

一种改进的高平均效用项集挖掘算法

罗相洲

(中铁第四勘察设计院集团有限公司 数智化事业部, 湖北 武汉 430061)

摘要: 高平均效用项集挖掘(HAUM)作为数据挖掘中的关键研究方向,旨在综合考虑项集的效用与长度,更精确地衡量项集的实际价值,但普遍存在计算时间较长的问题,限制了其在大规模数据集中的应用。为此,提出一种基于更加严格的上界剪枝策略的改进算法 FHAUI+。该算法是一阶段算法,引入两种新的上界剪枝策略有效缩小了搜索空间,显著减少了项集间的比较连接次数。实验表明,FHAUI+在多个基准数据集上的表现优于现有最先进算法,尤其在运行时间和项集连接次数方面优势明显。

关键词: 高效用;平均效用;数据挖掘;频繁模式;效用挖掘

DOI: 10.11907/rjdk.251492

中图分类号: TP18

文献标识码: A

文章编号: 1672-7800(XXXX)0XX-0001-06

扫描二维码阅读全文:



An Improved High Average Utility Itemset Mining Algorithm

LUO Xiangzhou

(Digital and Intelligent Business Division, China Railway Siyuan Survey and Design Group Co. Ltd., Wuhan 430061, China)

Abstract: High average utility itemset mining (HAUM), as a key research direction in data mining, aims to comprehensively consider the utility and length of itemsets, and more accurately measure their actual value. However, there is a common problem of long computation time, which limits its application in large-scale datasets. Therefore, an improved algorithm FHAUI+ based on a stricter upper bound pruning strategy is proposed. This algorithm is a one-stage algorithm that introduces two new upper bound pruning strategies to effectively reduce the search space and significantly decrease the number of comparison connections between itemsets. Experiments have shown that FHAUI+ outperforms existing state-of-the-art algorithms on multiple benchmark datasets, particularly in terms of runtime and number of connections to the dataset.

Key Words: high utility; average utility; data mining; frequent pattern; utility mining

0 引言

传统频繁项集仅考虑项集的出现频率,未考虑项集自身权重,因此频繁项集挖掘算法提取的信息对许多应用而言不够充分^[1-3]。为了从事务数据库中发掘更多有用信息,Yao等^[4]提出高效用项集挖掘任务,即如果一个项集在数据库中出现的次数乘以自身权重不小于用户指定阈值,则将这些项集称为高效用项集。为了减少项集的搜索空间来更快地发现高效用项集,FHM、HUPMP、ULB-Miner等大量算法被提出,显著提升了挖掘性能^[5-13]。

尽管,高效用项集相较于频繁项集挖掘可揭示更多有用信息,但存在一个重要问题——未考虑每个项集的长度,原因为包含更多商品的项集往往更容易成为高效用项集。为此,考虑到项集长度带来影响的高平均效用项集挖

掘算法被提出。Hong等^[14]设计了一个二阶段算法TPAU,采用类似Apriori的方法挖掘高平均效用项集。同时,研究者们在此基础上提出了多种高效算法:基于树的HAUPTree算法、基于投影的PAI算法、基于预估平均效用共现矩阵(EAUCS)的FHAUI、EHAUPM等^[6,15-24]。当前,FHAUI、EHAUPM是最先进的高平均效用项集挖掘算法,采用类似AU-list、EAUCS的结构进行挖掘,区别在于平局效用上界的紧凑策略不同。虽然,FHAUI、EHAUPM均取得了不错的效果,但仍与AU-list一样存在拼接耗时和搜索空间过大的问题。

为了解决这些问题,本文在FHAUI的基础上提出FHAUI+算法,设计了一个更严格的效用列表结构TAU-list(Tighter Average-Utility list),以有效减少搜索空间,提升算法执行效率。具体为,提出一个更紧凑的预估平均效用共现矩阵TEAUCM(Tighter Estimated Average Utility Co-

收稿日期:2025-07-28

作者简介:罗相洲(1991-),男,中铁第四勘察设计院集团有限公司数智化事业部工程师,研究方向为数据库、数据挖掘、大数据。

occurrence Matrix),对项集之间的拼接进行显著剪枝,以进一步优化算法挖掘过程。

1 高平均效用项集挖掘的相关概念

假设 $I = \{i_1, i_2, \dots, i_m\}$ 为包含 m 个项的集合, $D = \{T_1, T_2, \dots, T_n\}$ 为包含 n 条事务 T 的数据库。其中, T 为 I 的子集, 每条事务 $T_q \in D (1 \leq q \leq n)$ 都有一个唯一的 q , 记为 Tid ; 事务 T_q 中的每个项 $i_j (1 \leq j \leq m)$ 都有一个外部权重(数量等), 记为 $q(i_j, T_q)$; 每个项 i_j 含有一个内部权重(利润等), 记作 $p(i_j)$; 包含 k 个不同项的集合 $X = \{i_1, i_2, \dots, i_k\}$ 称为 k -项集, 其中 k 为项集长度。表 1 为数据库的示例, 每一项对应的内部权重如表 2 所示。本文设最小平均效用阈值 $minUti=24$ 。

Table 1 Database D

表 1 数据库 D

Tid 事务	(项, 外部权重)
T_1	(A, 1), (B, 6), (C, 3), (D, 3), (F, 6)
T_2	(B, 2), (C, 3), (E, 2)
T_3	(A, 2), (C, 1), (D, 2), (E, 1)
T_4	(A, 1), (B, 9), (C, 3), (D, 2), (F, 2)
T_5	(A, 3), (B, 9), (C, 3), (D, 1), (E, 1)
T_6	(C, 4), (D, 1), (E, 1)

Table 2 Inner profit

表 2 内部权重

项	A	B	C	D	E	F
内部权重	5	1	2	3	4	1

定义 1 事务 T_q 中的项 i_j 的效用值记为 $u(i_j, T_q)$ 。

$$u(i_j, T_q) = q(i_j, T_q) \times p(i_j) \quad (1)$$

定义 2 事务 T_q 中项集 X 的效用值记为 $u(X, T_q)$, 例如表 1 中 $u(AB, T_1) = u(A, T_1) + u(B, T_1) = 11$ 。

$$u(X, T_q) = \sum_{i_j \in X \wedge T_q \in D} u(i_j, T_q) \quad (2)$$

定义 3 数据库 D 中项集 X 的效用值记为 $u(X)$, 例如表 1 中 $u(AF) = u(AF, T_1) + u(AF, T_4) = 18$ 。

$$u(X) = \sum_{X \subseteq T_q \wedge T_q \in D} u(X, T_q) \quad (3)$$

定义 4 事务 T_q 中 k -项集的平均效用值记为 $au(X, T_q)$, 例如表 1 中 $au(AB, T_1) = u(AB, T_1) / 2 = 5.5$ 。

$$au(X, T_q) = \sum_{i_j \in X \wedge X \subseteq T_q} u(i_j, T_q) / |X| = k \quad (4)$$

定义 5 数据库 D 中项集 X 的平均效用值记为 $au(X)$, 例如表 1 中 $au(AF) = au(AF, T_1) + au(AF, T_4) = 9$ 。

$$au(X) = \sum_{X \subseteq T_q \wedge T_q \in D} au(X, T_q) \quad (5)$$

定义 6 如果一个项集 X 的平均效用值不低于最小平均效用阈值 $minUti$, X 就称为高平均效用项集 **HAUI**, 例如表 1 中项集 $au(AB) = 24.5 > minUti = 24$, 因此 AB 为 **HAUI**。

$$HAUI \leftarrow \{X | au(X) \geq minUti\} \quad (6)$$

定义 7 事务 T_q 最大的效用值记为 $tmu(T_q)$, 例如表 1 中

$$tmu(T_1) = 9。$$

$$tmu(T_q) = \max \{u(i_j, T_q) | i_j \subseteq T_q\} \quad (7)$$

定义 8 事务 T_q 第二大的效用值记为 $tsmu(T_q)$, 例如表 1 中 $tsmu(T_2) = 6$ 。

$$tsmu(T_q) = \begin{cases} tmu(T_q) & \text{if multi max} \\ \max \{u(i_j, T_q) | i_j \subseteq T_q \setminus tmu(T_q)\} & \text{if one max} \\ 0 & \text{if only one item in } T_q \end{cases} \quad (8)$$

定义 9 数据库 D 中项集 X 的平均效用上界记为 $auub(X)$, 例如表 1 中 $auub(F) = tmu(F, T_1) + tmu(F, T_4) = 18$ 。

$$auub(X) = \sum_{X \subseteq T_q \wedge T_q \in D} tmu(X, T_q) \quad (9)$$

$$auub(F) = tmu(F, T_1) + tmu(F, T_4) \quad (10)$$

剪枝策略 1 若数据库 D 中项集 X 的平均效用上界为 $auub(X) < minUti$, 那么 X 的超集 Y 不为高平均效用项集, 即 $au(Y) < minUti$, 其中 $X \subseteq Y$ 。

证明 1: $\forall X \subseteq D, auub(X) = \sum_{X \subseteq T_q \wedge T_q \in D} tmu(X, T_q) \geq \sum_{Y \subseteq T_q \wedge T_q \in D} tmu(Y, T_q) = auub(Y) \geq au(Y)$, 因为 $auub(X) < minUti$, 所以 $au(Y) < minUti$ 。

定义 10 在事务中, 项按 $auub$ 进行升序排序的方式记为 $<$, 即 $auub(i_1) < auub(i_2) < \dots < auub(i_n)$, 例如表 3 中统计的 $auub$, 存在 $F < B < E < A < D < C$ 。

Table 3 Upper bound on the average utility of each item

表 3 每个项的平均效用上界

项	A	B	C	D	E	F
$auub$	43	41	59	51	41	18

定义 11 如果项集 X 的平均效用上界不小于 $minUti$, 则称项集 X 为高平均效用上界项集 **HAUUBI**, 也称为候选高平均效用项集。

$$HAUUBI \leftarrow \{X | auub(X) \geq minUti\} \quad (11)$$

性质 1 事务最大平均效用向下闭包属性。如果一个项集不属于 **HAUUBI**, 那么该项集不是 **HAUI** 且该项集的超集也不是 **HAUI**。

定义 12 在一个排序的事务中, 项集 X 及其后续的项中最大的效用值记为 $rfmu(X, T_q)$, 例如表 1 中 T_1 按 $auub$ 排序后为 $F(6) < B(6) < A(5) < D(9) < C(6)$, 那么对项集 AB 的 $rfmu(AB, T_1) = 9$ 。

$$rfmu(X, T_q) = \max \{u(i_j, T_q), \dots, u(i_N, T_q)\} \quad (12)$$

式中: N 为事务 T_q 的长度。

定义 13 在一个排序的事务中, 项集 X 及其后续的项中, 第二大的效用值记为 $rsmu(X, T_q)$, 例如表 1 中 T_1 按 $auub$ 排序后, 对项集 AB 的 $rsmu(AB, T_1) = 6$ 。

$$rsmu(X, T_q) = \begin{cases} rfmu(X, T_q) & \text{if multi max} \\ \max \{u(i_j, T_q), \dots, u(i_N, T_q) \setminus rfmu(i_j, T_q)\} & \text{if one max} \\ 0 & \text{if } |X| = 1 \text{ and last item} \end{cases} \quad (13)$$

式中: N 为事务 T_q 的长度;当项集 X 中仅有一项且为排序事务中最后一项时令 $rsmu(X, T_q) = 0$ 。

2 FHAUI+算法

2.1 构建TAU-list

2.1.1 第一次数据库扫描

首先,逐行扫描数据库的每一项,统计所有项的 $auub(i_j)$;其次,依据剪枝策略1删除 $auub(i_j) < minUti$ 的项,剩下的项按 $auub$ 进行升序排序。以表1为例,统计的各个项的 $auub(i_j)$ 如表3所示。其中, $auub(F) = 18 < minUti = 24$,项 F 被删减掉;剩下的项按 $auub(i_j)$ 升序排序后为 $B < E < A < D < C$ 。

2.1.2 第二次数据库扫描

首先,重新扫描数据库的每条事务,每条事务按第一次数据库扫描得到的排序及删减项进行重整,并将重整后的数据库记为 D' ,如表4所示;其次,在扫描时进行TEAUCM矩阵及项集的TAU-list结构构建。

Table 4 Database D'

表4 数据库 D'

Tid事务	(项, 外部权重)	tmu	tsmu
T_1	(A, 1), (B, 6), (C, 3), (D, 3), (F, 6)	9	6
T_2	(B, 2), (C, 3), (E, 2)	8	6
T_3	(A, 2), (C, 1), (D, 2), (E, 1)	10	6
T_4	(A, 1), (B, 9), (C, 3), (D, 2), (F, 2)	9	6
T_5	(A, 3), (B, 9), (C, 3), (D, 1), (E, 1)	15	9
T_6	(C, 4), (D, 1), (E, 1)	8	4

在二项矩阵TEAUCM的构建过程中,先扫描数据库每行事务并排序、删减项,统计每条事务中2-项集 $tmu(T_q)$ 、 $tsmu(T_q)$,再将所有事务中2-项集的 $tmu(T_q)$ 、 $tsmu(T_q)$ 分别求和并除以2(2为项集长度),未出现的2-项集记为0。例如,在表4中计算2-项集 BE 的二项矩阵 $TEAUCM$ 为 $(tmu(BE, T_2) + tsmu(BE, T_2) + tmu(BE, T_5) + tsmu(BE, T_5))/2 = 19$ 。图1为 $TEAUCM$ 的二项矩阵(由于项集之间具有对称结构,所以本文仅计算下三角矩阵)。

项	B	E	A	D
E	19			
A	27	20		
D	27	33	35	
C	34	26	35	41

Fig. 1 Binomial matrix $TEAUCM$

图1 二项矩阵TEAUCM

TAU-list由以下3部分构成:当前项集名称、四列列表结构和后三列加和汇总的表头。列表的四列结构分别为: tid (项集的事务 Tid)、 iu (项集的效用值)、 $rfmu$ (排序事务中包含项集及其后续项的最大效用值)、 $rsmu$ (排序事务中包

含项集及其后续项的第二大效用值)。汇总表头包含: $sumUti$ (列表中 iu 的加和)、 $sumOfRfmu$ (列表中 $rfmu$ 的加和)、 $sumOfRsmu$ (列表中 $rsmu$ 的加和)。1-项集TAU-list的构建是在第二遍扫描数据库时所构建完成,例如以表4为基础构建的TAU-list结构如图2所示。其中,项 A 的第一列表示 tid ,出现在事务 T_1 、 T_3 、 T_4 、 T_5 中;第二列表示 iu ,对应事务中的效用值;第三、四列分别表示 $rfmu$ 、 $rsmu$,因为按 $B < E < A < D < C$ 排序,故对于项 A ,其 $rfmu$ 、 $rsmu$ 是从项 A 及其之后的项 D 、 C 所统计而来;项 A 的 $sumUti$ 、 $sumOfRfmu$ 、 $sumOfRsmu$ 分别等于 A 的 iu 、 $rfmu$ 、 $rsmu$ 列之和。

B	E	A
sumUti:26	sumUti:20	sumUti:35
sumOfRfmu:41	sumOfRfmu:41	sumOfRfmu:40
sumOfRsmu:27	sumOfRsmu:22	sumOfRsmu:24
1 6 9 6	2 8 8 6	1 5 9 6
2 2 8 6	3 4 10 6	3 10 10 6
4 9 9 6	5 4 15 6	4 5 6 6
5 9 15 9	6 4 8 4	5 15 15 6

D	C
sumUti:27	sumUti:34
sumOfRfmu:35	sumOfRfmu:34
sumOfRsmu:20	sumOfRsmu:0
1 9 9 6	1 6 6 0
3 6 6 2	2 6 6 0
4 6 6 6	3 2 2 0
5 3 6 3	4 6 6 0
6 3 8 3	5 6 6 0
	6 8 8 0

$\uparrow$
 $\uparrow$
 $\uparrow$
 $\uparrow$

tid iu $rfmu$ $rsmu$

Fig. 2 TAU-list of 1-itemset

图2 1-项集的TAU-list

2.2 挖掘TAU-list中高平均效用项集

本文采用递归深度优先搜索的方式从TAU-list中挖掘高平均效用项集,按 $auub(i_j)$ 升序序列将每个项集与后面项进行比较拼接,直至挖掘出所有的高平均效用项集。例如,图2的递归深度优先搜索如图3所示,先从项 B 开始依次拼接 BE 、 BEA 、 $BEAD$ 等。由于拼接过程非常耗时,因此在拼接的过程中会用到剪枝策略使搜索空间减少。

本文设项集 X 的扩展1-项集包括 X_a 、 X_b 且满足 $a < b$,当在构建项集 X 的扩展2-项集 X_{ab} 的TAU-list时, $X_{ab}.tid = X_a.tid = b.tid$ 、 $X_{ab}.iu = X_a.iu + b.iu$ 、 $X_{ab}.rfmu = a.rfmu$ 、 $X_{ab}.rsmu = a.rsmu$ 。例如,以图2中1-项集的TAU-list进行扩展时,扩展后的部分2-项集如图4所示。

剪枝策略2 数据库 D 中任意项集 X 且 $|X| \geq 2$,如果有 $(tmu(X) + tsmu(X))/2 < minUti$,则 X 及其任意超集 Y 均为不为高平均效用项集。

证明2: 记某项集 S 的总效用为 $util(S)$,则平均效用为:

$$avg(S) = \frac{util(S)}{|S|} \quad (14)$$

在构造TEAUCM时,矩阵元素 $TEAUCM[a, b]$ 被设计

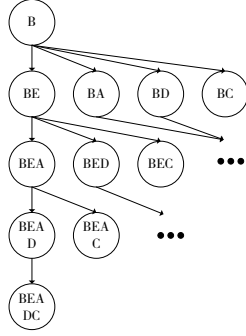


Fig. 3 Depth-first search

图3 深度优先搜索

为 $\{a, b\}$ 在数据库上效用的一个上界,因此对任意包含 a, b 的超集 S 存在:

$$util(S) \leq TEAUCM[a, b] \quad (15)$$

于是,该超集的平均效用满足为:

$$avg(S) = \frac{util(S)}{|S|} \leq \frac{TEAUCM[a, b]}{|S|} \leq \frac{TEAUCM[a, b]}{2} \quad (16)$$

因为 $|S| \geq 2$,若右端小于阈值 $minUti$,即 $\frac{TEAUCM[a, b]}{2} < minUti$,则对于任意 $S \supseteq \{a, b\}$ 都有 $avg(S) < minUti$,因此 S 不可能是 $HAUI$,则可安全剪枝。实验表明,剪枝策略2相较于剪枝策略1在多项集中更紧凑,因此在多项集中的剪枝效果更好。

BE	BA	BD
sumUtils:23	sumUtils:29	sumUtils:42
sumOfRfmu:23	sumOfRfmu:33	sumOfRfmu:33
sumOfRsmu:15	sumOfRsmu:21	sumOfRsmu:21
2 10 8 6	1 11 9 6	1 15 9 6
5 13 15 9	4 14 9 6	4 15 9 6
	5 24 15 9	5 15 9 6

Fig. 4 Partial TAU-list of 2-itemset

图4 2-项集的部分TAU-list

剪枝策略3 在TAU-list中,若项集 X 的扩展1-项集 X_a 在生成扩展2-项集 X_{ab} 时, X_a 的 $(sumOfRfmu + sumOfRsmu)/2 < minUti$,则 X_{ab} 为非高平均效用项集。

证明3: 设 $X_a.T$ 包含的事务集合为 $T(X_a)$ 。在任一事务 $t \in T(X_a)$ 中,若要在 X_a 后拼接两个新项 b, c 来尽可能提升平均效用,所能提供给新项的总单项效用在该事务上至多为 $rfmu_t + rsmu_t$ 。因此,拼接两个项后在全库上这两个新项合计能获得的效用之和的上界为 $\sum_{t \in T(X_a)} (rfmu_t + rsmu_t) = sumOfRfmu + sumOfRsmu$ 。将该上界按新增项数归一化后得到每个新增项能“分担到”的平均上界为:

$$\frac{sumOfRfmu + sumOfRsmu}{2} \quad (17)$$

由于任何由 X_a 扩展得到的长度为 $|X_a| + 2$ 的项集 S 的平均效用 $\leq$ 该归一化上界,若该归一化上界仍小于 $minUti$,则任何二项扩展都不可能成为 $HAUI$,从而 X_a 的所有更长超集均不可行,故可剪枝。

算法1 从TAU-list中挖掘高平均效用项集 *FHAUIplus*

输入: X 为项集 X 的TAU-list; exX 为项集 X 的1项扩展项集的TAU-list; $minUti$, $TEAUCM$ 。

输出:高平均效用项集 *HAUIs*。

```

1 foreach TAU-list  $X_a$  in  $exX$  do
2   if  $X_a.sumUtils/|X_a| \geq minUti$  then
3      $HAUIs \leftarrow HAUIs \cup X_a$ 
4   end
5   if  $(X_a.sumOfRfmu + X_a.sumOfRsmu)/2 \geq minUti$  then
6      $exXa \leftarrow \emptyset$ 
7   foreach TAU-list  $X_b \in exX$  where  $a < b$  do
8     if  $TEAUCM(a, b)/2 \geq minUti$  then
9        $X_{ab} = X_a \cup X_b$ 
10       $X_{ab} \leftarrow constructTAU(X, X_a, X_b)$ 
11      if  $X_{ab} \neq \emptyset$  then
12         $exXa \leftarrow exXa \cup X_{ab}$ 
13      end
14    end
15  end
16   $FHAUIplus(X_a, exXa, minUti, TEAUCM)$ 
17 end
18 end
```

伪代码中: X 为当前项集的TAU-list; $exX-X$ 为所有1-项扩展(按定义10排序)的TAU-list集合; $minUti$ -为最小平均效用阈值; $TEAUCM$ -为更紧凑的预估平均效用共现矩阵。第2~4行判断项集 X 的扩展1-项集 X_a 的平均效用值是否不小于 $minUti$,如果不小于则 X_a 为高平均效用项集。第5行判断如果 X_a 符合剪枝策略3才能被进一步扩展。第7~15行遍历排序事务中出现在项 a 之后的每一项,若项 a 与项 b 符合剪枝策略2则进一步将项集 X_a 、项 b 的TAU-list进行拼接,扩展为 X_{ab} 。第16行对项集 X_a 的所有扩展1-项集集合进行递归深度挖掘,直到挖掘出所有的高平均效用项集。

2.3 时间与空间复杂度分析

2.3.1 时间复杂度分析

*FHAUI+*的预处理阶段主要包括两次数据库扫描与事务重排,第一次扫描用于统计各项的平均效用上界,复杂度为 $O(N)$,事务重排复杂度为 $O(n \log \bar{l})$, n 为事务数, $\bar{l}$ 为事务平均长度;第二次扫描同时构建 $TEAUCM$ 与1-项TAU-list,复杂度分别为 $O(n^2)$ 与 $O(N)$ 。在挖掘阶段,算法对候选项集逐一进行拼接操作,每次拼接的代价为 $O(|X_a.T| + |X_b.T|)$,设实际发生的拼接次数为 C ,拼接时平均TAU-list长度为 $\bar{\tau}$,则挖掘阶段复杂度为 $O(C \cdot \bar{\tau})$ 。

综上,算法总的时间复杂度为 $O(n \log \bar{l} + n^2 + N + C \cdot \bar{\tau})$,剪枝策略2、剪枝策略3在实际运行中显著减少了 C 的规模,提升了整体效率。

2.3.2 空间复杂度分析

算法空间开销主要包括3个部分:① $TEAUCM$ 为对称矩阵,仅需存储下三角部分,复杂度为 $O(m^2)$, m 为项的个数;②1-项TAU-list需要为每个事务中的每个项保存一条

记录,总空间为 $O(N)$;③在递归过程中仅需保留当前搜索路径上的中间TAU-list,最坏情况下空间复杂度为 $O(mn)$,但实际中由于剪枝远小于理论上界,因此FHAUI+算法的总体空间复杂度为 $O(m^2 + N + mn)$ 。

3 实验结果与分析

本文在Mushroom、Chess、ChainStore、Kosarak等4个经典数据集上(见表5,所有数据集均为经典的测试基准数据集,数据集来源于SPMF^[18]),验证FHAUI+算法与FHAUI、EHAUPM的性能差异,实验代码均采用Java实现,运行平台为一台64位Windows 11操作系统,内存为16 GB,CPU为Intel Core Ultra 5 135H 3.60 GHz。

在实验过程中,最小平均效用阈值 $minUti$ 的选取对结果影响较大。具体为:阈值过高,能挖掘出的高平均效用项集数量有限,运行时间较短;阈值过低,则候选项集急剧膨胀,运行时间显著增加。本文为了确保实验的公平性与可比性,结合数据特征,采用自适应参数设置方法。具体而言,利用1-项TAU-list的统计量来确定数据集的尺度基准值,以反映已发现效用与潜在扩展效用的平衡。同时,在此基础上根据不同数据集的规模与稠密度,设置一系列典型的 $minUti$ 值作为实验区间:对于Mushroom、Chess等事务较长且稠密的数据集,阈值范围选择相对较高,以避免产生过多冗余候选;对于ChainStore、Kosarak等规模大且稀疏的数据集,阈值范围则适当降低,以保证能挖掘出足够的数据。此外,在这些区间进行多点测试,以全面比较不同算法在不同阈值条件下的性能。

Table 5 Database features

表5 数据特征

数据集	事务平均长度	项个数	事务条数
Mushroom	23	119	8 124
Chess	37	75	3 196
ChainStore	7	46 086	1 112 949
Kosarak	8	41 270	990 002

图5为FHAUI+、FHAUI、EHAUPM等3个算法在不同最小平均效用阈值情况下的运行时长,横坐标为最小平均效用阈值,纵坐标为运行时长,单位为s。由此可见,随着最小平均效用阈值的下降,算法的执行耗时会随之增加,原因为候选项集也随之增多,需要进行运算的也更多。由此可见,FHAUI+算法在4个数据集上均优于FHAUI、EHAUPM,证明了TAU-list和TEAUCM结构使用的更紧凑平均效用上界可有效减少算法挖掘高平均效用项集的搜索空间,提升了算法运行效率。

图6为FHAUI+、FHAUI、EHAUPM等3个算法在不同最小平均效用阈值情况下的拼接次数,横坐标为最小平均效用阈值,纵坐标为拼接次数。由此可见,拼接次数越少,搜索空间的剪枝策略效果越明显、性能更高。比较图5、图6可见,算法的运行时间会随着拼接次数增加而增加,进一

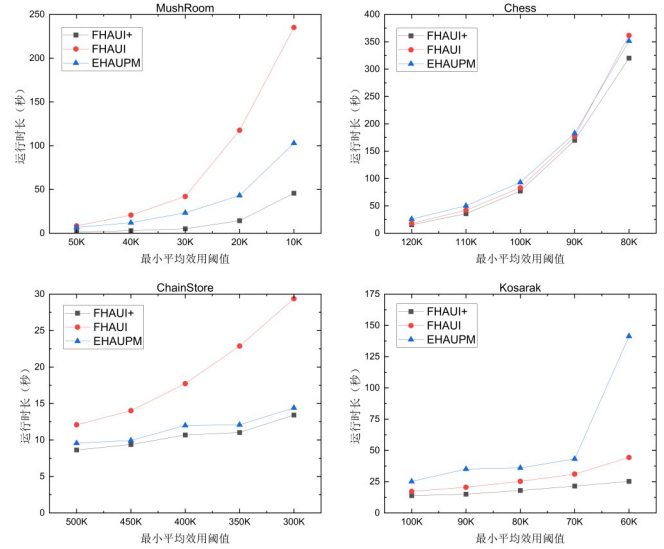


Fig. 5 Runtime duration

图5 运行时长

步说明了所提的3个删减策略优于其他两个算法。实验表明,FHAUI+在运行时间与拼接次数上的优势与本文提出的两项改进具有直接对应关系。

相较于FHAUI、EHAUPM而言,FHAUI+通过TAU-list和TEAUCM提供的二项上界,在剪枝上更紧凑,在实际挖掘过程中显著减少了需要执行的拼接操作,在稀疏且规模大的数据集上这种上界紧致性带来的收益尤为显著:TEAUCM能在拼接前大量剔除低潜力对,TAU-list的后缀top-2上界则在分支层面进一步截断不可行分支,使搜索树的膨胀得到有效控制。实验显示,在事务较长、稠密的数据集中,尽管剪枝空间相对减小,但FHAUI+仍能维持或优于现有方法的性能,证明了所提方法具有良好的通用性和稳健性。

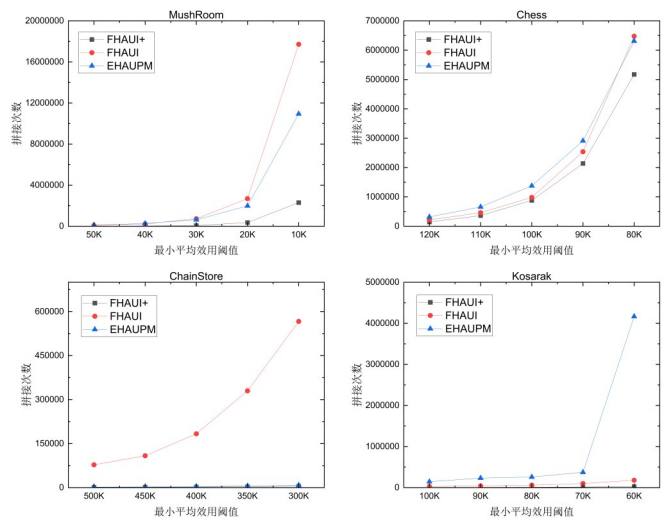


Fig. 6 Number of connections

图6 连接次数

综上,本文对高平均效用项集挖掘的启示与未来研究方向主要两方面:①将更紧凑上界思想推广到更广的框架,例如针对高维/稀疏等极端场景,研发TEAUCM存储格

式或采用近似估计降低 $O(m^2)$ 的存储开销;②探索算法在动态/流式数据与并行/分布式环境下的扩展性,以满足实际系统对时效性的需求。

4 结语

本文提出一种改进的高平均效用挖掘算法 FHAUI+。该算法给出了一个更紧凑的预估平均效用共现矩阵 **TEAUCM**,设计了一个效用列表结构 TAU-list。在4个经典数据集上,将所提模型与 FHAUI、EHAUPM 算法的比较结果表明,FHAUI+算法具有更好的时效性,在搜索空间裁剪上提升明显。

尽管,本文算法已取得一定成效,但在处理稀疏性强或维度较高的数据集时,算法的鲁棒性尚未得到充分验证;算法在动态数据环境中的扩展性也是后续研究的方向。

参考文献:

- [1] AGRAWAL R, IMIELIŃSKI T, SWAMI A. Mining association rules between sets of items in large databases[EB/OL]. <https://www.cs.columbia.edu/~gravano/Qual/Papers/14%20-%20Mining%20association%20rules%20between%20sets%20of%20items%20in%20large%20databases.pdf>.
- [2] AGRAWAL R, SRIKANT R. Fast algorithms for mining association rules in large database[C]// 20th International Conference on Very Large Data Bases, 1994: 487-499.
- [3] CUI Y, BAO Z Q. Survey of association rule mining[J]. Application Research of Computers, 2016, 33(2): 330-334.
崔妍,包志强. 关联规则挖掘综述[J]. 计算机应用研究, 2016, 33(2): 330-334.
- [4] YAO H, HAMILTON H J, BUTZ C J. A foundational approach to mining itemset utilities from databases[C]// Proceedings of the SIAM International Conference on Data Mining, 2004: 482-486.
- [5] LIU Y, LIAO W K, CHOUDHARY A. A fast high utility itemsets mining algorithm[C]// 1st International Workshop on Utility-based Data Mining, 2005: 90-99.
- [6] LIN C W, HONG T P, LU W H. Efficiently mining high average utility itemsets with a tree structure[C]// Asian Conference on Intelligent Information and Database Systems, 2010: 131-139.
- [7] LIU M, QU J. Mining high utility itemsets without candidate generation[C]// 21st ACM International Conference on Information and Knowledge Management, 2012: 55-64.
- [8] LAN G C, HONG T P, TSENG V S. An efficient gradual pruning technique for utility mining[J]. International Journal of Innovative Computing Information and Control, 2012, 8(7): 5165-5178.
- [9] FOURNIER-VIGER P, WU C, ZIDA S, et al. FHM: faster high-utility itemset mining using estimated utility co-occurrence pruning[C]// 21st International Symposium on Foundations of Intelligent Systems, 2014: 83-92.
- [10] SAHOO J, DAS K A, GOSWAMI A. An efficient approach for mining association rules from high utility itemsets[J]. Expert Systems with Applications, 2015, 42(13): 5754-5778.
- [11] WANG J H, LUO X Z, WU Q. An algorithm of high utility itemsets based on project[J]. Journal of Chinese Computer Systems, 2016, 37(6): 1212-1216.
王敬华,罗相洲,吴倩. 基于投影的高效用项集挖掘算法[J]. 小型微型计算机系统, 2016, 37(6): 1212-1216.
- [12] DAWAR S, GOYAL V, BERA D. A hybrid framework for mining high-utility itemsets in a sparse transaction database[J]. Applied Intelligence, 2017, 47(3): 809-827.
- [13] DUONG Q H, FOURNIER-VIGER P, RAMAMPIARO H, et al. Efficient high utility itemset mining using buffered utility-lists[J]. Applied Intelligence, 2018, 48(7): 1859-1877.
- [14] HONG T P, LEE C H, WANG S L. Effective utility mining with the measure of average utility[J]. Expert Systems with Applications, 2011, 38(7): 8259-8265.
- [15] LAN G C, HONG T P, TSENG V S. Efficiently mining high average-utility itemsets with an improved upper-bound strategy[J]. International Journal of Information Technology & Decision Making, 2012, 11(5): 1009-1030.
- [16] LIN J, LI T, FOURNIER-VIGER P, et al. An efficient algorithm to mine high average-utility itemsets[J]. Advanced Engineering Informatics, 2016, 30(2): 233-243.
- [17] WANG J H, LUO X Z, WU Q. Fast high average-utility itemset mining algorithm based on utility-list structure[J]. Journal of Computer Applications, 2016, 36(11): 3062-3066.
王敬华,罗相洲,吴倩. 基于效用表的快速高平均效用挖掘算法[J]. 计算机应用, 2016, 36(11): 3062-3066.
- [18] FOURNIER-VIGER P, LIN J, GOMARIZ A, et al. The SPMF open-source data mining library version 2[C]// 2016 Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 2016: 36-40.
- [19] LIN J, REN S, FOURNIER-VIGER P, et al. EHAUPM: efficient high average-utility pattern mining with tighter upper bounds[J]. IEEE Access, 2017, 5: 12927-12940.
- [20] SONG W, LIU L, HUANG C. Generalized maximal utility for mining high average-utility itemsets[J]. Knowledge and Information Systems, 2021, 63(11): 2947-2967.
- [21] ZENG X, ZHANG X L, LI X W. An effective algorithm for mining high average utility co-location patterns[J]. Journal of Dali University, 2022, 7(6): 9-17.
曾新,张晓玲,李晓伟. 高平均效用 co-location 模式挖掘的一种有效算法. 大理大学学报, 2022, 7(6): 9-17.
- [22] LI G, SHANG T, ZHANG Y. Efficient mining high average-utility itemsets with effective pruning strategies and novel list structure[J]. Applied Intelligence, 2022, 53(5): 6099-6118.
- [23] BHUVANESWARI M S, BALAGANESH N, MUNESWARAN K. H-map-based technique for mining high average utility itemset[J]. IETE Journal of Research, 2023, 69(12): 8816-8828.
- [24] WANG B, LI X H, ZHOU W, et al. High average fuzzy utility itemset mining algorithm[J]. Computer Engineering and Design, 2024, 45(5): 1398-1405.
王斌,李晓华,周伟,等. 高平均模糊效用项集挖掘算法[J]. 计算机工程与设计, 2024, 45(5): 1398-1405.

(责任编辑:刘嘉文)