货成本较高的物资。

（2）(R,S)策略

(R,S)策略即连续性检查的固定订货点、最大库存策略，其核心内容通常和

（Q, R）策略一样，都需要对库存物资进行持续性核查，当发现库存量小于订货点水平 R 时，立即下单来将库存物资数量补充到最高点 S，比如假设对库存量进行核查,其库存数量为 I，那么需要下单的数量则为 S-I。(R,S)策略与(Q,R)策略不同之处是(Q,R)策略订货量是固定的，而该策略下单的数量是根据现有的库存数量不断变动的。

（3）(t,S)策略

(t,S)策略即周期性检查策略，其核心内容通常是以某段时间 t 为一个周期定期核验库存状态，进行下单使得实际库存数量与库存量最高点 S 保持一致，比如假设对库存数量进行核查时，其库存数量为 I，那么需要下单的数量则为 S-I，在下一个周期再次核查并补充后，此刻原仓库物资的数量为 I_1，需要补充的物资数量为$(S-I_1)$，以此类推，不断定期核查库存并与库存数量最高点 S 保持一致。(t,S)策略不需要关注订货的时间，只需要明确库存物资的固定检查周期 t 和库存数量最高点 S，其通常应用于对物资需求数量较少、价值较低的市场。

（4）(t,R,S)策略

(t,R,S)策略即综合库存策略，是在(R,S)策略和(t,S)策略的基础上发展而成的，其核心内容通常是设置一个固定的检查周期 t、固定订货点水平 R、最大库存量 S。每隔一个检验周期 t 对库存数量进行清点，如果发现库存数量小于订货点水平 R，立即发出订单使库存数量保持到最高点。

2.3.3 相关需求的库存控制

相关需求的库存控制一般采用的是物资需求计划策略。物资需求计划（Material Requirement Planning，MRP）是企业集中订货的前提和基础，是根据客户需求订单和外部市场需求预测具体制定产品的生产计划,然后根据产品生产进度计划、组成产品各物料结构表和相应库存状况，依据平台系统监测到客户对各物料的需求相关信息,从而确定对于所需物料的生产进程和下单时间的一种高效方法。其一般的计算步骤如图 2.2 所示。

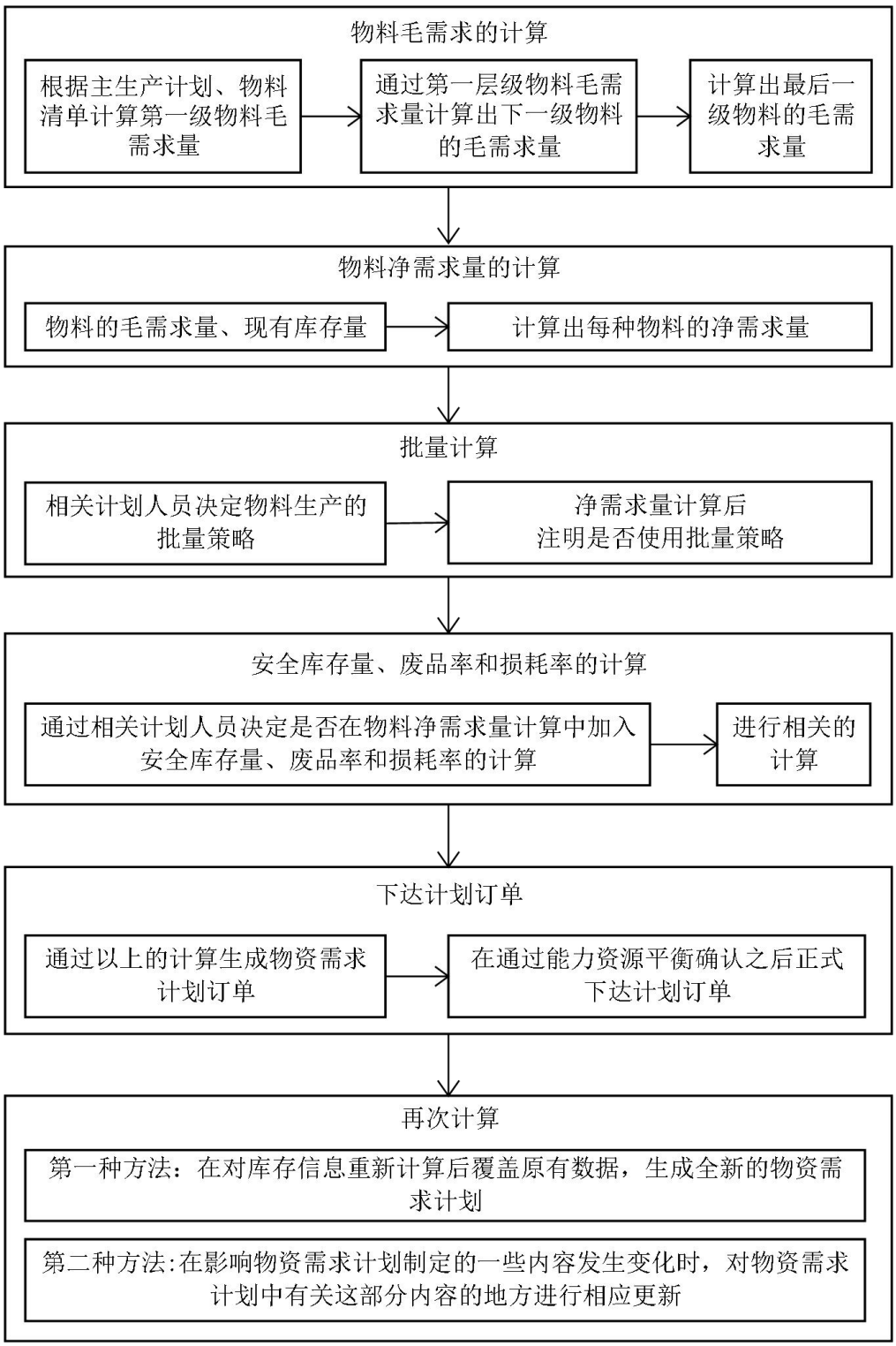


图 2.2 物资需求计划计算步骤

相关需求物资的关联性能够通过一个产品各部件的组成关系较为直接地体现出来，而实际生活中各独立需求物资之间也存在着隐藏的关联性，因此本文通

过挖掘闭频繁模式发现其中的关联信息。

2.4 ABC 库存管理法

一个企业每年需要采购大量的物资来满足自身生产和销售的需要，如果不对这些物资进行分类会导致管理需要耗费大量的人力、物力和时间，造成昂贵的管理成本。因此对物资进行合理的分类工作至关重要，针对不同类别的物资采用不同的管理模式和库存控制方法。比较常用的是 ABC 库存分类管理法。

ABC 库存分类管理法（Activity Based Classification，ABC）又叫帕雷托分析法，依据库存单位的性质和使用金额的大小将库存物资分成 A、B、C 三类，其重要程度递减，最后对于不同类别的物资采用不同的库存管理模式：

（1）A 类物资在库存数量上仅占 5%-15%，但是占用了库存资金的 60%-80%。因此 A 类物资是关键的少数，需要加强对这类物资的管理和控制，企业一般是采用连续控制的方式进行管理，经常检查其库存量避免缺货的情况，一旦库存数量小于设定的订货数量，马上进行采购来保证足够的库存量。同时，对于这类物资也不能出现库存积压的情况，在保证现有库存属于安全库存的条件下，通过小批量、多批次的方式按需订货，希望能最大程度地减少库存数量，提高资金周转量，减少库存管理成本。

（2）B 类物资在库存数量上占 15%-25%，占用了库存资金的 15%-25%。对于 B 类物资的库存控制不用和 A 类物资一样严格，但是也不能过于宽松。对于 B 类物资，企业一般是采用定期控制的方式进行管理，定期检查其库存量并按照经济批量订货方式补充库存。对于某些物资允许适量缺货的情况。

（3）C 类物资在库存数量上占 60%-80%，占用了库存资金的 5%-15%,属于不太重要的物资，因此对其控制方法可以粗放一些。通常采用定量订货控制方法，大批量集中订货，对于订货的数量不必严格控制，减少订货次数。

表 2.3 总结了 ABC 三类库存的性质及控制方法。

表 2.3 ABC 三类库存性质及控制方法

类别	A	B	C
数量比例	5%-15%	15%-25%	60%-80%
资金占用比例	60%-80%	15%-25%	5%-15%
重要性	十分重要	一般重要	不重要
控制方式	连续控制	定期控制	定量控制
库存检查周期	经常检查	一般检查	按年度或季度检查
是否允许库存积压	不允许	适当允许	允许
是否允许缺货	不允许	适当允许	允许

ABC 分类管理法可以有效区分企业库存中 “重要的少数”和“不重要的多数”，这样企业能够将管理的重点对象放在那些价值最高、对库存影响最大的少数物资上，有效地节约了库存管理成本。然而，ABC 分类管理法也存在不足之处，该方法的分类标准单一，物资的重要性完全由资金占用量决定。但是 B 类甚至 C 类中一些物资虽然价值不高，但是对企业至关重要或者在市场上比较难得到，属于比较短缺的物资，如果缺货将会严重影响企业的经营。因此，为了弥补这一不足，在传统 ABC 分类法的基础上，将库存物资定性分析和定量分析相结合，从而更准确地对库存物资进行分类管理。此外，综合考虑提前期、成本、价格和关键性等多种要素的影响，多准则 ABC 库存分类方法(Multi-criteria ABC Inventory Classification，简称 MCIC)被提出，相关研究情况见 1.2.5 节。

2.5 本章小结

本章介绍了与本文研究内容有关的理论知识，分别介绍了关联规则的相关理论、频繁项集挖掘包括频繁项集和闭频繁项集的定义和性质、有关于独立需求和相关需求定义及库存控制方法的库存理论知识、ABC 库存管理法相关理论，为后文实验奠定了理论基础。

3 改进的闭频繁项集挖掘算法

3.1 DCI_Closed 算法

DCI_Closed 算法是一个经典的闭频繁项集挖掘算法，是本章改进的闭频繁项集挖掘算法 DCI_ESCS 的基础。

DCI_Closed 算法输入三个参数:一个闭项集 CLOSED_SET 和两个项集：前置集 PRE_SET 和后置集 POST_SET，输出包含 CLOSED_SET 的所有非重复的闭频繁项集。该算法的中心思想是通过用后置集 POST_SET 集合中的所有元素来深入探索从闭项集 CLOSED_SET 获得的每个有效的新生成器,从而扩展闭项集 CLOSED_SET。算法首先对数据库 D 进行扫描以确定频繁 1-项集 $i \subseteq FIs$,并构建包含 tid-lists 结构的垂直数据集 VD（vertical database）。然后建立起递归调用过程，通过用 POST_SET 中的各项扩展闭项集 CLOSED_SET 来构建所有可能的生成器。注意在这个过程中不频繁的和重复的生成器(非保序的生成器)作为无效生成器被丢弃。在之后的递归调用中，将不再考虑由 i∈POST_SET 获取的那些无效生成器，只扩展有效的生成器来计算它们的闭包。值得注意的是，每个生成器 $newgen \leftarrow CLOSED\_SET \cup i$ 是根据保序属性严格扩展的,即对于所有项 $j \in POST\_SET$ 集合满足 $i \prec j$。不属于生成器闭包 c(new gen)的所有项 j 都包含在新的 POST_SET 集合中，用于下一次递归调用。在这个过程最后，会生成一个新的闭项集 $CLOSE\_SETNEW \leftarrow c(newgen)$。根据这个新的闭项集，通过递归调用上述过程，可以构建新的生成器和相应的闭项集集合。

DCI_Closed 算法采取了大量优化措施，以降低搜索和闭包计算的成本，避免重复计算。

（1）DCI_Closed 采用了垂直数据库，将数据集使用垂直位图存储在主内存中。在对数据集进行两次连续扫描后，一个位图矩阵 $D_{M\times N}$ 被存储在主存中。当且仅当第 j 个事务包含第 i 个频繁的单个项时，D（i，j）位被设置为 1。因此，矩阵的第 i 行代表了第 i 项的 tid-list。这种结构的表示方法能够使算法与数据更好地结合，当访问矩阵中某一行时，就能快速找到该项所在的事务集合并计算出支持度信息，事务集合的大小即为该项的支持度大小。

（2）为了确定项集 X 是闭项集，必须将 X 所在的事务集合 g(X)与 X 的前置集 PRE_SET（后置集 POST_SET）中包含的所有项 j 所在的事务集合 g（j）进行比较，项 j 即根据字典顺序，在 X 中包含的所有项之前（之后）的项。通过访问前置集来检查重复的生成器，访问后置集以计算闭包。特别是对于所有 $j \in PRE\_SET \cup POST\_SET$，我们已经知道 $g(X) \notin g(j)$，否则这些项 j 一定包含在 X 中。因此，由于 g（X）已经与所有 g（j）进行了比较，对于 X 的前置集（后置集）中包含的所有项 j，DCI_Closed 算法保存了关于 g（j）和 g（X）之间每次比较的一些重要信息。当必须利用 g（j）来寻找更多包含或者是扩展 X 的闭项集时，这些信息可以大大降低寻找的成本。

虽然 DCI_Closed 算法挖掘闭频繁项集的效率已经很出众了，但是它在剪枝策略上仍存在局限性，大多针对 1-项集进行剪枝，删去没有希望的 1-项集以及超集，对于 2-项集或者是 n-项集并没有相应的剪枝策略，在挖掘效率上还有待提升。

3.2 改进的 DCI_ESCS 算法

针对 DCI_Closed 算法的不足之处，本节将文献[20]提出的储存所有 2-项集支持度信息的 ESCS 结构融入到 DCI_Closed 算法中，对 2-项集进行剪枝，剪掉以支持度小于最小支持度阈值的 2-项集为根结点的子树。通过新的 ESCS 结构储存支持度信息以及采用新的剪枝策略可以减少扫描数据库的次数，避免重复操作，在挖掘过程中及时删去没有希望的非闭频繁项集，从而一定程度上提升了挖掘效率。

3.2.1 搜索空间

该算法的搜索空间是一个集合枚举树的树状结构，并且使用闭包攀爬的方式进行闭频繁项集的挖掘，以表 2.1 数据库为例，得到包含闭项集的频繁项集格如图 3.1 所示。集合枚举树的构造过程如下：

①创建一个空的根节点 ∅ ；

②创建包含所有项的根的 4 个子节点：A、B、C、D；

③对每个子节点创建表示 2-项集的子节点，直至产生所有表示 2-项集的子节点：AB、AC、AD、BC、BD、CD；

④同理，对所有表示 2-项集的子节点创建表示 3-项集的子节点，直至产生所有表示 3-项集的子节点：ABC、ABD、ACD、BCD；

⑤重复上述过程，创建出表示所有项集的子节点。

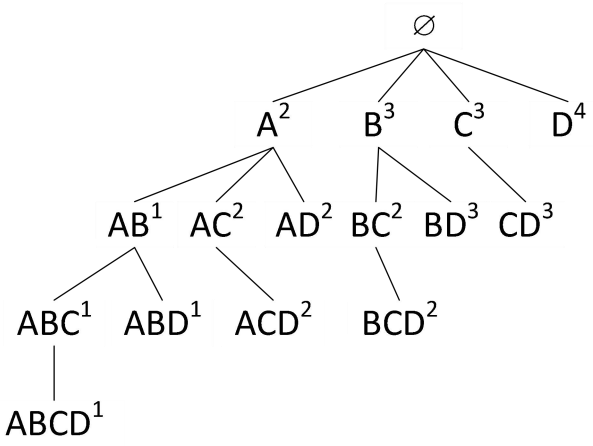


图 3.1 集合枚举树

3.2.2 剪枝策略

剪枝策略是影响算法挖掘效率的主要因素之一，它的重要性体现在：如果某个项集满足剪枝条件，则以该项集为根结点的整棵子树都应该被剪掉，因此更有效的剪枝策略能够及时剪去不满足阈值的项集，从而可以减小搜索空间和算法的运行时间。

本文提出的改进 DCI_ESCS 算法改变了单一的剪枝策略，在原有剪枝策略上进行扩展，对 1-项集和 2-项集分别采用了不同的剪枝策略：

策略 1 SC 剪枝。SC 剪枝被应用于 1-项集，如果对 1-项集 X 有 $SC(X)<minsup$，则项集 X 不是频繁项集亦不是闭频繁项集，且 X 的扩展项集也一定都不是频繁项集亦不是闭频繁项集。

策略 2 ESCS 剪枝。ESCS 剪枝被用于 2-项集，如果对 2-项集 X 有 $SC(X)<minsup$，则项集 X 不是频繁项集亦不是闭频繁项集，且 X 的所有扩展

项集也都不是频繁项集亦不是闭频繁项集。

以表 2.1 数据库为例，假定最小支持度阈值为 3，原 DCI_Closed 算法与改进的 DCI_ESCS 算法的剪枝范围见图 3.2 和图 3.3。从图中可以看出，改进的 DCI_ESCS 算法的剪枝范围更大，能够及时发现没有希望的 2-项集并且剪去以该项集为根结点的子树。如果只采用原算法的 1-项集剪枝策略，只有以 A 为结点的子树被剪去，对于结点 B、C、D 来说算法还是会一一生成它们所有的扩展项集：BC、BD、BCD、CD。而对于改进的 DCI_ESCS 算法来说，不仅剪去了 A 的子树，针对于 2-项集的 ESCS 剪枝策略也剪去了项 BC 为结点的子树，不再生成扩展项集 BCD。能够有效地减少搜索空间，提升算法的挖掘效率。并且如果一个事务数据库事务数和项数越多、越复杂，那么针对于某一项生成的扩展项集越多，通过 ESCS 剪枝策略剪去的子树中包含的项就越多，算法效率提升越明显。

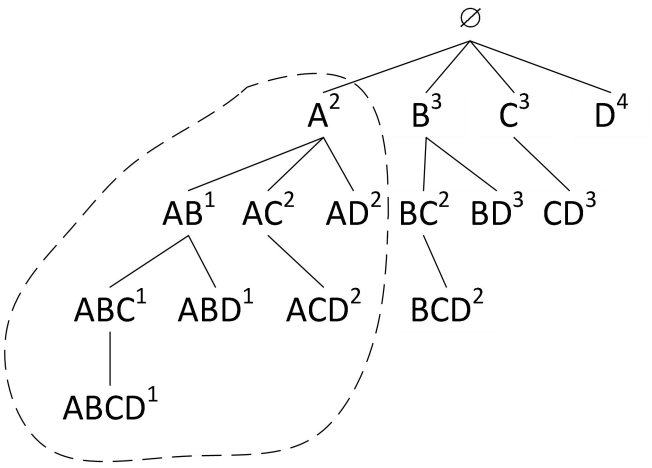


图 3.2 原算法剪枝范围

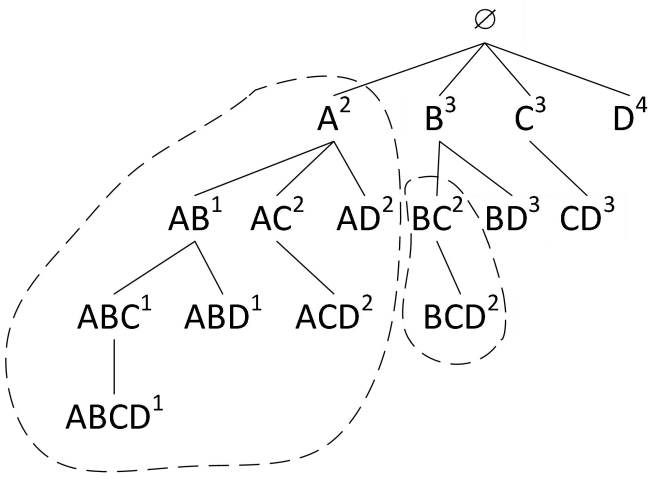


图 3.3 改进算法剪枝范围

3.2.3 DCI_ESCS 算法描述

DCI_ESCS 算法的主程序见 Algorithm 3.1，输入的是闭项集 CLOSED_SET，初始为⌀ 、用来构建生成器的后置集 POSE_SET 和用来检查生成器是否重复的前置集 PRE_SET，输出的是所有带有支持度信息的闭频繁项集。其中主程序中生成器重复检查过程 is_dup 见 Algorithm 3.2 ，输入的是生成器 newgen 和前置集 PRE_SET，输出的是 TRUE 或 FALSE，TRUE 代表该生成器重复，FALSE 代表不重复。支持度共现结构构建及支持度提取过程 calcIJ_FI 见 Algorithm 3.3，输入的是 2-项集（i,j），输出的是 ESCS 结构中对应 i 和 j 的支持度。

Algorithm 3.1 DCI-ESCS 主程序

```
输入：闭项集 CLOSED_SET
      前置集 PRE_SET 和后置集 POST_SET
输出：闭频繁项集及支持度
1  初始数据转换成垂直数据库格式
2  按项升序的顺序构建初始 POST_SET
3  for each i in POST_SET do
4      if SC(i) ≥ minsup then                    #1-项集剪枝
5          newgen ← CLOSED_SET ∪ i               #构建生成器
6          if is_dup(newgen , PRE_SET) = FALSE then          #生成器重复检查
7              CLOSED_SET_New ← newgen
8              POST_SET_New← ⌀
9              for each j in POST_SET 且 i ≺ j do
10                 if calclJ_FI(i,j)≥ minsup then            #2-项集 ESCS 剪枝
```

```
11                      if g(newgen) ⊆ g(j) then          #j 是 newgen 的超集
12                          CLOSED_SET_New ← CLOSED_SET_New ∪ j
13                      else
14                          POST_SET_New ← POST_SET_New ∪ j
15                      end if
16                  end if
17              end for
18              输出闭频繁项集 CLOSED_SET_New 及其支持度
19              DCI_ESCS(CLOSED_SET_New, PRE_SET, POST_SET_New)
20              PRE_SET ← PRE_SET ∪ i
21          end if
22      end if
23 end for
```

Algorithm 3.2 is_dup（生成器重复检查过程）

```
输入：newgen
      PRE_SET
输出：TRUE or FALSE
1   for each j in PRE_SET do
2       if g(new_gen)⊆g(j) then
3           return TRUE
4       end if
5   end for
6   return FALSE
```

Algorithm 3.3 calcIJ_FI（支持度共现结构构建及支持度提取过程）

```
输入：项 i 和项 j
输出：ESCS 结构中对应 i 和 j 的支持度
1   mapESCS = new HashMap<Integer, Map<Integer, Long>>()
2   Map<Integer, Long> mapESCS_Item = mapESCS.get(i)
3   if mapESCS_Item ≠ ∅ do
4       supp = mapFMAP_Item.get(j)
5       if supp ≠ ∅ do
6           return supp
7       end if
8   end if
9   cnt=0
10  tidset1 = database.get(i)
11  tidset2 = database.get(j)
12  for each tid in tidset1 do
13      if tid ⊆ tidset2 do
14          cnt++
15      end if
16  end for
```

```
17  if mapESCS_Item ≠ ∅ do
18      mapFMAPItem .put(j, cnt)
19  else
20      mapFMAPItem = new HashMap<Integer, Long>();
21      mapFMAPItem .put(j, cnt)
22  end if
23  mapFMAP.put(i, mapFMAPItem)
24  return cnt
```

初始数据是传统的事务数据库，为了更适用于新算法，需要将其转换成垂直数据库的格式（Algorithm 3.1 第 1 行），垂直数据库表示为项-事务集合的格式，即项和该项所在的事务的集合。以表 2.1 中的数据为例，转换后的垂直数据库如表 3.1 所示,其中 0 表示第一条事务，1 表示第二条事务，以此类推。

表 3.1 垂直数据库

Item	TIDset
A	[0,1]
B	[1,2,3]
C	[0,1,3]
D	[0,1,2,3]

构建初始 POST_SET（Algorithm 3.1 第 2 行）包含两个过程：首先扫描一次数据库，计算 1-项集的支持度即 TIDset 的大小进行剪枝，在 POST_SET 中保留不小于最小支持度阈值的项，此处设 $minsup = 2$;然后将 POST_SET 中的项按照支持度升序的顺序进行排序。本章节所有计算都是以表 2.1 数据为例，后面不再阐述。由于 $SC(A) = 2 \geq minsup = 2$； $SC(B) = 3 \geq minsup$； $SC(C) = 3 \geq minsup$；$SC(C) = 4 \geq minsup$，四个项都不满足剪枝条件，予以保留。按照 SC 升序的顺序得到的初始 $POST\_SET = [A,B,C,D]$。

该算法采用深度优先的挖掘方式，首先对项集 A 进行挖掘，直至挖掘出所有以 A 结点为根结点的满足最小支持度阈值的闭频繁项集。然后再分别对项集 B、C、D 进行挖掘，直至挖掘出所有符合条件的闭频繁项集。为了更直观地展现算法挖掘过程，下面展现以 1-项集 A 为根结点的计算过程。

对于项集 A，有 $SC(A) = 2 \geq 2$，满足最小阈值条件，构建生成器 $newgen = (CLOSED\_SET) \cup (i) = \emptyset \cup A = [A]$（Algorithm 3.1 第 5 行）。Algorithm 3.1

第 6 行是对构建的生成器进行重复检查，详细见 Algorithm 3.2。此时 $PRE\_SET = \emptyset$，返回 FALSE，执行 if 条件语句，得到 $CLOSED\_SET_{New} = newgen = [A]$。Algorithm 3.1 第 9-10 行构建了所有满足 $j \in POST\_SET$ 且 $i \prec j$ 的 2-项集支持度共现 ESCS 结构，并对小于最小支持度阈值的 2-项集剪枝，详细见 Algorithm 3.3。经过计算构建的 ESCS 如表 3.2 所示：

表 3.2 i=A 时的 ESCS 结构

Item	A	B	C
B	1		
C	2		
D	2		

对于 2-项集 AB，$SC(AB) = 1 < minsup = 2$，满足剪枝条件被删去；对于 2-项集 AC，$SC(AC) = 2 \geq minsup = 2$，$g(newgen) = g(A) = [0,1]$，$g(j) = g(C) = [0,1,3]$，满足 $g(newgen) \subseteq g(j)$，根据定理 2 和定义 4，项 C 是 A 的一个闭包，且 $SC(A) = SC(AC)$，因此项集 A 不是一个闭频繁项集，此时 $CLOSED\_SET_{New} = [A \cup C] = [AC]$；对于 2-项集 AD，同理，$SC(AD) = 2 \geq minsup = 2$，$g(newgen) = g(A) = [0,1]$，$g(j) = g(D) = [0,1,2,3]$，满足 $g(newgen) \subseteq g(j)$，根据定理 2 和定义 4，项 D 也是 A 的一个闭包，且 $SC(A) = SC(AD)$，此时 $CLOSED\_SET_{New} = [AC \cup D] = [ACD]$。注意由于 $SC(A) = SC(AC) = SC(AD) = SC(ACD)$，项集 A、AC、AD 都存在超集与其支持度相同，因此它们都不是闭频繁项集，而项集 ACD 不存在任何超集与其支持度相同，并且支持度满足最小支持度阈值条件，所以项集 ACD 是一个闭频繁项集，将其输出。接下来对 $i = B$、C、D 进行同样的挖掘过程，直至发现所有闭频繁项集。

3.3 实验与结果分析

本节实验是通过比较改进算法 DCI_ESCS 和原算法 DCI_Closed 分别在五个数据集上不同最小支持度阈值下的挖掘时间来分析改进 DCI_ESCS 算法的性能。实验环境：（1）硬件：i7-9700K，256G 内存，1T 固态硬盘的惠普台式电脑；（2）软件： Ubuntu、Java。

3.3.1 实验数据集

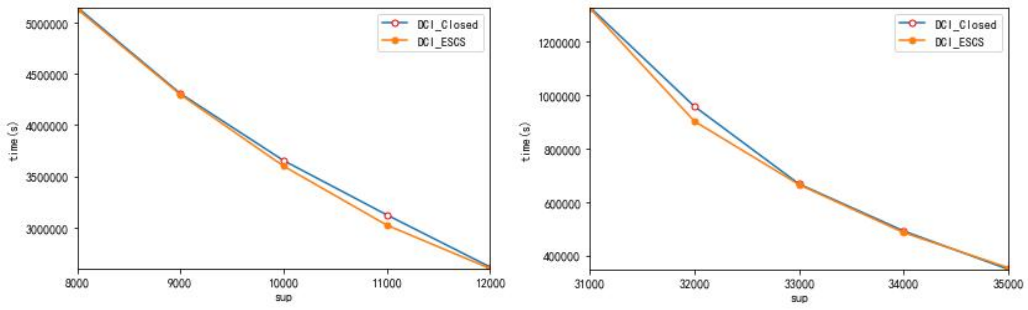
本次实验使用的数据集均是从 SPMF（见网址：http://www.philippe-fournier-viger.com/spmf）公开资源库中下载的标准数据集，这些数据集的特征归纳见表 3.3。

表 3.3 数据集特征

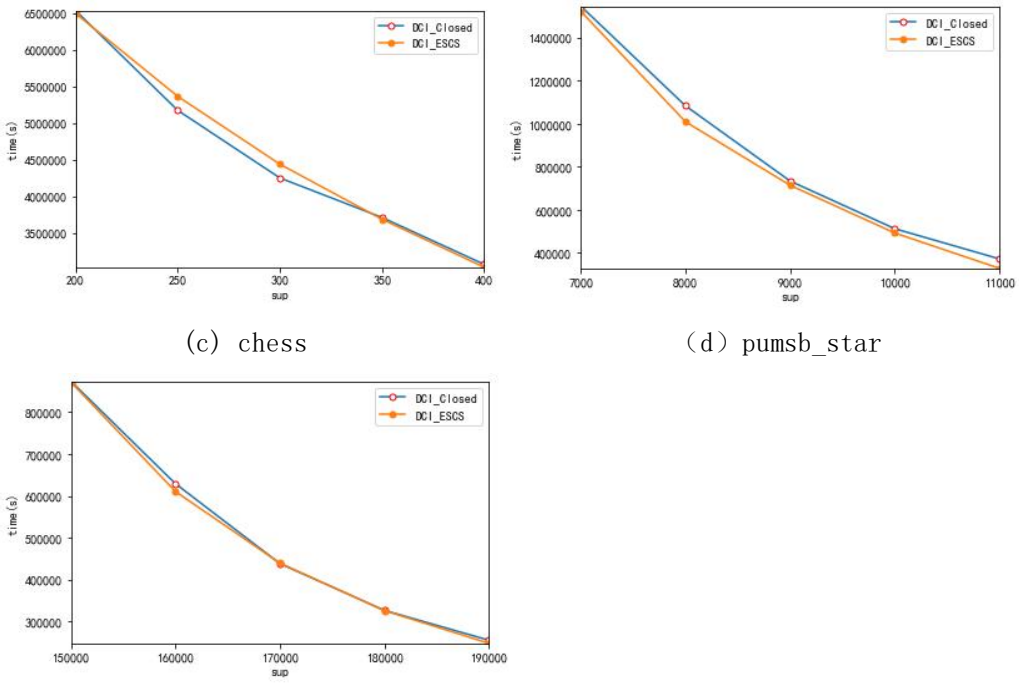
数据集	事务数	项集数	平均事务长度	性质
connect	67557	130	43	稠密，事务长
pumsb	49046	7117	74	稠密，事务非常长
chess	3196	76	37	稀疏，事务短
pumsb_star	49046	7117	50	稠密，事务较长
accidents	340183	468	33.8	较稠密，事务较长

3.3.2 运行时间对比分析

为了验证改进的 DCI_ESCS 算法的有效性和可行性，本节实验比较了 DCI_ESCS 算法和 DCI_Closed 算法在不同最小支持度阈值下的项集挖掘时间。为避免实验结果的偶然性，实验过程中对于 5 个数据集上的对照实验，都取 5 次结果的平均值。根据改进前后算法的运行时间随最小支持度阈值变化绘制折线图，如图 3.4 所示



(a) connect　　(b) pumsb



(c) chess　　（d）pumsb_star

（e）accidents

图 3.4 不同数据集上的运行时间

从图中可以看出，改进的 DCI_ESCS 算法在数据集 connect、pumsb、pumsb_star 和 accidents 上均有一定程度的改进。随着最小支持度阈值的增大，两个算法挖掘出的闭频繁项集的数量减少，挖掘时间也在减少，此外经过验证改进的DCI_ESCS 算法与DCI_Closed 算法在同一个数据集同一个支持度阈值下挖掘出的闭频繁项集数量是一致的，这说明本文提出的 DCI_ESCS 算法是可行的。经过进一步计算，得到改进的 DCI_ESCS 相对 DCI_Closed 算法挖掘时间的变化率，在数据集 connect、pumsb、pumsb_star 和 accidents 上运行时间减少的平均幅度分别为：1.05%、1.27%、5.35%和 1.21%。

同时也可以看出，改进算法在数据集 chess 上表现不是很好，运行时间平均增长了 1.12%。考虑到五个数据集的属性，初步得到结论：改进的 DCI_ESCS 算法适用于事务和项集较长的、较稠密的数据集上的闭频繁项集挖掘，在 2-项集剪枝时剪掉了大量不满足阈值的项集，减少了搜索空间，提高了挖掘效率，是十分有效的。后续为了验证该实验结论，我们也会在更多的数据集上进行实验分析。

3.4 本章小结

本章通过深入研究 DCI_Closed 算法的实现过程，分析其存在的不足之处，即大多针对 1-项集的剪枝策略，而在 1-项集组合连接时生成 2-项集的时候没有采取相应的剪枝策略，导致生成大量没有希望的 2-项集，需要花费大量时间进行算法的扫描、连接和计算工作，算法效率还有待提高。因此本章针对上述 DCI_Closed 算法的不足之处做出改进，将储存所有的 2-项集的支持度信息的 ESCS 结构融入到 DCI_Closed 算法中，提出了 DCI_ESCS 算法，针对 2-项集提出新的剪枝策略，尽早剪掉不满足最小支持度阈值的 2-项集及超集，从而减少算法搜索空间，提高算法挖掘效率，最后通过实验证明了该算法的可行性和有效性。

4 DCI_ESCS 算法在 ABC 库存优化问题上的应用

当今社会经济飞速发展，不断变化的市场环境对企业的发展提出巨大挑战，及时的产品供应和完善的库存管理制度对企业的发展具有非常重大的意义。为了满足产品供应企业往往会提前订购大量的库存物资，因此根据这些物资的属性进行合理分类，对不同类别物资采用不同库存控制与管理方法，可以有效地降低企业成本。但是，目前对于该问题的研究中不论是传统的 ABC 分类法、多准则 ABC 分类大都以单个物资为单位考虑其性质，很少涉及物资之间的关联性。频繁项集挖掘算法虽然能体现物资的关联性信息，但是会产生大量冗余的频繁项集，对于库存分类调整缺少可操作性，而利用闭频繁项集压缩无损的特点能够有效地解决这一问题。

现有的闭频繁项集挖掘算法有很多，均可以应用到 ABC 库存分类优化中，考虑到挖掘效率及分类调整的及时性，本章将采用改进的 DCI_ESCS 算法在一家企业往年药品销售数据上对库存分类优化问题进行研究。其中，4.1 节对库存分类优化问题进行定义，4.2 节对库存药品进行初步分类，4.3 节进行实验，通过挖掘出的闭频繁模式优化 4.2 节中的分类结果。

4.1 问题定义

企业药品库存分类优化问题就是根据不同药品的性质比如提前期、市场获得难易程度、库存成本、药品之间关联性等对药品性质重新定义，在原有的分类类别基础上改进不合理的药品分类。本文提出的利用不同药品之间的关联性改进药品库存分类是从销售的角度出发，通过分析客户订单找出常常一起购买的药品来优化库存分类，更好地满足客户需求，降低库存成本。由于企业药品销售数据中同一销售时间同一客户编号的相关明细数据可以看成是一个购物篮，即每位客户在一次订单中的药品购买模式，因此企业所有的药品销售数据就代表着所有药品购买模式，那么企业药品库存分类优化问题就可以转化为在企业药品销售明细数据中自动挖掘出经济、有效、安全的购买模式的问题，细分如下：

（1）采用文献[72]的方法，根据企业药品采购数据以平均单位成本、全年货币使用量和提前期为分类标准将库存药品分成 A、B、C 三类，作为后续分类优

化工作的依据。

（2）利用改进 DCI_ESCS 算法发现药品销售数据中的闭频繁模式，根据闭频繁模式合理改进库存分类。将挖掘出的闭频繁模式中的所有药品项与现有的库存分类类别进行对比和调整。比如，若某个药品原本属于 B 类，在购买模式中经常和某个 A 类药品一起出现，则证明该药品可能作为某 A 类的辅助药品一起使用，对用户订单至关重要，因此将该药品的类别从 B 类调整到 A 类，进行重点库存控制与管理，防止因缺货导致对企业销售 A 类药品产生影响。

4.2 初始 ABC 库存分类

本节是将现有库存药品分成 A、B、C 三类，为本章核心工作提供初始分类依据。由于不是本章重点内容，在这里仅做简要描述。

传统 ABC 库存分类方法以年货币使用量作为分类的唯一标准，忽视了对库存管理十分重要的其他指标，比如提前期、物资成本、关键性、市场获得难易程度等等，导致其分类结果具有较大局限性。因此多准则 ABC 库存分类方法（MCIC）被提出并且目前已有多种研究模型用以解决 MCIC 问题，涉及到层次分析法、数学规划、群智能算法等领域。因此本节使用文献[72]中的 MCIC 方法对库存物资进行分类。

本节实验数据是来自甘肃省兰州市一家医药企业 2017 年药品采购的明细数据。在实际操作过程中可能会因为工作人员疏忽或信息对接问题等原因导致数据输入不当，从而导致企业数据库储存的原始数据中有一些空值或多个表格中数据对接不上等问题，这些问题可能会干扰实验过程，影响库存分类及后续数据挖掘的结果，为了保证实验结果的准确性和有效性，需要对这些数据进行预处理。

（1）原始数据描述

采购数据包括入库时间、供货商编号、药品编号、进货数量、进货单价、签订日期。这些数据采用数据字典结构储存了相应的字段名称、字段类型及字段含义。如表 4.1 所示。

表 4.1 采购数据的数据字典

字段名称	字段类型	字段含义
入库时间	datatime	表示采购药品到货入库的日期
供货商编号	varchar	表示每个供货商的唯一标识
药品编号	varchar	表示每个采购药品的唯一标识
进货数量	number	表示本次采购中药品的数量
进货单价	number	表示本次采购中药品的单价
签订日期	datatime	表示采购订单合同签订的日期

（2）数据清洗

对数据库采购数据进行整理，检查是否存在字段值为空、入库时间早于签订日期等情况。经检验发现不存在字段值为空的数据，但是在单号为 2017011245 这次采购订单中，合同签订日期为 2017 年 12 月 26 日，药品入库日期为 2017 年 1 月 12 日，可能是信息录入有误，因此将这条信息删除。

（3）数据归一化处理

为了便于分析 DPC 多准则 ABC 分类模型的性能，实验数据采用了文献[70]、[72]中参与 ABC 库存分类的三个准则，即分类要素数据：平均单位成本(Average Unit Cost，AUC)、全年货币使用量(Annual Dollar Usage，ADU)和提前期(Lead Time，LT)。这三个准则均是通过上述数据清洗之后的采购数据计算得出，其中平均单位成本通过采购数据计算同一药品编号采购价格的平均值得出，全年货币使用量是通过采购数据计算出同一药品编号在所有出现的订单中采购额的累加值得出，提前期（lead time）是指从发出订单到收到订货之间的间隔时间，通过采购数据中入库时间与签订合同时间之差计算得出。

为了避免不同属性之间的量纲影响，采用与文献[68]、[72]同样方法，对三种属性值进行归一化处理，转换为标准 0-1 标度上。归一化公式如下：

$$y_{ij}^{*} = \frac{y_{ij} - \min_{i=1,2..I}\{y_{ij}\}}{\max_{i=1,2,..I}\{y_{ij}\} - \min_{i=1,2,..I}\{y_{ij}\}} \quad (4\text{-}1)$$

其中 $i = 1,2,\ldots,M$，$j = 1,2,\ldots,N$ ，y_{ij}是输入值，y_{ij}^{*}是归一化后的标准值，i 是备选方案的数量，M 是库存项目总数，j 是属性数量和 N 是属性总数。部分归

一化后的数据如表 4.2 所示。

得到三个准则数据后，采用文献[72]的方法，将库存药品进行分类，分类结果见表 4.2。690 个 SKU 中有 60 个分到 A 类（8.7%），186 个分到 B 类（27.0%），444 个分到 C 类 (64.3%)。

表 4.2 归一化与分类结果

SKU	AUC	ADU	LT	Normalized			初始分类
				AUC	ADU	LT	
0010	23.74	5231945.58	6.20	0.003	0.153	0.344	B
0013	9.00	54946.40	16.17	0.001	0.002	0.435	C
0015	167.40	622852.05	11.86	0.023	0.018	0.396	B
0024	30.81	249584.07	7.27	0.004	0.007	0.354	C
0031	18.56	17426.70	8.50	0.003	0.001	0.365	C
0039	202.00	181627.32	6.00	0.028	0.005	0.342	B
0043	53.67	79289.78	4.90	0.008	0.002	0.332	C
0045	53.45	661063.70	4.50	0.007	0.019	0.329	C
0048	62.61	707189.00	9.57	0.009	0.021	0.375	C
0049	53.54	18619081.15	5.33	0.007	0.546	0.336	B
.....	.....	.....	.....	.....	.....	.....	.....

4.3 ABC 库存分类调整优化

4.3.1 数据预处理

本节实验数据是来自甘肃省兰州市一家医药企业 2017 年药品销售的明细数据。该企业的数据库储存的原始数据格式并不适用于闭频繁项集的挖掘过程，因此在挖掘之前需要将其进行转换。

（1）原始数据描述

销售数据包括销售时间、客户编号、药品编号、销售数量、销售金额、销售单价和成本单价，各字段的含义见表 4.3 的数据字典。

表 4.3 销售数据的数据字典

字段名称	字段类型	字段含义
销售时间	datatime	表示本次订单药品销售出的日期
客户编号	varchar	表示每个客户的唯一标识
药品编号	varchar	表示每个药品的唯一标识
销售数量	number	表示本次订单中销售药品的数量
销售金额	number	表示本次订单中销售药品的总金额
销售单价	number	表示本次订单中销售药品的单价
成本单价	number	表示本次订单中销售药品的成本单价

（2）数据格式转换

该企业信息系统中的数据库原始数据格式是以企业在一次订单销售给一位客户的一件药品的相关信息储存的，而本节使用的挖掘算法的数据是以类似于购物篮格式存在的，即一位客户在一次订单中购买的全部药品的集合。因此原始数据并不适用于闭频繁项集的挖掘过程，需要对其进行转换，转换成事务数据库的格式。如 T_1：1 3 6 8 为一条事务，其中 1 3 6 8 代表该事务中的所有项，处理过程见表 4.4。

表 4.4 数据处理过程

数据转换

```
#合并同一销售时间、客户、药品编号的销售数量和销售金额，创建表 xs1
1  create table xs1 as
2  select stime,sname,ypbh,sum(xssl) as xssl,sum(xsje) as xsje
3  from xiaoshou
4  group by stime,sname,ypbh;
5  create table xs2 as
6  select stime,sname,group_concat(ypbh separator',') as ypbh,sum(xsje),
   group_concat(xsje separator',') as xsje
7  from xs1
8  group by stime,sname;
```

转换之后事务数据库的部分数据见表 4.5。

表 4.5 转换后的事务数据库

事务标识符	药品集合
T_1	0045,0049,0050,0745,0747,0761,1205,1206,1577,2046,2047
T_2	0456,0669,1757,2431
T_3	0048,0077,0475,0590,0593,0653,1352,1368,1404,1422,1482,
	1509,1558,1866,2362,2405,2418,2461
……	……

4.3.2 实验结果与分析

本节采用第 3 章提出的改进闭频繁项集挖掘算法 DCI_ESCS 在 4.3.1 小节预处理后的企业药品销售明细数据上进行实验。本次实验的目的是通过挖掘出的闭频繁模式来调整库存分类，优化库存管理方式。因此对于一个闭频繁模式，如果只包含一种类别的药品，那么这个模式对于分类优化问题是无效的。一个有效的闭频繁模式至少包含两种不同的类别。

为了方便计算，本次实验设定最小支持度阈值为 40，DCI_ESCS 算法总共挖掘出 463 个闭频繁模式，其中有效的闭频繁模式为 26 个，表 4.6 中给出了这些有效闭频繁模式中药品项集、类别对应及支持度计数信息。

表 4.6 有效闭频繁模式相关信息

闭频繁模式	类别对应	支持度计数
2036　0747	C　B	64
0045　0747	C　B	63
2036　0049	C　B	57
1248　1368	A　B	53
0045　0745	C　B	52
2036　2109	C　B	52
0045　0049	C　B	50
1404　0456	C　B	46
1254　1769	B　C	46

续表 4.6 有效闭频繁模式相关信息

闭频繁模式	类别对应	支持度计数
0711　1254	A　B	45
1769　0456	C　B	45
2036　0049　0747	C　B　B	45
1509　0456	A　B	45
0045　0745　0747	C　B　B	44
1248　1656	A　B	43
0711　0486	A　B	43
1769　0486	C　B	43
0045　0050	C　B	42
0045　0049　0747	C　B　B	42
2036　0745	C　B	42
2489　1509	B　A	42
1248　0456	A　B	41
0208　0456	C　B	41
0117　1769	B　C	40
0711　1656	A　B	40
1769　1807	C　B	40

根据表 4.6，对药品库存分类进行调整，分类调整前后对比情况见表 4.7（按支持度降序排列）。

表 4.7 药品库存分类调整前后对比

药品编号	原类别	新类别
2036	C	B
0045	C	B
1368	B	A
1404	C	B

续表 4.7 药品库存分类调整前后对比

药品编号	原类别	新类别
1769	C	B
1254	B	A
0456	B	A
1656	B	A
0486	B	A
2489	B	A
0208	C	B

以编号为 0045 的药品为例进行分析，从表 4.6 可以看出，0747 号、0745 号、0049 号以及 0050 号药品分别经常和 0045 号药品一起出现在购物篮中，也就是说，大量客户在购买 0747 号或 0745 号或 0049 号或 0050 号药品的同时也购买了 0045 号药品。0045 号药品可能是其他四种药品的辅助用药，必须配合使用或配合使用可以提高治疗效果。如果 0045 号药品出现缺货情况会直接影响客户对于其他四种药品的购买，一方面导致订单数量减少，另一方面会导致其他四种药品库存的积压，严重影响企业的经营。因此调整 0045 号药品的库存分类类别，从 C 类调整到 B 类，从原先的大批量定量订货策略调整到定期订货策略，加强了对这种药品的库存控制，优化了整个企业的库存管理方式。

4.4 本章小结

本章首先给出了 ABC 库存分类优化的明确定义，着重解释了闭项集挖掘模式对于库存分类优化的必要性，将该实际问题转化为初始 ABC 分类和根据有效闭频繁模式调整初始分类。然后开始进行实验，首先利用一种多准则库存分类模型将库存药品分成 A、B、C 三类，然后利用改进的 DCI_ESCS 算法在销售数据上挖掘出闭频繁项集，选取出有效的项集作为用户购买模式。根据用户购买模式合理地调整 A、B、C 分类，提高那些本身类别较低但是对高类别药品有重要影响的关键药品的类别，对其加强库存管理，从而提升企业管理水平。

5 总结与展望

5.1 工作总结

本文围绕ABC库存管理优化这一实际问题，从库存分类的角度出发，首先对现有闭频繁项集挖掘算法进行研究，分析了一个经典的闭频繁项集挖掘算法DCI_Closed的具体实现过程及原理，并针对不足之处对其进行改进。其次将闭频繁项集挖掘算法应用到库存分类优化中，解决使用大量频繁模式造成分类矛盾的问题。具体工作总结如下：

（1）通过查阅文献总结归纳了与本文研究内容有关的主题当前在国内外的研究现状，包括频繁项集挖掘算法、高效用项集挖掘算法、闭项集挖掘算法、数据流挖掘算法和多准则ABC库存分类方法。

（2）整理总结了与本文研究有关的理论研究，对其原理及相关定义做出详细解释。一是关联规则的定义、原理及挖掘流程，二是频繁项集挖掘的相关定义与性质，包括频繁项集挖掘和闭频繁项集挖掘，三是库存理论相关知识，包括独立需求与相关需求的定义和各自的库存控制方法，四是ABC库存管理法的定义以及对每一类的库存管理方法。

（3）提出了改进的闭频繁项集挖掘算法DCI_ESCS。首先详细阐述了经典的闭频繁项集挖掘算法DCI_Closed的实现过程，分析了该算法的优势与不足之处。针对于不足之处，提出改进的DCI_ESCS算法，详细说明改进算法的搜索空间、剪枝策略及实现过程，最后在多个公开数据集上进行实验，分析在不同最小效用阈值下的时间效率。结合不同数据集的性质，发现该改进算法在事务和项集较长的、较稠密的数据集上表现良好。

（4）利用闭频繁模式对ABC库存分类进行优化。首先对ABC库存分类优化问题给出了明确的定义，将该实际问题细化为初始ABC库存分类、闭频繁项集挖掘算法应用与分类调整。在初始分类中首先描述了企业药品采购数据的数据字典，其次对数据进行清洗、三个分类指标计算和归一化处理，最后利用多准则分类方法将库存药品分成A、B、C三类。在闭频繁项集挖掘算法应用与分类调整中，首先描述了企业药品销售数据的数据字典，其次对数据进行格式转换，利用

DCI_ESCS 算法挖掘出闭频繁项集，最后筛选出有效的闭频繁模式对原有库存分类进行调整。调整后的药品库存分类更加合理，能够降低企业库存成本，提高资金周转量。

5.2 工作展望

本文针对算法剪枝策略提出了一个改进的闭频繁项集挖掘算法 DCI_ESCS，并将其应用到企业药品 ABC 库存分类优化问题上，取得了一定成效，但是仍存在一些局限亟待解决，也成为未来工作的相关研究方向：

（1）本文针对于 DCI_Closed 算法剪枝策略局限性做出改进，针对于 2-项集提出了 ESCS 剪枝策略，但是对于 n-项集（n≥3）的剪枝策略还有待研究，希望未来能提出更多的剪枝策略来提高算法挖掘效率。

（2）本文在发现用户购买药品关联性时只考虑了药品的数量即支持度计数，没有考虑效用这一层面，而通过闭高效用挖掘算法挖掘出的闭高效用模式可以从效用值这一角度来对药品关联性重新进行阐释，发现更多隐藏的信息，也成为我们今后工作的重点内容。

（3）本文通过提高部分药品的库存类别来对库存分类进行调整，即从 B 类调整到 A 类、从 C 类调整到 A 类或 B 类。但是在实际中，部分 A 类物品也可能存在分类不合理的情况，针对于 A 类物品的库存分类调整在未来工作中有待进一步研究。

参考文献

[1] Dickie H F. ABC inventory analysis shoots for dollars not pennies[J]. Factory Management and Maintenance, 1951,109(7) : 92-94.

[2] Flores B E, Whybark D C. Multiple Criteria ABC Analysis[M]. Springer US, 2000.

[3] Ramanathan R. ABC inventory classification with multiple-criteria using weighted linear optimization[J]. Computers & operations research,2006,33(3): 695-700.

[4] 肖依永,常文兵,郭伟宏.基于关联规则的 ABC 库存分类方法[J].系统工程,2008(06):10-15.

[5] 刘振宇,徐维祥.多支持度关联规则在库存管理中的应用[J].内蒙古大学学报(自然科学版),2012,43(03):313-318.

[6] Agrawal R, Imieliński T, Swami A. Mining association rules between sets of items in large databases[C]//ACM SIGMOD RECORD International Conference on Management of Data. New York: ACM, 1999: 207-216.

[7] Agrawal R, Srikant R. Fast algorithms for mining association rules[C]//Proceedings 20th International Conference on Very Large Databases,Morgan Kaufmann, 1994:487-499.

[8] Savasere A, Omiecinski E R, Navathe S B. An Efficient Algorithm for Mining Association Rules in Large Databases.[J]. Vldb Journal, 1995:432-444.

[9] Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation[C]// ACM SIGMOD RECORD International Conference on Management of Data. New York: ACM, 2000: 1-12.

[10] Zaki M J.Scalable Algorithms for Association Mining[J]. IEEE Transactions on Knowledge & Data Engineering, 2000, 12(3):372-390.

[11] 王红梅,胡明,赵守峰. 基于垂直格式的频繁项集挖掘分段算法[J]. 吉林大学学报:理学版, 2016, 54(3):553-560.

[12] Liu Y, Liao W K, Choudhary A N. A Two-Phase Algorithm for Fast Discovery of

High Utility Itemsets[C]//Advances in Knowledge Discovery and Data Mining. Springer-Verlag, 2005:689-695.

[13] Ahmed C F, Tanbeer S K, Jeong B S, et al. Efficient Tree Structures for High Utility Pattern Mining in Incremental Databases[J]. IEEE Transactions on Knowledge and Data Engineering, 2009, 21(12): 1708-1721.

[14] Fournier-Viger P, Wu C W, Zida S, et al. FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning[C]//Proceedings of the International symposium on methodologies for intelligent systems, Roskilde, Denmark.2014: 83–92.

[15] Zida S , Fournier-Viger P , Lin C W , et al. EFIM: a fast and memory efficient algorithm for high-utility itemset mining[J]. Knowledge & Information Systems, 2017, 51(2):1-31.

[16] Lin C W, Gan W, Fournier-Viger P, et al. Mining high utility itemsets with multiple minimum utility thresholds[C]//Proceedings of the Eighth International Conference on Computer Science & Software Engineering. 2015 : 9–17.

[17] Lin C W, Gan W S, Fournier-Viger P, et al. Efficient Mining of High-Utility Itemsets Using Multiple Minimum Utility Thresholds[J]. Knowledge-Based Systems, 2016, 113:100-115.

[18] Gan W S, Lin C W, Fournier-Viger P, et al. More Efficient Algorithms for Mining High-Utility Itemsets with Multiple Minimum Utility Thresholds[C]//Proc of 27th International Conference on Database and Expert Systems Applications.Switzerland:Springer International Publishing,2016:71-87.

[19] Krishnamoorthy S. Efficient mining of high utility itemsets with multiple minimum utility thresholds[J].Engineering Applications of Artificial Intelligence, 2018, 69:112-126.

[20] 王斌, 吕瑞瑞, 房新秀,等.多最小效用阈值的频繁高效用项集快速挖掘算法[J]. 计算机应用研究, 2019, 36(12):3623 -3627.

[21] 孙蕊, 韩萌, 张春砚,等. 精简高效用模式挖掘综述[J]. 计算机应用研究, 2021, 38(4):7.

[22] Tseng V, Wu C W, Fournier-Viger P, et al. Efficient algorithms for mining top-k

high utility itemsets[C]// IEEE Trans on Knowledge and Data Engineering, 2016, 28 (1): 54-67.

[23] Duong Q H, Liao B, Fournier-Viger P, et al. An efficient algorithm for mining the top-k high utility itemsets using novel threshold raising and pruning strategies[J]. Knowledge-Based Systems, 2016, 104: 106-122.

[24] Srikumar K. Mining top-k high utility itemsets with effective threshold raising strategies[J]. Expert Systems With Applications, 2019, 117: 148–165.

[25] Han X, Liu X, Li J, et al. Efficient top-k high utility itemset mining on massive data[J]. Information Sciences,2020.

[26] Gunawan R, Winarko E, Pulungan R. A BPSO-based method for high-utility itemset mining without minimum utility threshold[J]. Knowledge-Based Systems,2020,190:105164.

[27] Baek Y, Yun U, Kim H, et al. Approximate high utility itemset mining in noisy environments[J]. Knowledge-Based Systems, 2021, 212:106596.

[28] Singh K, Kumar A, Singh S S, et al. EHNL: An Efficient Algorithm for Mining High Utility Itemsets with Negative Utility Value and Length Constraints[J]. Information Sciences, 2019,484:44-70.

[29] Fournier-Viger P, Zhang Y, Lin J C W, et al. Mining local and peak high utility itemsets[J]. Information Sciences,2019,481:344-367.

[30] Pasquier N , Bastide Y, Taouil R, et al. Discovering frequent closed itemsets for association rules[C]// Proceedings of the 7th International Conference on Database Theory. Springer, Berlin, Heidelberg, 1999:398-416.

[31] Pei J, Han J, Mao R. Closet: An efficient algorithm for mining frequent closed itemsets[C]//ACM SIGMOD workshop on research issues in data mining and knowledge discovery. NewYork:ACM,2000, 4(2): 21-30.

[32] Wang J, Han J, Pei J. CLOSET+ searching for the best strategies for mining frequent closed itemsets[C]//Proceedings of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.NewYork:ACM，2003:236-245.

[33] Grahne G, Zhu J. Efficiently using prefix-trees in mining frequent itemsets[C]//

FIMI '03,Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations:Vol90.CEUR-WS.org,2003:65.

[34] Zaki M J, Hsiao C J. CHARM: An efficient algorithm for Closed Itemset Mining[C]//Proceedings of the Second SIAM International Conference on Data Mining. Philadelphia: Society for Industrial and Applied Mathematics, 2002: 457-473.

[35] Lucchese C, Orlando S. DCI Closed: A fast and memory efficient algorithm to mine frequent closed itemsets[C]// FIMI'04， Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, 2004:Vol 126.

[36] 党红恩,赵尔平,刘炜,等.利用数据变换与并行运算的闭频繁项集挖掘方法[J].湘潭大学自然科学学报,2018,40(01):119-122.

[37] Aryabarzan N, Minaei-Bidgoli B. NEclatClosed: A vertical algorithm for mining frequent closed itemsets[J]. Expert Systems With Applications,2021,174:114738.

[38] Tseng V S, Wu C W, Fournier-Viger P, et al. Efficient algorithms for mining the concise and lossless representation of high utility itemsets[J]. IEEE Trans on Knowledge and Data Engineering, 2015, 27 (3): 726-739.

[39] Mai T, Nguyen L T T. An efficient approach for mining closed high utility itemsets and generators[J]. Journal of Information and Telecommunication, 2017,1(3):193-207.

[40] Fournier-Viger P, Zida S, Lin C W, et al. EFIM-Closed: Fast and memory efficient discovery of closed high-utility itemsets[M]// Machine Learning and Data Mining in Pattern Recognition. Springer International Publishing, 2016.

[41] Wu C W, Fournier-Viger P, Gu J Y, et al. Mining closed+high utility itemsets without candidate generation [C]// Conference on Technologies and Applications of Artificial Intelligence, 2015.

[42] Dam, Thu-Lan, et al. CLS-Miner: efficient and effective closed high-utility itemset mining[J].Frontiers of Computer Science 13.2 (2019): 357-381.

[43] Thu-Lan D, Heri R, Kjetil N. Towards efficiently mining closed high utility itemsets from incremental databases[J]. Knowledge-Based Systems, 2019, 165: 13-29.

[44]CHI Y, WANG H, YU P S，et al．Catch the moment: maintaining closed frequent itemsets over a data stream sliding window[J]．Knowledge and Information Systems, 2006, 10(3) : 265-294．

[45]YEN S-J，WU C-W，LEE Y-S，et al．A fast algorithm for mining frequent closed itemsets over stream sliding window[C]// Proceedings of the 2011 IEEE International Conference on Fuzzy Systems．Piscataway，NJ: IEEE，2011: 996-1002．

[46]NORI F，DEYPIR M，SADREDDINI M H．A sliding window based algorithm for frequent closed itemset mining over data streams[J]．Journal of Systems and Software，2013，86(3) : 615－623．

[47]Ge C C, Fu X F．Mining closed weighed frequent patterns from a sliding window over data stream[J]．Advanced Materials Research,2013,756-759(9) : 2606-2609．

[48]Zeng Q，Han G，Li W，et al．WCSPMPD-Stream: mining weighted closed sequential patterns with pattern decay over data streams[J]．Journal of Computational Information Systems，2014，10(1) : 435-442．

[49]Flores B E, Olson D L, Dorai V K. Management of multicriteria inventory classification. Mathematical and Computer modelling[J] ,1992,16(12) : 71-82.

[50]Partovi F Y, Burton J. Using the analytic hierarchy process for ABC analysis[J]. International Journal of Operations & Production Management ,1993.

[51]严婷婷,李宏余.基于 AHP 的库存产品分类模型研究[J].物流技术,2005(11):39-42.

[52]徐向宇,李乃梁,王晶,等.AHP-DEA 的备件 ABC 分类法[J].机械设计与制造,2016(08):269-272.

[53]李付伟,孙昊,王利强,等. FAHP 在包装材料库存分类管理中的应用[J].包装工程,2016,37(13):201-206.

[54]Bhattacharya A, Sarkar B, Mukherjee S K. Distance-based consensus method for ABC analysis[J]. International Journal of Production Research, 2007, 45(15) : 3405-3420.

[55]Chen J X. Multiple criteria ABC inventory classification using two virtual items[J]. International Journal of Production Research,2012, 50(6) : 1702-1713.

[56] Zhou P, Fan L. A note on multi-criteria ABC inventory classification using weighted linear optimization[J]. European journal of operational research,2007, 182(3) : 1488-1491.

[57] Ng W L. A simple classifier for multiple criteria ABC analysis[J]. European Journal of Operational Research,2007,177(1) : 344-353.

[58] Hadi-Vencheh A. An improvement to multiple criteria ABC inventory classification[J]. European Journal of Operational Research,2010, 201(3) : 962-965.

[59] 丁斌,孙连禄.基于多标准的 ABC 分类库存模型[J].电子科技大学学报(社科版),2013,15(02):31-36

[60] 朱双凯.基于组合赋权-ABC 法的地铁库存物资分类研究[J].建筑经济,2021,42(S1):433-436.

[61] Porras E, Dekker R. An inventory control system for spare parts at a refinery: An empirical comparison of different re-order point methods[J]. European Journal of Operational Research, 2008,184(1):101-132.

[62] Guvenir H A, Erel E. Multicriteria inventory classification using a genetic algorithm[J]. European journal of operational research,1998, 105(1) : 29-37.

[63] Partovi F Y, Anandarajan M. Classifying inventory using an artificial neural network approach[J]. Computers & Industrial Engineering ,2002,41(4) : 389-404.

[64] Yu M C. Multi-criteria ABC analysis using artificial-intelligence-based classification techniques[J]. Expert Systems with Applications, 2011,38(4): 3416-3421.

[65] 于俊甫,于珍,谷新平,等. 基于 SMOTE-SVM 的多准则库存分类[J]. 工业工程与管理:1-11.

[66] Tsai C Y, Yeh S W. A multiple objective particle swarm optimization approach for inventory classification[J]. International journal of production Economics, 2008,114(2) : 656-666.

[67] Aydin Keskin G, Ozkan C. Multiple criteria ABC analysis with FCM clustering[J]. Journal of Industrial Engineering ,2013.

[68]陈希,范波,吴奇石.基于模糊聚类分析的汽车配件库存分类研究[J].制造业自动化,2020,42(03):110-116.

[69]刘晔,刘晓.基于数据分析的汽车备件分类策略研究[J].工业工程与管理,2018,23(03):80-86.

[70]Lolli F, Ishizaka A, Gamberini R. New AHP-based approaches for multi-criteria inventory classification[J]. International Journal of Production Economics,2014, 156 : 62-74.

[71]吴龙涛,王铁宁,朱域.基于改进灰色聚类的装备器材库存分类方法[J].火力与指挥控制,2017,42(12):76-80.

[72]Zowid F M, Babai M Z, Douissa M R, et al. Multi-criteria inventory ABC classification using Gaussian Mixture Model[J]. IFAC-PapersOnLine, 2019, 52(13) : 1925-1930.

攻读硕士学位期间从事的科研工作及取得的成果

参与课题：

2018 年度甘肃省创新基地和人才计划自然科学基金(18JR3RA216)；

甘肃省电子商务技术与应用重点实验室开放基金课题（2018GS DZSW 63A14）。

发表论文：

[1]刘文杰，秦伟德，张晓蝶．闭项集挖掘算法研究综述[J]．大众标准化,2022(08):151-153.

致 谢

三年研究生生活转瞬即逝，直到现在还能回味起刚入学时激动万分的心情，回想起这三年的点点滴滴，不免感慨万分。在这三年中我收获的不仅是知识能力上的提升，还收获了一份份来自老师和同学们真挚的情谊，在这里我真心对他们表示感谢！

感谢我的导师杨海军教授，杨老师不仅在科研道路上为我点明方向，也在日常生活给予我鼓励和支持。在我初入研究生生活对研究方向迷茫无措时，您细致耐心地与我进行探讨，让我明确了前进的目标，也有了不断努力不断探索的底气；在我研究遇到困难时，您温暖有力的话语给予我莫大的鼓励，也让我认识到研究中存在的问题；在我每次向您请教问题时，您都会不厌其烦地为我讲解……我在这三年的收获离不开您对我的鼓励和悉心指导，成为您的学生是我的荣幸，在这里我向您致以最真挚的感谢！

感谢我的家人，谢谢你们二十多年来对我无怨无悔的付出，对我生活的照料和学业上对我的支持，让我无论经历任何大风大雨都有一个温暖的港湾可以停靠！

感谢实验室的琴姐和小伙伴们，在学业上我们共同探讨，生活上互相鼓励，这三年我们留下了很多美好的回忆，我一定会倍加珍惜！

感谢我的舍友，申宇、大莉和晓利，遇到你们是我的幸运，即使在多年之后我还会回忆起和你们畅聊到深夜、一起逛街玩耍、一起为对方排忧解难、一起开怀大笑的时光！

感谢答辩组的所有专家教授，在百忙之中抽出宝贵时间指导我的研究工作，提出宝贵意见和建议，鞭策我不断努力不断进步！

最后谢谢所有关心和帮助过我的人，谢谢你们！