doi:10.3969/j.issn.1001-4616.2025.02.013

象群优化的高效用项集挖掘算法

何菲菲,韩萌,张瑞华,李春鹏,孟凡兴

(北方民族大学计算机科学与工程学院,宁夏 银川 750021)

[**摘要**] 启发式高效用项集挖掘是近年数据挖掘领域的一个热点研究课题. 为了解决启发式高效用项集挖掘算法过早收敛导致的项集丢失问题,设计了一种新的启发式高效用项集挖掘算法,旨在较少的迭代次数内获取更多的高效用项集. 其中,提出的基于母象因子的位差进化策略,有效缩减了搜索空间,提高了算法的执行效率. 为了防止算法收敛过快陷入局部最优,提出两阶段种群多样性维护策略,保持了种群多样性和收敛性间的平衡. 在真实数据集上进行的大量实验表明,提出的算法在高效用项集数量、时空效率和算法收敛性方面均优于现有的先进算法.

[**关键词**] 高效用项集挖掘,启发式算法,象群优化,进化策略,多样性维护策略

[**中图分类号**]TP311.13　[**文献标志码**]A　[**文章编号**]1001-4616(2025)02-0124-15

Elephant Herding Optimization Algorithm for Mining High Utility Itemsets

He Feifei, Han Meng, Zhang Ruihua, Li Chunpeng, Meng Fanxing

(School of Computer Science and Engineering, North Minzu University, Yinchuan 750021, China)

Abstract: Heuristic high utility itemset mining is an active research topic in the field of data mining in recent years. To solve the problem of itemset loss caused by the early convergence of heuristic high utility itemset mining algorithms, a new algorithm is designed to discovering more high utility itemsets within fewer iterations. The proposed strategy of positional evolution based on the female elephant factor is proposed to reduce effectively the search space and improve the execution efficiency of the algorithm. Moreover, in order to prevent the algorithm from converging too quickly and falling into local optimum, the proposed strategy of two-phase population diversity maintenance which keeps a balance between population diversity and convergence. Extensive experiments on real datasets show that the proposed algorithm outperforms the advanced heuristic high utility mining algorithms.

Key words: high utility itemset mining, heuristic algorithms, elephant herding optimization, evolution strategy, population diversity maintenance strategy

频繁项集挖掘(frequent itemset mining,FIM)能挖掘出大于支持阈值的高频项集[1]. FIM 关注了项集的频率,但忽略了项集的重要性. 为了解决 FIM 的局限性,高效用项集挖掘(high utility itemset mining,HUIM)同时考虑了项集的频率和效用,旨在从定量数据集中发现高效用的项集[2]. HUIM 在现实场景中具有广泛的应用价值,尤其是在分析大规模交易数据时,其典型应用和价值主要体现在以下几个方面.

在零售和市场分析中,HUIM 可用于零售业中的购物篮分析. 通过挖掘高效用项集(high utility itemsets,HUIs),零售商可以发现对总销售额收益最大的商品组合,从而优化产品促销策略和库存管理. 在个性化推荐系统方面,HUIM 可以识别用户倾向购买的高价值商品组合,为用户提供更为精准的商品推荐,从而提升用户体验. 在医疗数据分析领域,HUIM 可以用于发现与特定疾病高度相关的药物组合或治疗方案,辅助医生进行更有效的治疗决策. 在电信网络流量分析中,HUIM 可以用于发现对网络资源消耗较高的用户行为模式. 这些高效用项集有助于网络服务提供商优化带宽管理、提高服务质量,并通过挖掘用户行为进行精准营销. 在金融数据分析中,HUIM 可用于交易模式挖掘,识别出带来高利润的投资组合

收稿日期:2024-07-01.

基金项目:国家自然科学基金项目(62062004)、宁夏自然科学基金项目(2023AAC03315)、北方民族大学中央高校基本科研业务费专项资金资助项目(2021KJCX10)、北方民族大学研究生创新项目(YCX24124).

通讯作者:韩萌,博士,教授,研究方向:大数据挖掘. E-mail:2003051@nmu.edu.cn

或交易策略. 此外,通过挖掘金融或电商交易中的 HUIs,可以发现异常高利润或异常高交易频率的组合,从而帮助识别潜在的欺诈行为.

现有的许多算法被设计以有效地挖掘 HUIs,典型的算法包括阶层组合、新型模式增长结构和投影等技术[3]. HUIM 的早期工作是两阶段算法[4],这种两阶段算法需要对数据集进行多次扫描,尤其是应对大型数据集,若生成的候选项集过多,就会导致算法的性能很差. 为了有效地生成候选项集并避免数据多次扫描,基于模式增长和树的方法被证明对 HUIM 是有效的,包括 IHUP-tree[5]、UP-tree[6] 和 HUITWU-tree[7]. 尽管这些树结构通常非常紧凑,但算法的性能与构建条件树的数量密切相关,导致了显著的内存需求. Liu 等[8]将原始数据转换为列表结构,避免了候选集生成和重复扫描数据集,从而提高了挖掘效率. 其他 HUIM 算法还包括基于最大项集的算法[9]、基于投影的方法[10]和基于位图的方法[11].

然而随着数据规模的增大,枚举所有的 HUIs 是很难实现的,这些算法的性能也会相应降低. 为了解决传统挖掘算法的性能瓶颈,近年来,启发式算法受到广泛研究并应用于 HUIM 领域. Kannimuthu 等[12]首次提出基于遗传算法(genetic algorithm, GA)的 HUIM 两种方法,其能在指数搜索空间中快速挖掘 HUIs. Lin 等[13]首次设计了基于粒子群优化(particle swarm optimization,PSO)的 HUIM 算法,并进一步开发了 OR/NOR 树[14]结构,减少了数据扫描次数和无效计算,改善了进化过程中面临的组合问题. Song 等[15]使用轮盘赌选择 HUIs 作为下一种群的初始目标,以改善种群多样性. 他们还首次提出一种基于集合粒子群的高效用项集挖掘算法 HUIM-SPSO[16],使高速运动的粒子在下一迭代中有更高的保留机会,进一步提高了多样性. Nawaz 等[17]提出基于爬山的高效用项集挖掘算法 HUIM-HC,提高了算法的时间效率. 然而在启发式 HUIM 算法中,HUIs 的发现是迭代生成的,上一迭代结果会对下一迭代 HUIs 发现产生影响. 如果在较小的迭代次数内没有随机生成新的个体,则结果很容易陷入局部最优,挖掘的 HUIs 数量也是有限的[18]. 但随机性过强会使得算法很难收敛,也会导致一定数量的 HUIs 丢失. 所以如何在有限时间内挖掘更多的高效用项集、提高种群多样以减少项集丢失是启发式 HUIM 算法正面临的挑战.

相较于其他启发式算法,象群优化(elephant herding optimization,EHO)具有全局搜索能力强、控制参数少等优点,且它的连续、组合、约束优化已经在许多基准问题和各种应用领域中得到了证明[19]. 因此本文设计了基于象群优化的高效用项集挖掘算法(high utility itemsets mining based elephant herding optimization,HUIM-EHO)以有效挖掘 HUIs. 本文的主要贡献如下:

(1)本文设计了基于母象影响因子的位差进化策略,基于该策略提出一种新的启发式高效用项集挖掘方法. 该方法有效缩减了搜索空间,提高了算法的执行效率,能够在较少的迭代次数内快挖掘出更多的 HUIs.

(2)为了避免算法因过早收敛而陷入局部最优,本文提出了一种两阶段种群多样性维护策略,保持了种群多样性和收敛性间的平衡.

(3)在密集和稀疏数据集上进行了大量实验,结果表明该方法在挖掘出的高效用项集的数量、时间效率、内存占用和收敛性方面均优于目前最先进的算法.

1 背景知识

1.1 高效用项集挖掘

令 $I=\{i_1,i_2,\cdots,i_m\}$ 是一组不同的项,$\mathrm{DB}=\{T_1,T_2,\cdots,T_n\}$ 是事务数据集,每个事务 $T_q\in\mathrm{DB}(1\leqslant q\leqslant n)$ 是 I 的子集. 项的内部效用 $q(i_j,T_q)$,表示项 i_j 在事务 T_q 中的数量. 项的外部效用 $p(i_j)$ 表示项 i_j 的利润. 项集 $X=\{i_1,i_2,\cdots,i_k\}(1\leqslant k\leqslant m)$ 是 I 的非空子集. 表 1 是一个小型事务数据集示例,表 2 展示了该数据集中不同项的外部效用.

表 1 事务数据集示例

Table 1 Example of transaction database

T_{id}	事务
T_1	a:3,c:2,e:1,g:2
T_2	c:1,d:2,f:3
T_3	a:1,b:2,c:5,e:1,g:1
T_4	b:3,d:2,f:1
T_5	b:4,c:2

表 2 项的外部效用

Table 2 External utility of each item

项	a	b	c	d	e	f	g
外部效用	2	5	3	6	9	5	13

定义 1[20](项的效用) 项 i_j 在事务 T_q 的效用表示为 $u(i_j,T_q)$,是项 i_j 在事务 T_q 中内部效用与 i_j 的外部效用的乘积,可由公式(1)计算.

$$u(i_j,T_q)=q(i_j,T_q)\times p(i_j), \tag{1}$$

例如,在表 1 和表 2 中,$u(a,T_1)=p(a)\times q(a,T_1)=2\times3=6$.

定义 2[20](项集在事务中的效用)　项集 X 在事务 T_q 中的效用表示为 $u(X,T_q)$,是项集 X 中所有项的效用之和,其中 X 包含于 T_q,可由公式(2)计算.

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

例如,$u(\{a,c\},T_1)=u(a,T_1)+u(c,T_1)=2\times3+3\times2=12$.

定义 3[20](项集在数据集中的效用)　项集 X 在数据集 DB 中的效用表示为 $u(X)$,是数据集 DB 中所有包含 X 的事务中项集 X 的效用之和,可由公式(3)计算.

$$u(X)=\sum_{X\in T_q\wedge T_q\in DB}u(X,T_q),\tag{3}$$

例如,$u(a,c)=u(\{a,c\},T_1)+u(\{a,c\},T_3)=2\times3+3\times2+2\times1+3\times5=29$.

定义 4[20](事务效用)　事务 T_q 的效用表示为 $tu(T_q)$,是 T_q 中所有项的效用之和,可由公式(4)计算.

$$tu(T_q)=\sum_{i_j\in T_q}u(i_j,T_q),\tag{4}$$

例如,$tu(T_5)=u(b,T_5)+u(c,T_5)=5\times4+3\times2=26$.

定义 5[21](事务加权效用)　项集 X 在数据集 DB 中的事务加权效用表示为 $TWU(X)$,是数据集 DB 中包含 X 的所有事务效用之和,可由公式(5)计算.

$$TWU(X)=\sum_{X\in T_q\wedge T_q\in DB}tu(T_q),\tag{5}$$

例如,$TWU(\{a,c\})=tu(T_1)+tu(T_3)=47+49=96$.

定义 6[22](最小效用)　最小效用表示为 min_util,是数据集 DB 中所有事务效用之和与用户自定义最小效用阈值 δ 的乘积,可由公式(6)计算.

$$min_util=\sum_{T_q\in DB}tu(T_q)\times\delta,\tag{6}$$

例如,假设 $\delta=35\%$,$min_util=(47+30+49+32+30)\times35\%=65.8$.

定义 7[22](高事务加权效用项集)若在数据库 DB 中,项集 X 的事务加权效用不小于 min_util,则 X 是高事务加权效用项集(high transaction weighted utilization itemset,HTWUI),否则 X 是低事务加权效用项集(low transaction weighted utilization itemset,LTWUIs).

表 3 显示了当 min_util 为 65.8 时发现的 1-项集的高事务加权效用,即 1-HTWUIs. 文中常见的符号及含义如表 4 所示,包含项集、最小效用阈值、事务加权效用、高效用项集以及高事务加权效用项集等.

表 3　1-HTWUIs 的事务加权效用

Table 3　Transaction weighted utility of 1-HTWUIs

1-Item	*a*	*b*	*c*	*e*	*g*
TWU	96	111	156	96	77

表 4　常见符号的含义

Table 4　Meaning of common symbols

符号	含义	符号	含义
X	项集	HUIs	高效用项集
δ	最小效用阈值	HTWUI	高事务加权效用项集
min_util	最小效用	LTWUI	低事务加权效用项集
TWU	事务加权效用	1-HTWUI	具有高事务加权效用的 1 项集

1.2　象群优化

在自然界中,大象生活在社会群体中. 这些群体通常由多个氏族组成,每个氏族都由一位女族长领导. 此外,公象在成长过程中也可能离开这些氏族独自生活. 为了解决各种全局优化问题,Li 等[19]于 2015 年首次提出 EHO 算法,将大象的生物行为抽象为氏族更新和氏族分离两种操作,每个氏族通过迭代更新象的位置找到问题空间的最佳解决方案.

A. 氏族更新操作

由于氏族的象在母象的领导下生活,所以对于氏族 ci 中的每个象 j,它的位置都受母象的影响,根据公式(7)进行位置的更新,其中 $x_{\mathrm{new},ci,j}$ 和 $x_{ci,j}$ 代表氏族 ci 中的象 j 的新位置和旧位置. $\alpha\in[0,1]$ 是母象

$x_{best,ci}$对 $x_{ci,j}$影响的比例因子,$r\in[0,1]$是来自均匀分布的一个随机向量.

$$x_{new,ci,j}=x_{ci,j}+\alpha.r.\ (x_{best,ci}-x_{ci,j}). \tag{7}$$

然而,当 $x_{ci,j=}x_{best,ci}$时,不能通过等式(7)更新. 母象作为问题空间中最佳的解决方案,需根据公式(8)进行位置更新,其中$\beta\in[0,1]$是 $x_{center,ci}$对 $x_{new,ci,j}$的影响因子,$x_{center,ci}$是氏族 ci 的中心位置,对于第 d 维,它可以通过公式(9)进行计算. 其中,$1\leqslant d\leqslant D$ 表示第 d 个维度,D 是总维度;n_{ci}是氏族 ci 中象的总数,$x_{ci,j,d}$是第 d 维的象 $x_{ci,j}$.

$$x_{new,ci,j}=\beta\times x_{center,ci}, \tag{8}$$

$$x_{center,ci,d}=\frac{1}{n_{ci}}\times\sum_{j=1}^{n_{ci}}x_{ci,j,d}. \tag{9}$$

B. 氏族分离操作

如前所述,象群由一些氏族组成,每个氏族都有固定数量的公象和母象. 在象群中,每一代都会有固定数量的公象离开氏族,并在远离主象群的地方独自生活. 在求解优化问题时,远离氏族公象在问题空间中被认为是最差的解决方案,根据公式(10)进行分离操作,其中 x_{max} 和 x_{min}分别是象位置的上界和下界,$\gamma\in[0,1]$是氏族 ci 中最差个体 $x_{worst,ci}$随机分布.

$$x_{worst,ci}=x_{min}+(x_{max}-x_{min}+1)\times\gamma. \tag{10}$$

C. 象群优化算法的伪代码

EHO 算法的伪代码如算法 1 所示. 算法第 1 步设置当前迭代次数为 1、最大迭代次数为 MaxGen 并初始化种群. 第 2-10 步迭代循环更新氏族和进行分离操作. 其中,第 3 步是根据象的适应度值进行排序;第 4-9 步是对氏族 ci 内每个象进行的更新操作,第 10 步为氏族分离操作,第 11 步表示种群迭代次数加一,当满足最大迭代次数时算法结束.

算法 1　EHO 算法

1:　设迭代次数 $t=1$,最大迭代次数 $MaxGen$,初始化氏族,
2:　对于当前迭代 $t<MaxGen$,
3:　进行适应度评估并根据适应度值对氏族中所有象进行排序;
4:　对氏族 c_i 中的每个象,
5:　使用公式(7)更新 $x_{ci,j}$以形成 $x_{new,ci,j}$;
6:　如果 $x_{ci,j}=x_{best,ci}$,
7:　使用公式(8)和(9)生成 $x_{new,ci,j}$;
8:　结束条件判断;
9:　结束条件循环;
10:　使用公式(10)更新氏族 c_i 中的最差的象 $x_{worst,ci}$;
11:　$t=t+1$;
12:　结束循环.

1.3　相关工作

随着数据的爆炸式增长,传统的 HUIM 算法已无法面临指数增长的搜索空间. 为了解决这一问题,已相继开发了不同启发式类别的 HUIM 算法.

Kannimuthu 等[12]首次将启发式算法应用于高效用项集挖掘领域,提出了基于遗传算法的 HUIM 两种方法,即需指定最小效用阈值的 HUPEumu-GRAM 算法和无需指定最小效用阈值的 HUPEwumu-GRAM. 其能在指数搜索空间中快速挖掘 HUIs,但容易陷入局部最优,收敛速度较慢. Zhang 等[23]设计了一种基于改进遗传算法的高效用项集挖掘算法. 该方法提出基于 1-HTWUI 的修复策略和精英邻域搜索策略使优秀个体进化概率提高,扩大了最优解的搜索空间. Lin 等[24]关注到隐私保护问题,提出了一种基于 GA 的高效用项集隐私保护算法 PPUMGAT+. 该方法扩展了 pre-large 概念,使用上下支持阈值作为缓冲区作为安全删除效用边界以隐藏敏感的高效用项集. 此外,他们还提出了一种闭合高效用项集挖掘算法 DcGA[25]. 该算法使用聚类模型将相关事务划分成组,利用紧凑遗传算法(cGA)处理分组事务以找到局部闭合高效用

项集,然后执行级联操作获取全局闭合高效用项集. 紧凑型遗传算法与通用遗传算法具有相似的精度,但减少了时间和内存消耗. 为了在大数据集上快速的挖掘闭合高效用项集(closed high utility itemsets, CHUIs),他们进一步设计了一种多目标模型 MCUI-Miner[26]. 该模型利用 k-means 分组,使用了基于 GA 的三层 MapReduce 框架生成 CHUIs. Luna 等[27]设计了基于遗传的 top-k HUIM 算法(TKHUIM-GA). 该方法通过考虑每个项的效用来指导搜索过程,以产生初始解决方案,并相应地组合解决方案,从而减少运行时间和内存消耗. 利用高效的数据表示来减少运行时间和内存使用. 与现有方法不同,TKHUIM-GA 的关键优势是可以处理含正、负整数的效用值,主要的局限性是不确定是否能找到最佳解.

不同于遗传算法,粒子群优化无需进行交叉和变异操作,且其具有控制参数少、容易实现等优点[28]. Lin 等[13]首次设计了基于 PSO 的 HUIM 算法,即 HUIM-BPSO. 该方法将获取 1-HTWUI 的数量作为粒子的大小,使用 sigmoid 函数更新粒子. 他们进一步开发了 OR/NOR 树结构用于修剪冗余组合[14]. 这种修剪策略减少了扫描次数和无效计算,改善了进化过程中面临的组合问题. 但这种树结构限制了搜索空间,导致种群的多样性降低. Sivamathi 等[29]开发了一种基于无序复杂粒子群优化的 HUIM 算法(SCE-PSO). 该方法随机选取项集点群,将数据集分成多个复合体,每次迭代点被重新分配给复合体以避免陷入局部最优. 靳晓乐等[30]提出一种双重二元粒子群优化算法 HUIM-DBPSO. 该方法使用最小效用阈值和适应度函数分散候选项集子空间以挖掘更多的 HUIs. Song 等[15]设计了基于仿生框架的三种算法,即 HUIF-GA、HUIF-PSO 和 HUIF-BA,使用轮盘赌选择 HUIs 作为下一种群的初始目标,以改善种群多样性. 他们[16]还首次提出了一种基于集合粒子群的 HUIM 算法(HUIM-SPSO),该方法使用切割集来更新位置,使高速运动的粒子在下一迭代中有更高的保留机会,进一步提高了多样性. 尽管该方法极大的提升了性能,但仍然需要很长的运行时间. Gunawan 等[31]提出了一种无需预设最小效用阈值的 HUIM 方法(HUIM-BPSO-nomut). 该方法选择高效用事务作为粒子的初始种群,设计了粒子速度更新公式,提高了整体效率. Fang 等[32]提出了一种多调整策略的启发式 HUIM 算法(HUIM-IBSPO). 该方法使用重启策略来防止算法过早收敛;采用适应度值哈希策略避免重复的适应度计算;提出粒子运动方向调整策略以发现更多的 HUIs;使用局部搜索策略来减少对重复 HUIs 的冗余评估、增加对新 HUIs 的可搜索性. 实验证明,该方法在发现的 HUIs、收敛速度和执行时间方面均表现出较好性能. Logeswaran 等[33]进一步确认粒子群优化算法的加速度系数和惯性权重对算法性能和生成结果产生的影响,提出一种自适应粒子群优化方法(APSO-RLOFF). 它采用强化学习 Q-Learning 的 RL 算法自适应校准 PSO 的控制参数,从而提高粒子群控制的性能.

Song 等[34]首次使用 PSO 挖掘高平均效用项集,提出了两种算法,即 HAUI-PSO 和 HAUI-PSOD. 前者更有效,后者可在相对较少的迭代中发现更多的高平均效用项集. Logeswaran 等[35]首次创建了多目标优化的 HUIM 模型,即 MOPSO-PHUIM. 该方法能在无阈值预设下识别出潜在的高效用项集. Tung 等[36]提出了一种基于二元粒子群优化的 top-k 高效用项集挖掘算法(TKO-BPSO). 该算法采用 sigmoid 函数更新粒子以减少进化过程中的组合问题,利用候选项集的效用阈值提升策略有效地修剪搜索空间. 此外,该方法还使用了一种效用列表结构来维护数据集中项的效用信息. Carstensen 等[37]提出了一种基于 PSO 的 top-k HUIM 方法(TKU-PSO). TKU-PSO 使用 CUV 策略修剪 1-LTWUI,并基于确定性的方法初始化第一种群. 如果种群大小大于 1-HTWUI 的数量,则由轮盘赌轮选择生成粒子并经过 PEV 检测. 此外,该方法使用最小解适应度阈值提升策略修剪候选项集,通过适应度估计策略来避免冗余项集的重复评估. 实验表明,TKU-PSO 性能显著提高,且总体准确率达到了 99.8%.

此外,Nawaz 等[17]提出基于爬山的高效用项集挖掘算法 HUIM-HC,使用了基于位图编码向量的修剪策略,大大减少了搜索空间,提高了算法的时间效率. 但该方法由于过早收敛导致了大量 HUI 的丢失,甚至在稀疏数据集上几乎挖不出任何项集. Song 等[38]从人工鱼群的角度研究高效用项集挖掘问题,提出算法 HUIM-AF. 与 HUPEumu-GRAM 和 HUIM-BPSOsig 相比,该方法可以发现更多的 HUIs. Gao 等[39]将哈里斯鹰优化算法与白鲸优化算法相结合,提出了一种基于好点集的鲸鱼初始化策略,即 WHO 算法. 该策略有助于在初始阶段引导搜索并增加种群多样性,进而提高收敛速度和算法性能. 该算法为优化问题和数据挖掘问题提供了新的解决方案. 其他启发式算法也被用于 HUIM,如人工蜂群[40]、差分进化[41]、蚁群[42]等.

2 HUIM-EHO 算法

2.1 提出的策略

2.1.1 基于母象影响因子的位差进化策略

为了适应二进制搜索空间,本文提出基于母象影响因子的进化策略以提高算法的整体效率. 如前所述,对于氏族 ci 中的每个象 j,它的位置都受母象的影响. 所以本文设计新的位置更新公式(11),将公式(7)的计算值直接适配为一个整数,该整数表示每个象要改变的比特数. 当 $x_{ci,j}=x_{\text{best},ci}$ 时,不能通过等式(11)更新. 为了保留母象的影响,本文设计等式(12)更新母象的位置向量. 其中 fitness(E_i)表示位置向量 E_i 的适应度值,它等于 E_i 对应项集的效用值.

在每次迭代进化过程中,算法首先通过等式(13)对当前种群中的每个象进行适应度评估,然后通过公式(12)获取每次迭代种群中适应度最高的个体以更新母象的位置向量. 若要获取下一种群象的位置向量,则当前种群中的每个象将根据公式(11)进行位置的更新. 其中,$x_{\text{best},ci}$ 为母象的位置向量;$x_{ci,j}$ 为种群 i 中的第 j 个象的位置向量;$\beta\in[0,1]$ 表示母象影响因子.

$$x_{n,ci,j}=\lfloor\beta\,|\,x_{\text{best},ci}\oplus x_{ci,j}\,|\,\rfloor, \tag{11}$$

$$x_{\text{best},ci}=\max\{\text{fitness}(E_i)\}, \tag{12}$$

$$\text{fitness}(E_i)=u(X). \tag{13}$$

例如,在解决 HUIM 问题时,以事务数据集表 1 和效用列表 2 为例,假设 $min_util=66$,则 1-HTWUIs 有 5 个,分别是$\{a\}$,$\{b\}$,$\{c\}$,$\{e\}$,$\{g\}$. 算法首先对原始数据集进行位图表示,如图 1 所示. 在事务 T_i 中,若存在项 j,则该项被表示为 1,否则为 0. 如图 2 所示,位置向量 E_i=<10111>表示项集$\{a,c,e,g\}$. 若要获取下一种群每个象的位置向量,需要先根据公式(12)对当前种群的每一个象进行适应度值评估,将适应度值最高的个体作为母象的位置向量. 然后将当前种群中的每个象的位置向量与母象的位置向量进行异或操作,并在每个向量的异位处进行判断,若母象的异位为 1,象的异位为 0,则将该异位 0 变为 1;若母象的位置异位为 0,象的异位为 1,则该异位不变. 过程如图 3 所示,假设在当前种群中,母象的位置向量为<10001>,象的位置向量为<01000>,则生成下一种群的象的位置向量为<11001>.

	a	b	c	e	g
T_1	1	0	1	1	1
T_2	0	0	1	0	0
T_3	1	1	1	1	1
T_4	0	1	0	0	0
T_5	0	1	1	0	0

图 1 重组事务数据集的位图表示

Fig. 1 The bitmap representation of the reorganized transaction database

	a	b	c	e	g
E_i	1	0	1	1	1

图 2 象的位置向量

Fig. 2 The position vector of elephant

2.1.2 两阶段种群多样性维护策略

在使用象群优化进行 HUIM 时,为了防止算法过早陷入局部最优,进一步维护多样性和收敛性间的平衡,本文设计了两阶段的多样性维护策略. 第一阶段是对当前种群多样性进行评估,第二阶段对不满足多样性阈值的个体进行自适应突变.

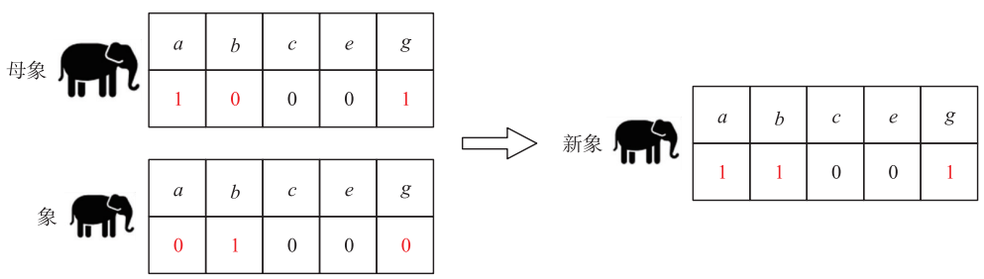


图 3 象的位置向量更新过程

Fig. 3 The process of the position vector update of elephant

(1)多样性评估

在进化过程中,尤其是在初期,若种群的多样性在发现任何 HUIs 之前迅速下降,则表明该算法此时可能陷入了局部优化,进而找不到问题搜索空间的解决方案. 为此,在种群的每次迭代中,本文设计公式(14)对当前种群多样性评估,以避免算法在找到任何 HUIs 之前陷入局部最优. 若氏族 ci 多样性评估值 PD_{ci} 小于用户自定义的多样性阈值 σ,则对当前种群进行突变. 公式(14)中,$diff_{ci}$ 表示氏族 ci 中不同象的个体总数,pop_size_{ci} 表示氏族 ci 的种群大小.

$$PD_{ci}=\frac{diff_{ci}}{pop_size_{ci}}. \tag{14}$$

(2)个体自适应突变

突变是一种保持种群多样性的常用策略. 然而大部分维护方法都是对种群个体进行简单的突变,且这种突变方式对算法收敛性产生了很大的影响. 简单的突变是改变种群个体位图编码向量上一个或多个值或遵循用户自定义的突变概率. 通常情况下,如果突变概率设置太高,搜索将变成原始随机搜索,算法很难达到收敛. 在进化初期,较大的突变概率用于进行更广泛的搜索,随着迭代次数增加,突变概率降低,较小的突变概率用于使算法收敛. 其次,适应度值高的个体是种群中的优秀个体,优秀个体保证了种群在进化中更快找到最优解,避免了搜索空间中的干扰. 所以适应度值越高,突变概率设置应越低. 所以本文提出了象的自适应突变,设计公式(15)计算突变概率 $P_{x_{ci,j}}$. 式中 $fitness_{ci}$ 表示氏族 ci 的适应度总值,$fitness_{x_{ci,j}}$ 表示突变个体 $x_{ci,j}$ 的适应度值,max_iter 表示用户设置的最大迭代次数,$iter_{x_{ci,j}}$ 表示 $x_{ci,j}$ 当前种群的迭代次数. $P_{\max}$ 为最大突变概率,一般取 1;$P_{\min}$ 为最小突变概率,一般取 0.

$$P_{x_{ci,j}}=\frac{fitness_{x_{ci,j}}}{fitness_{ci}}\left(max_iter\times\frac{P_{\max}-P_{\min}}{iter_{x_{ci,j}}}-P_{\max}\right). \tag{15}$$

例如,假设 pop_size_{ci} 为 50,$diff_{ci}$ 为 5,σ 为 20%,算法首先根据公式(14)对当前种群的多样性进行评估,计算得到 PD_{ci} 为 10%,小于用户定义的多样性阈值,则此时需要对当前种群中质量较差的个体进行突变. 假设 $fitness_{x_{ci,j}}$ 为 10,$fitness_{ci}$ 为 100,$P_{\max}$ 取 1,$P_{\min}$ 取 0,$iter_{x_{ci,j}}$ 为 20,max_iter 为 100,则通过公式(15)计算的突变概率为 0.4,由于位置向量长度为 5,所以需要对向量的 2 位进行突变,假设突变个体的位置向量为<10000>,若随机选取位置向量的第 2 位和 5 位,则突变后的位置向量为<11001>.

2.2 算法描述

本文算法 HUIM-EHO 的伪代码如算法 2 所示. 该方法的输入包括事务数据集 DB、最小效用值 min_util、种群大小 pop_size 和最大迭代次数 max_iter,输出为一组高效用项集. 算法第 1-2 步扫描 DB,计算 1-项集的事务加权效用,1-HTWUIs 的数量决定了象位置向量的长度. 第 3 步使用 Pop_Init 算法初始化种群,第 4 步令存放高效用项集的集合 $SHUI$ 为空集,设置迭代次数为 1. 第 7-10 步对象位置向量对应的项集进行适应度值评估,若不小于最小效用且不属于集合 $SHUI$,就将其存放在高效用项集合中. 第 11 步使用 Diversity_Maintain 算法中的多样性维护策略对质量较差的个体进行自适应突变,第 12 步使用 Next_Pop 算法中基于母象影响因子的位差进化策略生成下一种群,循环迭代更新种群,直到满足最大迭代次数,算法结束.

算法 2 HUIM-EHO 算法

输入:事务数据集 DB,最小效用 min_util,种群大小 pop_size,最大迭代次数 max_iter.

输出:DB 中高效用项集 HUIs.

1: 扫描 DB 并计算 DB 中所有项的 TWU;
2: 确认所有的 1-HTWUIs,令 1-HTWUIs 的个数等于象位置向量的长度;
3: 调用算法 Pop_Init;//算法 3
4: 令高效用项集集合 $SHUI=\varnothing$,当前迭代次数 $iter=1$;
5: 如果 $iter \leqslant max_iter$,
6: $i=1$ to pop_size,
7: $X=IS(E_i)$;
8: 如果 $u(X)\geqslant min_util$ 且 X 不属于集合 $SHUI$,
9: 将 X 加入到 $SHUI$ 中;

```
10:     结束判断;
11:   调用算法 Diversity_Maintain;//算法 4
12:     调用算法 Next_Pop( );//算法 5
13    结束循环;
14:   iter++;
15  结束判断;
16: 返回 HUIs 集合 SHUI.
```

算法 3 实现象群初始化,算法第 1-2 步扫描数据集以确定 1-HTWUIs 并移除所有的 1-LTWUIs,然后对重组后的数据进行位图表示. 第 4-5 步生成介于 1 到|1-HTWUIs|之间的一个随机数 num_i,轮盘赌选择 num_i 位并设置为 1 生成位置向量 E_i. 第 6-8 步对该位置向量进行 PEVC 修剪,删除无希望的编码向量以生成初始个体. 第 3-10 步逐一生成初始个体并进行判断,直至形成第一迭代种群,算法结束.

算法 3　Pop_Init 算法

```
输入:事务数据集 DB,种群大小 pop_size.
输出:第一象群.
1:  扫描 DB 识别所有的 1-HTWUIs 并移除所有 1-LTWUIs;
2:  形成重组数据集 TD 并位图表示 TD;
3:  i=1 to pop_size,
4:    生成随机数 num_i;
5:    轮盘赌选择 num_i位并设置为 1 以形成位置向量 E_i;
6:    如果 num_i>1,
7:    对位置向量进行 PEVC 修剪;
8:    结束判断;
9:    i++;
10: 结束循环;
11: 返回位置向量 E_i.
```

算法 4 使用两阶段种群多样性维护策略对种群进行多样性维护. 算法第 1 步使用公式(14)对当前种群多样性进行评估,若多样性值小于用户设置的多样性阈值,则对当前种群中不是高效用项集的个体进行突变. 第 5-7 步使用公式(15)计算当前个体的突变概率,随机选取 E_i 的 m_i 位将位置向量的 0 变为 1 或 1 变为 0. 第 2-11 步逐一对种群内个体进行判断,直至满足种群多样性阈值,算法结束.

算法 4　Diversity_Maintain 算法

```
输入:当前种群.
输出:新的种群.
1:  使用公式(14)计算当前种群的多样性值;
2:  如果 PD_ci ≤ σ,
3:  i=1 to pop_size,
4:  X=IS(E_i);
5:  如果 u(X)<min_util,
6:  使用公式(15)计算突变概率以得到 E_i 的突变位数量 m_i;
7:  随机选取 E_i 的 m_i 位将位 0 变为 1 或 1 变为 0;
8:  结束判断;
9:  i++;
10: 结束循环;
11: 结束判断;
12: 返回新的位置向量 E_i.
```

算法 5 使用基于母象影响因子的位差进化策略对象的位置向量进行更新以生成下一迭代种群. 算法第 1-2 步使用公式(11)计算当前种群中个体的适应度值,然后把适应度值最高的个体作为母象,第 4-7

步使用公式(13)将当前种群中的每个象的位置向量与母象的位置向量进行逐位比较以获得不同位的数量,然后在每个向量异位处进行判断,若象的位置异位为 0,母象的异位为 1,则将该异位 0 变为 1;否则不变. 第 3-9 步逐一生成下一种群新个体,直至满足种群大小,算法结束.

算法 5 Next_Pop 算法

输入:当前迭代种群
输出:下一迭代种群
1: 使用公式(11)计算当前种群中所有象的适应度值 fitness(E_i);
2: 令 $x_{bset,ci}=\max\{\text{fitness}(E_i)\}$;
3: $i=1$ to pop_size,
4: 使用公式(13)得到异位处;
5: 如果 $x_{bset,ci}$的第 j 位是 1 且 E_i 的第 j 位是 0,
6: 将 E_i 的 j 位 0 变为 1 或者 1 变为 0;
7: 结束判断;
8: i++;
9: 结束循环;
10: 返回下一种群的位置向量 E_i.

2.3 算法时间复杂度分析

HUIM-EHO 算法主要包括五个部分:①位图表示原始数据集,②初始化种群,③适应度评估,④种群多样性维护,⑤种群更新. 设原始数据集的事务数为 M,不同项的个数为 N,种群大小为 PS,位置向量的维数为 $D(D\leqslant N)$,最大迭代次数为 T,则算法的时空复杂度分析如下.

HUIM-EHO 算法首先需要 $O(M\times N)$的时间复杂度扫描事务数据集以计算每个单项的 TWU,然后需要 $O(M\times N)$的时间复杂度删除所有的 1-LTWUIs 并构建位图. 在初始化阶段,HUIM-EHO 对种群内每个位置向量进行轮盘赌选择的时间复杂度为 $O(PS\times D)$,需要 $O(PS)$的时间复杂度判断该向量是否为有希望的编码向量,所以初始化种群的时间复杂度为 $O(PS\times D+PS)$. 在适应度评估阶段,若计算适应度值需要 $O(Fit)$时间复杂度,则每一迭代需要 $O(PS\times F)$时间复杂度评估种群内每个象,所以适应度评估的时间复杂度为 $O(PS\times Fit\times T)$. 在维护种群多样性过程中,种群多样性评估的时间复杂度为 $O(1)$,若质量较差个体有 q 个($q\leqslant PS$)、计算自适应突变概率的时间复杂度为 $O(P)$,则种群多样性维护需要的时间复杂度为 $O(q\times P\times T)$. 种群更新需要的时间复杂度为 $O(PS\times D\times T)$. 综上所述,在省略了低阶项后,HUIM-EHO 算法的总时间复杂度为 $O(2(M\times N)+PS\times((1+D)+T(Fit+P+D)))$.

3 实验

本节对提出的 HUIM-EHO 算法进行性能评估,由于确定算法在阈值较小时产生大量候选项集,内存出现组合爆炸且长时间运行无果,所以本文将提出算法 HUIM-EHO 与目前最先进的启发式算法 HUIF-PSO、HUIF-GA、HUIM-SPSO、HUIM-HC 和 WHO 在 HUIs 数量、时空效率、算法收敛性三个方面进行比较. 对比算法源码以及数据集可从 SPMF 平台下载. 实验运行环境为 16.0 GB 可用 RAM,AMD Ryzen 7 6800HS@ 3.20 GHz CPU 以 及 Windows 11 操作系统. 因为启发式算法具有较强的随机性,所以实验数据为运行 10 次结果取得的平均值. 实验使用了 6 个真实数据集,其参数如表 5 所示.

表 5 数据集特征

Table 5 Characteristics of datasets

数据集	事务数量	项数	平均事务长度	密度	总效用
Chess	3,196	75	37	49.33%	2,156,659
Connect	67,557	129	43	33.33%	50,999,845
Accidents	340,183	468	33.8	7.22%	196,141,636
Mushroom	8,416	119	23	19.33%	3,413,720
Retail	88,162	16,470	1,030	0.06%	14,910,915
Foodmart	4,141	1,559	4.42	0.28%	12,011,023

3.1 高效用项集数量

本次实验所有算法的种群大小设置为20,最大迭代次数设置为1 000. 各算法在不同效用阈值下挖掘出的高效用项集数量的变化趋势如图4所示. 总体来说,本文算法HUIM-EHO在Chess、Connect、Mushroom、Foodmart四个数据集上优于其他五种启发式HUIM算法. 具体而言,在最小效用阈值δ为15×10^{-4}的Chess数据集上,HUIM-EHO挖掘出的高效用项集数量比HUIF-PSO、HUIF-GA、HUIM-SPSO、HUIM-HC和WHO算法分别多12 645、16 096、22 926、18 240和11 745. 在δ为13×10^{-4}的Mushroom数据集上,HUIF-PSO、HUIF-GA、HUIM-SPSO、HUIM-HC、WHO和HUIM-EHO挖掘出的高效用项集数分别为6 844、7 074、613、8 659、9 490、17 322,HUIM-EHO挖掘出的高效用项集数量比HUIM-SPSO高达28倍. 由此可见,在HUIs数量方面,HUIM-EHO在Chess和Mushroom数据集上取得了明显的优势. 在Connect和Foodmart数据集上,HUIM-EHO挖掘出的HUIs数量远大于HUIM-SPSO、HUIM-HC两种算法. 特别是在Foodmart数据集上,HUIM-SPSO、HUIM-HC发现的HUIs数量不足100.

其中,HUIM-SPSO在所有数据集上挖掘的项集数量都不如其他对比算法,尤其在Retail数据集上,HUIM-SPSO运行超过3 h仍没有实验结果. 主要原因是该算法考虑了集合粒子群中高速运转的粒子,从而导致了大量项集的丢失. 在Accidents数据集上,HUIM-EHO发现的HUIs数量不如HUIM-SPSO和HUIF-GA两种算法. 在Retail数据集上,HUIM-EHO发现的HUIs数量仅次于HUIF-GA. Accidents和Retail数据集密度分别为7.22%、0.06%,是相对稀疏的数据集,且它们的事务数较大,其中Retail数据集的平均事务长度高达1 030. HUIF-GA算法可以在事务数较大且较稀疏的数据集上进行更广泛的搜索,从而挖掘出较多的高效用项集. 实验结果表明,与HUIF-PSO、HUIF-GA、HUIM-SPSO、HUIM-HC和WHO相比,HUIM-EHO可以在Chess、Connect、Mushroom、Foodmart四个数据集上发现更多的HUIs.

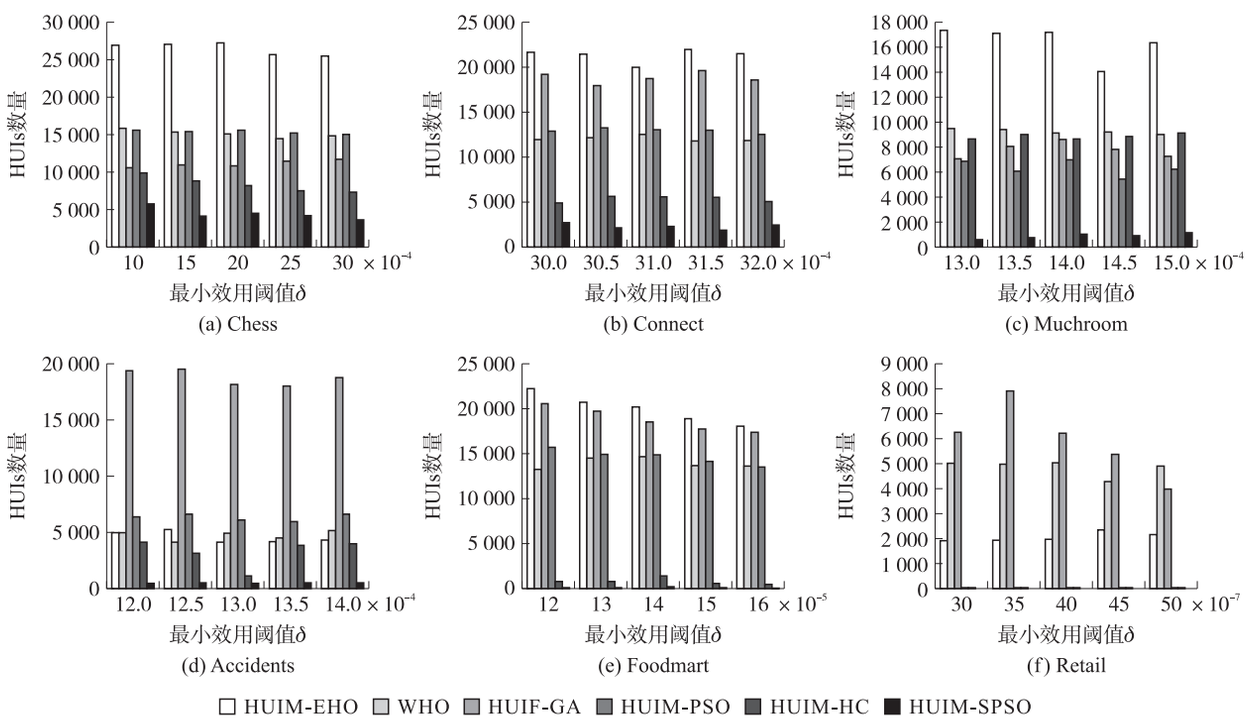


图4 HUIs数量对比

Fig. 4 Comparison of the number of HUIs

3.2 时空效率

由于启发式HUIM算法在相同迭代次数下挖掘的HUIs数量和时间总是成正比,如HUIM-HC在Foodmart数据集上表现出最好的时间效率,但在效用阈值为12×10^{-5}下,挖掘的项集数量不超过100. 尤其是当效用阈值设置的相对较高时,HUIM-HC几乎挖掘不出任何高效用项集. 其他算法虽然挖掘出了更多的项集,但也消耗了更多的时间. 所以本次实验记录了在种群大小为20、最大迭代次数为1 000下各算法挖掘相同HUIs数量所花费的时间.

各算法在不同数据集上的时间效率对比如图 5 所示. 与 HUIF-PSO、HUIF-GA、HUIM-SPSO、HUIM-HC、WHO 五种算法相比,HUIM-EHO 在 Chess、Connect、Mushroom、Accident 四个数据集上消耗了更少的时间. 在 Chess、Mushroom 和 Accidents 数据集上,HUIM-EHO 和 HUIF-PSO、HUIM-HC、WHO 算法表现出较为接近的时间效率. 主要原因是算法都使用了基于位图编码向量的修剪策略. 此外,HUIM-EHO 的位差进化策略在每次迭代进化中仅使用位图编码向量记录象的位置,没有 PSO 算法的速度、位移计算和 GA 算法的交叉和变异过程. 在 Foodmart 数据集上,HUIM-EHO 的时间性能不如 HUIM-SPSO、HUIF-GA、HUIF-PSO、HUIM-HC 和 WHO 五种算法. 在 Retail 数据集上,HUIM-EHO 的时间性能不如 HUIM-SPSO 和 WHO 算法. 主要原因是 Foodmart 和 Retail 数据集相对稀疏,需要更广泛的搜索空间,所以需要更多的执行时间. HUIM-SPSO 几乎在所有数据集都表现出最差的时间效率,特别在 Foodmart 和 Retail 数据集上,该算法平均执行时间很长,分别是 1 576 s 和大于 3 h,所以并未在图 5(e)和图 5(f)中显示.

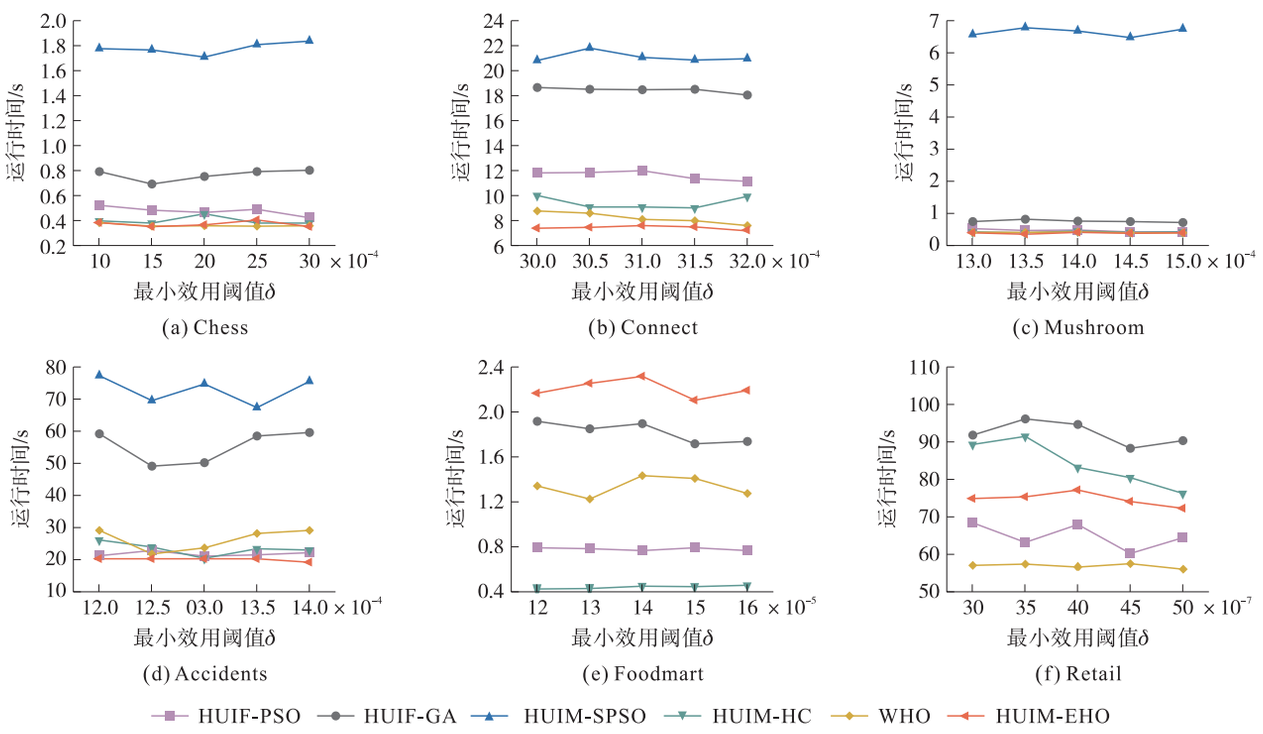


图 5　运行时间对比

Fig. 5　Comparison of running time

算法在不同数据集上内存消耗情况如图 6 所示. 与其他五种启发式高效用项集挖掘算法相比,HUIM-EHO 在 Chess、Mushroom、Accident 和 Retail 四个数据集上使用了更少的内存. 尤其在 Chess 和 Mushroom 数据集上平均占用内存都不足 100 MB. 具体而言,在最小效用阈值 δ 为 25×10^{-4}的 Chess 数据集上,HUIF-PSO、HUIF-GA、HUIM-SPSO、HUIM-HC、WHO 和 HUIM-EHO 使用的平均内存分别为 284.41、149.58、318.83、550.81、442.96、42.64 MB. 在 δ 为 14×10^{-4}的 Mushroom 数据集上,HUIF-PSO、HUIF-GA、HUIM-SPSO、HUIM-HC、WHO 和 HUIM-EHO 使用的平均内存分别为 107.01、75.81、144.12、500.27、187.61、46.58 MB. 这是因为 Chess 和 Mushroom 数据集的平均事务长度都相对较小,且事务和项的数量也是最少的. 而对于 Accidents 和 Retail 数据集,HUIM-EHO 占用内存相对较多. 主要原因是它们分别是最大的密集和稀疏数据集,对启发式 HUIM 算法来说,这意味着会形成更大的搜索空间,且 Retail 数据集为极度稀疏的数据集,大部分空间是未被搜索到的.

实验结果表明,HUIM-EHO 在 Chess、Connect、Mushroom、Accident 四个数据集上表现出最好的时间效率;在 Chess、Mushroom、Accident 和 Retail 四个数据集上使用了更少的内存.

3.3　收敛性

基于启发式的 HUIM 算法挖掘的 HUIs 数量与效用阈值成反比,且随阈值的增加,各算法所挖掘出来的 HUIs 数量达到接近水平. 为了使算法达到收敛,本次实验在 Chess、Connect、Mushroom、Accidents、

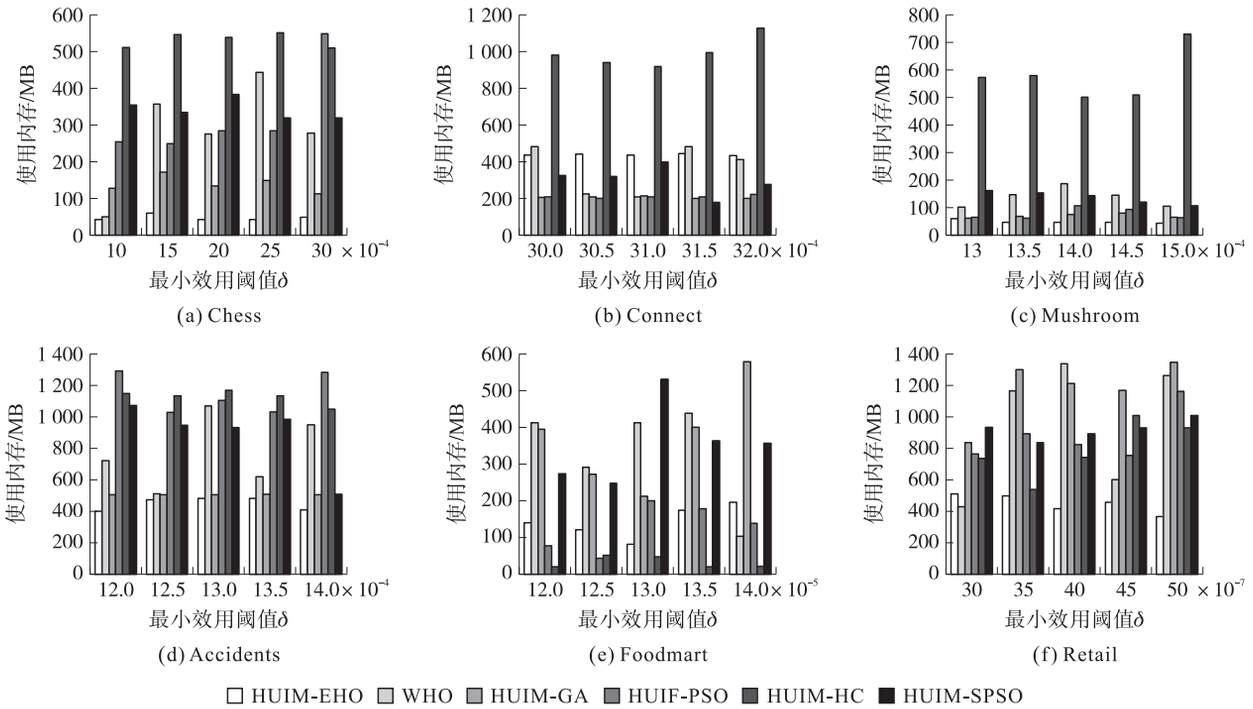


图 6　内存使用对比

Fig. 6　Memory usage comparison

Foodmart、Retail 数据集上设置的效用阈值分别为 28.5%、31.5%、14%、13%、0.16%和 0.5%,各算法的种群大小设置为 100,最大迭代次数设置为 2 000,每 200 迭代次数获取 HUIs 数量. 图 7 展示了各个算法在不同迭代次数下的收敛情况.

如图 7 所示,与 HUIF-PSO、HUIF-GA、HUIM-SPSO、HUIM-HC、WHO 五种算法相比,本文算法 HUIM-EHO 在 Chess、Connect、Mushroom、Foodmart、Retail 五个数据集上都表现出最好的收敛性. 具体而言,在 Chess 数据集上,HUIM-EHO、WHO 和 HUIF-PSO 分别收敛于 1 000、1 200 和 1 200 迭代次数,达到收敛状态的 HUIs 为 305;在 Connect 数据集上,HUIM-EHO 和 HUIF-PSO 都收敛于 1 200 次,HUIs 为 454;在

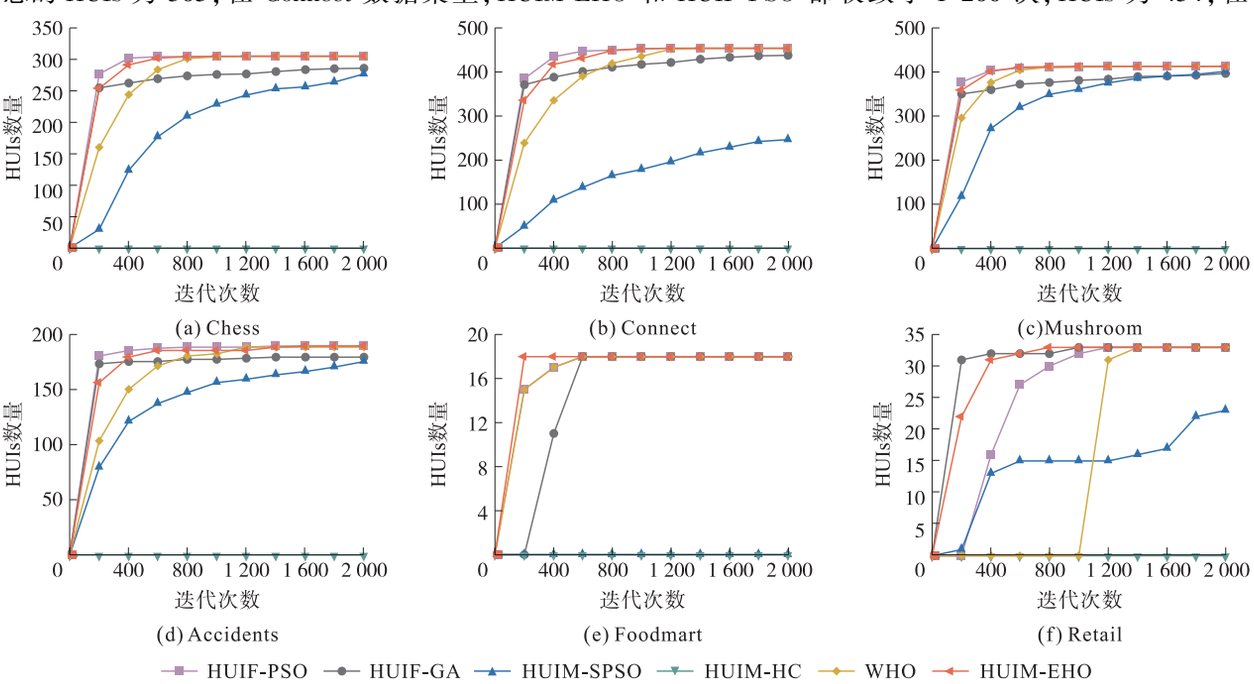


图 7　收敛性比较

Fig. 7　Comparison of convergence

Mushroom 数据集上,HUIM-EHO 和 HUIF-PSO 分别收敛于 1 000 和 1 400 迭代次数,HUIs 为 415;在 Foodmart 数据集上,HUIM-EHO、WHO、HUIF-PSO 和 HUIF-GA 分别收敛于 200、600、600、600 迭代次数,HUIs 为 18;在 Retail 数据集上,HUIM-EHO、WHO、HUIF-PSO 和 HUIF-GA 分别收敛于 800、1 400、1 200 和 1 000 迭代次数,HUIs 为 33. 虽然 HUIF-PSO 在 Chess、Connect、Mushroom、Accidents 数据集以及 HUIF-GA 在 Retail 数据集的进化初期比 HUIM-EHO 挖掘出更多的 HUIs,但 HUIM-EHO 最先达到收敛状态. 主要原因是 HUIM-EHO 中的种群多样性维护策略扩大了算法的搜索空间,使算法很难陷入局部最优并进一步减少在搜索过程中 HUI 的丢失. 在 Retail 稀疏数据集上,WHO 算法直到 1 200 迭代才发现 HUIs,说明 WHO 算法收敛性速度较慢.

此外,HUIM-SPSO 几乎在所有数据集上都未收敛,且在 Foodmart 数据集上运行超过 12 h 仍未显示结果,所以未在图 7(e)中显示. 在本次实验阈值设置较高的条件下,HUIM-HC 在所有数据集上都挖不出任何 HUIs. 实验结果表明,与其他四种算法相比,HUIM-EHO 在 Chess、Connect、Mushroom、Foodmart、Retail 数据集上都表现出最好的收敛性.

4 结论

本文设计了一种新的启发式 HUIM 算法,能够在较少的迭代次数内快挖掘出较多的 HUIs. 其中,基于母象影响因子的位差进化策略有效缩减了搜索空间,提高了算法的执行效率;两阶段种群多样性维护策略避免了算法因收敛过快陷入局部最优,很好的维持了种群多样性和收敛性间的平衡. 在真实数据集中进行的大量实验结果表明,与最先进的算法相比,本文所提出的算法在 HUIs 数量、时空效率和收敛性方面具有优越的性能. 但目前该方法只适用于静态数据集,且在稀疏数据集上的表现性能还有待提高. 未来,本研究组将对算法 HUIM-EHO 继续优化,并使用滑动窗口技术实现数据流上高效用项集的动态挖掘.

[参考文献]

[1] AGRAWAL S, VARGHESE T, SINHA T, et al. Data mining for category of online ads that is more profitable using ant colony optimization[C]//Computational Vision and Bio-Inspired Computing: Proceedings of ICCVBIC 2022. Singapore: Springer Nature Singapore, 2023: 743-755.

[2] KUMAR R, SINGH K. High utility itemsets mining from transactional databases: a survey[J]. Applied intelligence, 2023, 53(22): 27655-27703.

[3] SUKANYA N S, THANGAIAH P R J. Enhanced differential evolution and particle swarm optimization approaches for discovering high utility itemsets[J]. International journal of computational intelligence and applications, 2023, 22(1): 2341005.

[4] PAZHANIRAJA N, SOUNTHARRAJAN S, SUGANYA E, et al. Optimizing high-utility item mining using hybrid dolphin echolocation and Boolean grey wolf optimization[J]. Journal of ambient intelligence and humanized computing, 2023, 14(3): 2327-2339.

[5] 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.

[6] TSENG V S, SHIE B E, WU C W, et al. Efficient algorithms for mining high utility itemsets from transactional databases[J]. IEEE transactions on knowledge and data engineering, 2012, 25(8): 1772-1786.

[7] GUO S M, GAO H. HUITWU: an efficient algorithm for high-utility itemset mining in transaction databases[J]. Journal of computer science and technology, 2016, 31: 776-786.

[8] LIU M, QU J. Mining high utility itemsets without candidate generation[C]//Proceedings of the 21st ACM International Conference on Information and Knowledge Management. Mavi, Hawaii, USA: Association for Computing, 2012: 55-64.

[9] LIN M Y, TU T F, HSUEH S C. High utility pattern mining using the maximal itemset property and lexicographic tree structures[J]. Information sciences, 2012, 215: 1-14.

[10] LAN G C, HONG T P, TSENG V S. An efficient projection-based indexing approach for mining high utility itemsets[J]. Knowledge and information systems, 2014, 38: 85-107.

[11] SONG W, LIU Y, AND LI J, BAHUI: fast and memory efficient mining of high utility itemsets based on bitmap[J]. International journal of data warehousing and mining, 2014, 10(1): 1-15.

[12] KANNIMUTHU S,PREMALATHA K. Discovery of high utility itemsets using genetic algorithm with ranked mutation[J]. Applied artificial intelligence,2014,28(4):337-359.
[13] LIN J C W,YANG L,FOURNIER-VIGER P,et al. Mining high-utility itemsets based on particle swarm optimization[J]. Engineering applications of artificial intelligence,2016,55:320-330.
[14] LIN J C W,YANG L,FOURNIER-VIGER P,et al. A binary PSO approach to mine high-utility itemsets[J]. Soft computing,2017,21:5103-5121.
[15] SONG W,HUANG C. Mining high utility itemsets using bio-inspired algorithms:a diverse optimal value framework[J]. IEEE access,2018,6:19568-19582.
[16] SONG W,LI J. Discovering high utility itemsets using set-based particle swarm optimization[C]//Advanced Data Mining and Applications:16th International Conference. Foshan,China:Springer International Publishing,2020:38-53.
[17] NAWAZ M S,FOURNIER-VIGER P,YUN U,et al. Mining high utility itemsets with hill climbing and simulated annealing[J]. ACM transactions on management information system(TMIS),2021,13(1):1-22.
[18] SUBRAMANIAN K,KANDHASAMY P. Mining high utility itemsets using genetic algorithm based-particle swarm optimization (GA-PSO)[J]. Journal of intelligent & fuzzy systems,2023,44(1):1-21.
[19] LI W,WANG G G. Improved elephant herding optimization using opposition-based learning and K-means clustering to solve numerical optimization problems[J]. Journal of ambient intelligence and humanized computing,2023,14(3):1753-1784.
[20] GAO Z,HAN M,LIU S,et al. Survey of high utility itemset mining methods based on intelligent optimization algorithm[J]. Journal of computer applications,2023,43(6):1676.
[21] YUAN Q,TANG C,XU Y. Bat algorithm for high utility itemset mining based on length constraint[J]. Journal of computer applications,2023,43(5):1473.
[22] FANG W,JIANG H,LU H,et al. GPU-Based efficient parallel heuristic algorithm for high-utility itemset mining in large transaction datasets[J]. IEEE transactions on knowledge and data engineering,2024,36(2):652-667.
[23] ZHANG Q,FANG W,SUN J,et al. Improved genetic algorithm for high-utility itemset mining[J]. IEEE access,2019,7:176799-176813.
[24] LIN J C W,GAN W,FOURNIER-VIGER P,et al. High utility-itemset mining and privacy-preserving utility mining[J]. Perspectives in science,2016,7:74-80.
[25] LIN J C W,DJENOURI Y,SRIVASTAVA G,et al. A predictive GA-based model for closed high-utility itemset mining[J]. Applied soft computing,2021,108:107422.
[26] LIN J C W,DJENOURI Y,SRIVASTAVA G,et al. Efficient evolutionary computation model of closed high-utility itemset mining[J]. Applied intelligence,2022,52(9):10604-10616.
[27] LUNA J M,KIRAN R U,FOURNIER-VIGER P,et al. Efficient mining of top-k high utility itemsets through genetic algorithms[J]. Information sciences,2023,624:529-553.
[28] 高智慧,韩萌,李昂. HHUIM:一种新的启发式高效用项集挖掘方法[J]. 计算机应用研究,2024,41(1):94-101.
[29] SIVAMATHI C,VIJAYARANI S. Mining high utility itemsets using shuffled complex evolution of particle swarm optimization (SCE-PSO) optimization algorithm[C]//2017 International Conference on Inventive Computing and Informatics(ICICI). Coimbatore,India:IEEE,2017:640-644.
[30] 靳晓乐,刘峡壁,马骁. 基于双重二元粒子群优化的高效用项集挖掘算法[J]. 计算机工程,2018,44(12):202-207,214.
[31] 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.
[32] FANG W,ZHANG Q,LU H,et al. High-utility itemsets mining based on binary particle swarm optimization with multiple adjustment strategies[J]. Applied soft computing,2022,124:109073.
[33] LOGESWARAN K, SURESH P, ANANDAMURUGAN S. Particle swarm optimization method combined with off policy reinforcement learning algorithm for the discovery of high utility itemset[J]. Information technology and control,2023,52(1):25-36.
[34] SONG W,HUANG C. Mining high average-utility itemsets based on particle swarm optimization[J]. Data science and pattern recognition,2020,4(2):19-32.
[35] LOGESWARAN K,SATHASIVAM R,SURESH P,et al. Discovery of potential high utility itemset from uncertain database using multi objective particle swarm optimization algorithm[C]//2022 International Conference on Advanced Computing Technologies and Applications(ICACTA). Mumbai,India:IEEE,2022:1-6.
[36] TUNG N T,NGUYEN T D D,NGUYEN L T T,et al. A nature-inspired method to mine top-k multi-level high-utility itemsets[J]. Cybernetics and systems,2023:1-22.

[37] CARSTENSEN S,CHUN-WEI LIN J. TKU-PSO:an efficient particle swarm optimization model for top-k high-utility itemset mining[J]. International journal of interative multimedia and artificial intelligence,2024:1-12.
[38] SONG W, LI J, HUANG C. Artificial fish swarm algorithm for mining high utility itemsets[C]//Advances in Swarm Intelligence:12th International Conference. Qingdao,China:Springer International Publishing,2021:407-419.
[39] GAO Z, HAN M, LIU S, et al. High utility itemsets mining based on hybrid harris hawk optimization and beluga whale optimization algorithms[J]. Journal of intelligent & fuzzy systems,2024,46(4):7567-7602.
[40] SONG W, HUANG C. Discovering high utility itemsets based on the artificial bee colony algorithm[C]//Advances in Knowledge Discovery and Data Mining: 22nd Pacific-Asia Conference. Melbourne, VIC, Australia: Springer International Publishing,2018:3-14.
[41] SUKANYA N S, THANGAIAH P R J. Enhanced differential evolution and particle swarm optimization approaches for discovering high utility itemsets[J]. International journal of computational intelligence and applications, 2023, 22(1): 2341005.
[42] ARUNKUMAR M S, SURESH P, GUNAVATHI C. High utility infrequent itemset mining using a customized ant colony algorithm[J]. International journal of parallel programming,2020,48:833-849

[责任编辑:杜忆忱]

(上接第 123 页)

[7] XIONG L,ZHONG X,XIONG N N,et al. QR-3S:a high payload QR code secret sharing system for industrial Internet of Things in 6G networks[J]. IEEE transactions on industrial informatics,2020,17(10):7213-7222.
[8] LI L L,HUANG P C,XIE R. QR Code secret sharing algorithm based on CRC[C]//2022 International Conference on Image Processing and Computer Vision. Okinawa,Japan:IEEE,2023:134-139.
[9] Information Technology:Automatic Identification and Data Capture Techniques:QR Code Bar Code Symbology Specification[M]. ISO/IEC 18004:2015. Geneva,Switzerland:ISO,2015.
[10] ZHANG L,SUN J,ZHANG X,et al. Two-level QR Code scheme based on region matrix image secret sharing algorithm[J]. Mathematical biosciences and engineering,2023,20(9):16678-16704.
[11] RAWAT A S, DESHMUKH M, SINGH M. QR shares based secret sharing scheme using DWT, random permutation and arithmetic modulo operation for QR secret[C]//2023 10th International Conference on Signal Processing and Integrated Networks. Noida,India:IEEE,2023:772-777.
[12] CHUANG J C, HU Y C, KO H J. A novel secret sharing technique using QR Code[J]. International journal of image processing,2010,4(5):468-475.
[13] CHOW Y W,SUSILO W,YANG G,et al. Exploiting the error correction mechanism in QR Codes for secret sharing[C]// Information Security and Privacy:21st Australasian Conference. Melbourne,VIC,Australia:Springer International Publishing, 2016:409-425.
[14] CHENG Y, FU Z, YU B. Improved visual secret sharing scheme for QR Code applications[J]. IEEE transactions on information forensics and security,2018,13(9):2393-2403.
[15] LIN P Y. Distributed secret sharing approach with cheater prevention based on QR Code[J]. IEEE transactions on industrial informatics,2016,12(1):384-392.
[16] HUANG P C,CHANG C C,LI Y H. Sudoku-based secret sharing approach with cheater prevention using QR Code[J]. Multimedia tools and applications,2018,77:25275-25294.
[17] COX R. QArt codes[R/OL]. [2012-04-12]. http://research.swtch.com/qart.
[18] SAEIDI Z,YAZDI A,MASHHADI S,et al. High performance image steganography integrating IWT and Hamming Code within secret sharing[J]. IET image processing,2024,18(1):129-139.
[19] GOTTIMUKKALA A R, KUMAR N, DASH J K, et al. Image watermarking based on remainder value differencing and extended Hamming Code[J]. Journal of electronic imaging,2024,33(1):011003.
[20] MUNOZ-MEJIAS D, GONZALEZ-DIAZ I, DIAZ-DE-MARIA F. A low-complexity pre-processing system for restoring low-quality QR Code images[J]. IEEE transactions on consumer electronics,2011,57(3):1320-1328.

[责任编辑:杜忆忱]