计算机科学
COMPUTER SCIENCE

基于遗传算法的生物启发频繁项集挖掘策略

赵学健，赵可

引用本文

赵学健，赵可．基于遗传算法的生物启发频繁项集挖掘策略[J]．计算机科学，2023，50(11A)：220700200-8.

ZHAO Xuejian, ZHAO Ke. Bio-inspired Frequent Itemset Mining Strategy Based on Genetic Algorithm [J]. Computer Science, 2023, 50(11A): 220700200-8.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220700200

基于遗传算法的生物启发频繁项集挖掘策略

赵学健[1,2]　赵　可[1]
1 江苏省邮政大数据技术与应用工程中心(南京邮电大学)　南京 210003
2 宽带无线通信与传感网技术教育部重点实验室(南京邮电大学)　南京 210003

摘　要　精确频繁项集挖掘算法时间效率低下,在处理大规模数据集时力不从心。针对该问题,提出一种基于遗传算法的频繁项集挖掘策略GAA-FIM(Genetic Algorithm combining Apriori property based Frequent Itemset Mining),给出了编码操作、交叉操作、变异操作和选择操作的详细操作规则。该算法将遗传算法与精确频繁项集挖掘算法的向下闭包特性融合,改进了传统的有性繁殖的交叉操作方式,将具有良好遗传基因的个体优先加入到新一代候选种群中,并通过变异操作扩展新一代候选种群的规模,以提升算法的时间效率,获取更佳质量的频繁项集。基于合成数据集和真实数据集对GAA-FIM算法的性能进行了验证,实验结果表明GAA-FIM算法与GAFIM和GA-Apriori等算法相比具有更好的时间效率,频繁项集质量也得到了进一步提升。

关键词:频繁项集;遗传算法;生物启发;向下闭包特性;数据挖掘
中图法分类号　TP391

Bio-inspired Frequent Itemset Mining Strategy Based on Genetic Algorithm

ZHAO Xuejian[1,2] and ZHAO Ke[1]
1 Technology and Application Engineering Center of Postal Big Data, Nanjing University of Posts and Telecommunications, Nanjing 210003, China
2 Key Lab of Broadband Wireless Communication and Sensor Network Technology of Ministry of Education, Nanjing University of Posts and Telecommunications, Nanjing 210003, China

Abstract　Precise frequent itemset mining algorithms usually have a low time efficiency, particularly in processing large-scale data sets. To solve this problem, a frequent itemset mining algorithm, genetic algorithm combining apriori property based frequent itemset mining(GAA-FIM), is proposed, which combines the genetic algorithm and the downward closure property of precise frequent itemset mining algorithms. The detailed operation rules of coding operation, crossover operation, mutation operation and selection operation are described in detail. In GAA-FIM algorithm, individuals with good genes are preferentially added to the latest generation of candidate population through the asexual crossover operation process and the scale of the new generation candidate can be expanded through mutation operation. Therefore, the time efficiency of the proposed algorithm can be improved greatly and the frequent itemsets with better quality can be obtained. The performance of GAA-FIM algorithm is validated based on both synthetic data sets and real data sets. Experimental results show that the proposed GAA-FIM algorithm has a better time efficiency than GAFIM algorithm and GA-Apriori algorithm. Moreover, the quality of mining frequent itemsets has been further improved.

Keywords　Frequent itemset, Genetic algorithm, Bio-inspired, Downward closure property, Data mining

1　引言

频繁项集挖掘作为关联规则分析的重要步骤之一,在数据挖掘领域受到广泛关注。频繁项集挖掘的目标是发现事务数据集中频繁出现的,具有高度关联性的项目集。目前,研究人员已经提出了众多频繁项集挖掘算法,比如基于候选项集生成和测试的频繁项集挖掘算法,基于模式增长的频繁项集挖掘算法等,但是大多为精确频繁项集挖掘算法,可以获取到数据集中隐含的所有频繁项集[1]。

精确频繁项集挖掘算法通常需要多次扫描数据集,算法的运行时间会随着数据集规模的扩大呈指数型增长。对于海量数据集,精确频繁项集挖掘算法有时无能为力[2],比如社交网络数据集或大型生物信息数据集等。为了提高频繁项集挖掘的时间效率,近年来研究人员提出了基于生物启发的频繁项集挖掘算法,比如基于进化算法、群体智能算法的频繁项集挖掘策略。基于生物启发的频繁项集挖掘策略通常比精确频繁项集挖掘算法具有更好的时间效率,但是通常不能获得数据集中隐含的所有频繁项集。因此,使基于生物启发的频繁项集挖掘算法在提高时间效率的同时,进一步提升所获得频繁项集的质量,是一项极具挑战性的工作。

本文基于遗传算法提出一种生物启发频繁项集挖掘策略GAA-FIM(Genetic Algorithm combining Apriori Property

基金项目:国家自然科学基金(61972208);中国博士后科学基金(2018M640509)
This work was supported by the National Natural Science Foundation of China(61972208) and China Postdoctoral Science Foundation(2018M640509).
通信作者:赵学健(zhaoxj@njupt.edu.cn).

based Frequent Itemset Mining),该算法将遗传算法与精确频繁项集挖掘算法的向下闭包特性融合,通过交叉操作将具有良好遗传基因的个体优先加入到新一代候选种群中,并通过变异操作扩展新一代候选种群的规模,在保证算法时间效率的同时获取更佳质量的频繁项集。实验结果表明,GAA-FIM算法与当前基于生物启发频繁项集挖掘策略相比,具有更好的时间效率,频繁项集质量也得到了进一步提升。本文主要贡献如下:

(1)将精确频繁项集挖掘算法的向下闭包特性与遗传算法融合,给出了编码操作、交叉操作、变异操作和选择操作的具体操作规则。

(2)在编码操作过程中,将个体的编码位数设置为与初始种群规模大小相等,简化了编码操作;在交叉操作过程中,融合精确频繁项集挖掘策略的向下闭包特性进行交叉操作,将具有良好遗传基因的个体优先加入到新一代候选种群中;在变异操作过程中,通过设置候选种群规模控制系数动态调整候选新一代种群的规模,以获取更佳质量的频繁项集;在选择操作过程中,对当前种群规模进行自适应调整,进一步提升算法的时间效率。

(3)在合成数据集和真实数据集上对GAA-FIM算法与相关算法做了对比验证,同时评估和分析了算法中的参数对算法性能的影响,实验验证了算法的高效性。

2 相关工作

迄今为止,研究人员已经提出了众多的频繁项集挖掘算法,但是大多数均为精确频繁项集算法,比如前文提到的基于候选项集生成和测试的频繁项集挖掘算法、基于模式增长的频繁项集挖掘算法等。本文不再赘述精确频繁项集算法,主要对近似频繁项集挖掘策略进行梳理分析。基于生物启发的频繁项集挖掘作为近似频繁项集挖掘的主要策略之一,是指利用大自然中各种生物所体现出来的智能计算技术来帮助我们搜索项集空间,从而更加高效地获得频繁项集。基于生物启发的频繁项集挖掘策略可以进一步分为基于进化算法的挖掘策略和基于群体智能算法的挖掘策略。

目前,基于进化算法的频繁项集挖掘策略主要是基于遗传算法实现频繁项集的挖掘。Jacinto等率先提出了两种基于遗传算法的频繁项集挖掘策略——GENAR算法和GAR算法[3-4]。在此基础上,Alata等提出了AGA算法[5],Yan等提出了ARMGA算法[6]。ARMGA算法与AGA算法的区别主要体现在交叉操作上,ARMGA算法交叉操作采取的是两点交叉操作,AGA算法采取的是简单的单点交叉操作。但是,上述基于遗传算法的频繁项集挖掘策略均存在种群中个体大小不同的缺陷,导致交叉操作和变异操作的效率较低,种群中个体的表示方法需要进一步改进。Djenouri等针对基于生物启发的关联规则挖掘方法都会产生不可接受的规则这一问题,提出一种有效的基于遗传算法的频繁项集挖掘策略GA-FIM算法[7]。该算法的主要创新之处是引进了删除与分解策略以避免不可接受关联规则的产生,但是该算法完全没有考虑频繁项集挖掘自身的特点,没有与向下闭包特性很好地融合。此外,Djenouri等还提出了一种用于生物启发频繁项集挖掘算法的框架,并基于该框架提出了基于遗传算法的频繁项集挖掘算法——GA-Apriori算法和基于粒子群算法的PSO-Apriori算法[8]。改进后的GA-Apriori算法和PSO-Apriori融合了频繁项集的迭代特性,很好地解决了集中性搜索和多样化搜索的矛盾。Bagui等提出了一种基于遗传算法框架在概念漂移情况下适用于流数据的频繁项集挖掘算法。该算法探讨了概念漂移、滑动窗口大小和遗传算法约束之间的各种关系,给出了使用滑动窗口计算流数据中最小支持计数的公式,强调了窗口大小与每次漂移的事务比率是算法是否能够取得良好性能的关键[9]。

基于群体智能的频繁项集挖掘算法近年来也得到了广泛关注。Fong等指出,群体智能算法可以被很好地应用于数据挖掘领域,比如特征选择、聚类及频繁项集挖掘等领域[10]。Kuo等最早提出了一种基于蚁群算法的频繁项集挖掘策略对医疗数据进行了聚类和关联规则分析[11]。Wu等也基于蚁群算法提出了HUIM-ACS算法进行高效用频繁项集挖掘[12],HUIM-ACS算法能够将完整的项集空间映射到路径图中,并包含两次剪枝过程,能够保证在从起始点没有候选路径的情况下得到完整的高效用频繁项集。然而实验表明,基于蚁群算法的频繁项集挖掘策略的时间效率普遍较低。Kuo等提出了一种基于粒子群优化的频繁项集挖掘算法,该算法通过移动每个粒子的前点和后点对邻空间进行搜索,集中搜索效率明显提升。但该算法会产生大量的邻点,不利于多样化搜索[13]。针对该问题,Lin等同样基于粒子群优化提出了PSOFIM算法,以在集中性搜索和多样化搜索之间寻求平衡,进一步提升了搜索效率[14]。Djenouri等将改进蜂群算法应用于频繁项集挖掘,首先确定每个蜜蜂的探索区域,然后每个蜜蜂在所分配区域内挖掘频繁项集。每一轮中,蜜蜂在舞蹈区与其他蜜蜂相互交流,以便收敛于一个最好的频繁项集集合。实验表明该算法具有较好的时间效率[15]。Heraguemi等运用蝙蝠元启发算法进行频繁项集挖掘,提出了BATFIM算法。该算法在同一解空间范围内,运用一组蝙蝠群尝试发现相关频繁项集,实验表明该算法具有良好的时间性能[16]。Cao等[17]提出了一种基于封闭项集属性的多目标进化方法CP-MOEA,该方法利用封闭项目集属性指导种群在特定时间的进化。实验表明,该算法具有较好的时间效率和挖掘性能。

此外,研究员还提出了其他的近似频繁项集挖掘策略。Djenouri等[18]提出的基于元启发式的频繁项集并行挖掘框架CFIM,集成了3种元启发式算法CGA,CPSO和CBSO,频繁项集生成过程由多个节点并行执行,然后将挖掘结果发送到主节点,主节点负责完成合并,并通过考虑项目集的频率和多样性,保留高质量的项目集。Timur等[19]提出一种在大型事务数据库中近似挖掘频繁项集的新方法,该方法使用数据子集的频繁项集来近似整个数据集的频繁项集挖掘结果,能够高效地产生挖掘结果。但是该算法遗漏了一些真正的频繁项集。Ramesh等[20]在模糊关联规则挖掘的基础上提出一种新的频繁项集挖掘方法IFFP,用以发现导致乳腺癌的核心因素。分析发现,有丝分裂因子对乳腺癌的检测帮助较小,裸核指标对乳腺癌的发现具有较大的参考价值。实验表明,该方法在运行时间和内存使用方面具有良好性能。Fatemi等[21]基于聚类思想提出一种近似频繁项集挖掘算法,该算法将频繁项集挖掘问题转化为聚类问题,相似的事务被聚成一簇,每个簇心代表一个项目集,将其视为候选频繁项目集,通过计算

这些支持度,验证其是否为频繁项集。实验表明,该算法具有良好的执行速率和挖掘准确率。Yu 等[22]利用粗糙集理论,提出了一种新的近似频繁项集挖掘模型。该模型通过在所考虑的数据集上构建事务信息系统,提出了事务决策表,然后利用不可分辨关系计算支持度的上下近似,最后通过分而治之的方式,考虑基于支持度的准确性和定义的覆盖度,发现近似频繁项集。Wu 等[23]在 Type-2 模糊集理论下提出了一种快速的基于列表的多模糊频繁项集挖掘算法 LFFT2。实验结果表明,就执行时间和搜索空间中检查的候选频繁项集数量而言,LFFT2 算法明显优于传统的基于 Apriori 的频繁项集挖掘方法。Valiullin 等[24]提出一种适用于大型事务数据库的频繁项集挖掘方法,该方法使用数据库子集的频繁项集来近似完整数据集的结果,有效减少了假阴性频繁项集的数量,能够高效地产生高精度的结果,并且可以在分布式环境中实现。

综上表明,面对海量的大数据集,近似频繁项集挖掘策略成为研究热点。基于生物启发的频繁项集挖掘策略作为近似挖掘策略之一,受到广泛关注。然而,当前基于生物启发的频繁项集挖掘策略广泛采用了随机搜索策略,没有考虑频繁项集挖掘本身的特点,因此得到的频繁项集质量普遍不高,时间效率也有待进一步提升。

3 GAA-FIM 算法

3.1 问题描述

假设 D 是待挖掘的事务数据集,数据集 D 中包含一组事务集合,即 $D=\{T_1,T_2,\cdots,T_m\}$,并且每个事务包含一个特定的事务号 $q\in[1,m]$,m 为数据集 D 中的事务数量。数据集 D 中包含的项目集合为 $I=\{I_1,I_2,\cdots,I_n\}$,n 为数据集 D 中的项目数量,则有 $T_q\subseteq I$。项集 X 为项目集合 I 的子集,如果项集 X 包含 k 个不同的项目,称项集 X 为 k 项集。如果 $X\subseteq T_q$,称项集 X 出现在事务 T_q 中。为了进行频繁项集的挖掘,需要事先给出最小期望支持度阈值 ε,为频繁项集的判定提供依据。

针对事务数据集 D 的频繁项集挖掘问题,可进行如下形式化描述。在数据集 D 中,若用户给定了最小期望支持度阈值 ε,挖掘频繁项集即判断项集 X 是否满足 $\exp Sup(X)\geqslant\varepsilon\times|D|$ 的过程,其中 $\exp Sup(X)$ 为项集 X 的期望支持度,$|D|$ 表示数据集 D 中的事务数量。

3.2 算法描述

GAA-FIM 算法将遗传算法与精确频繁项集挖掘算法的向下闭包特性融合,并对编码、交叉、变异和选择过程进行了相应的改进,从而进一步提升频繁项集挖掘策略的时间效率,改善频繁项集挖掘的质量。GAA-FIM 算法包括以下 6 个步骤。

(1)初始化

首先,设置频繁项集集合 FIM 为空集,最小支持度阈值为 $min\text{-}support$,初次扫描数据集 D 获取频繁 1 项集集合 FIM_1,记录所有频繁 1 项集的支持度 $support(\{I_i\})$,将频繁 1 项集加入频繁项集集合。设置种群规模 $PSize=\lambda|FIM_1|\in(minPSize,maxPSize)$,其中 $minPSize$ 为最小种群规模,$maxPSize$ 最大种群规模,ξ 为种群规模控制系数(取值范围 0.1～0.9),λ 为候选种群规模控制系数(取值范围 2～10),$|FIM_1|$ 为频繁 1 项集包含的项目数量。接下来,在频繁 1 项集集合 FIM_1 中选择支持度最大的 $PSize$ 个频繁 1 项集构成当前种群 $CurP$。

(2)编码操作

对当前种群 $CurP$ 中的个体进行编码,编码方法遵循以下规则。

规则 1　种群中每个个体(频繁项集)由 m 个二进制位组成,其中 $m=PSize$。

规则 2　种群个体中若包含某个项目,则该项目所对应的二进制位为 1,否则为 0。比如当 $FIM_1=\{\{a\},\{b\},\{c\},\{d\},\{e\}\}$ 时,频繁 1 项集集合 FIM_1 中包含 5 个频繁 1 项集,则频繁项集 $\{a,c\}$ 对应的编码为 $\{1,0,1,0,0\}$。

(3)交叉操作

频繁项集的挖掘通常是一个循环递归的过程,频繁 k 项集是在频繁 $k-1$ 项集的基础上生成的;反之,频繁 k 项集的任意 $k-1$ 项子集也应该是频繁项集,这被称为频繁项集挖掘的向下闭包特性。遗传算法的交叉操作通常采用有性繁殖方式,即运用交叉算子,一个新个体由两个父辈个体通过交换个体的部分位获得。然而,对于频繁项集的挖掘问题,若采用传统的交叉方式,极可能违背频繁项集挖掘的向下闭包特性,无法获得新的频繁项集。因此,GAA-FIM 算法融合频繁项集挖掘的向下闭包特性进行交叉操作,通过在父方个体中注入 1 位母方个体最优遗传基因的方式生成候选频繁 k 项集。交叉操作遵循以下规则。

规则 1　选择当前种群 $CurP$ 中任意两个频繁项集 $indiv_i$ 和 $indiv_j$,个体 $indiv_i$ 作为父方个体,个体 $indiv_j$ 作为母方个体。若父方个体中第 k 位为 0,母方个体中第 k 位为 1,且在母方个体中第 k 位对应的项目所组成的频繁 1 项集具有最大的期望支持度,则将父方个体中第 k 位置为 1,生成新一代个体 $indiv_{ij}$,并将新个体 $indiv'_{ij}$ 加入候选新一代种群 $CanP$。比如当 $FIM_1=\{\{a\},\{b\},\{c\},\{d\},\{e\}\}$ 时,若 $indiv_i$ 对应的编码为 $\{1,0,1,0,0\}$,则其表示频繁项集 $\{a,c\}$;若 $indiv_j$ 对应的编码为 $\{0,1,0,0,1\}$,则其表示频繁项集 $\{b,e\}$。$indiv_i$ 中为 0 且 $indiv_j$ 中为 1 的位对应的项目组成的频繁 1 项集分别为 $\{b\}$,$\{e\}$,若频繁 1 项集 $\{b\}$ 具有最大的期望支持度,则将项目 b 对应的位置为 1,生成新一代个体 $indiv_{ij}=\{1,1,1,0,0\}$。

规则 2　新个体加入候选新一代种群 $CanP$ 时需要执行查重操作,即若候选新一代种群中已有该个体时,则不再重复加入。

(4)变异操作

交叉操作通过融合向下闭包特性,使得具有良好遗传基因的个体率先加入到新一代候选种群中,即将具有较高支持度的频繁项集率先加入到候选频繁项集。然而,通过交叉操作得到的新一代候选种群中最多有 $PSize$ 个新个体,这可能使得大量的频繁 k 项集并未包含在候选频繁 k 项集中。变异操作旨在通过适当扩展候选新一代种群的规模,获取更佳质量的频繁 k 项集。GAA-FIM 算法通过候选种群规模控制系数 λ(取值范围 2～10)控制候选种群规模。变异操作遵循以下规则。

规则 1　对于当前种群 $CurP$ 中任意个体 $indiv_i$,在个体 $indiv_i$ 的所有为 0 的位,随机选择 1 位,将该位置为 1,生成新一代个体 $indiv_i''$,并将新个体 $indiv_i''$ 加入候选新一代种群

$CanP$。重复执行该操作，直到候选种群 $CanP$ 规模大于等于 $\lambda PSize$。比如当 $FIM_1=\{\{a\},\{b\},\{c\},\{d\},\{e\}\}$ 时，若 $indiv_i$ 对应的编码为{1,0,1,0,0}，则其表示频繁项集{a,c}。个体 $indiv_i$ 的第 2 位、第 4 位和第 5 位为 0，假设随机选择第 5 位，将该位置为 1，生成的新一代个体对应的编码变为 $indiv_i''=\{1,0,1,0,1\}$。

规则 2 新个体加入候选新一代种群 $CanP$ 时需要执行查重操作，即若候选新一代种群中已有该个体时，不再重复加入。

(5)选择操作

选择操作的目的是从经过交叉、变异操作的候选新一代种群中选出适应度最好的 $PSize$ 个频繁项集，生成频繁 k 项集作为下一轮交叉、变异操作的初始种群。选择操作中，适应度即为频繁项集的期望支持度。选择操作遵循以下规则。

规则 1 对于候选新一代种群 $CanP$ 中的每个个体，通过扫描数据集，计算每个个体的适应度，并选择适应度最好的 $PSize$ 个个体形成新的当前种群 $CurP$。

规则 2 对新种群 $CurP$ 中的个体进行适应度判别，判别个体的适应度是否大于支持度阈值 $min\text{-}support$，若是则保留该个体，否则从当前种群 $CurP$ 中删除该个体，将当前种群中个体加入频繁项集集合 FIM，并对当前种群规模进行自适应调整。

(6)重复执行交叉、变异及选择操作，直至当前种群为空。

GAA-FIM 算法的伪代码如算法 1 所示。

算法 1 GAA-FIM 算法

```
Input: transactional database D
Output: the set of frequent itemsets FIM
1. set min-support, minPSize, maxPSize, ξ, λ
2. FIM←Ø
3. FIM_1 = FindOneFIM()
4. FIM = FIM ∪ FIM_1
5. PSize = ξ|FIM_1| ∈ (minPSize, maxPSize).
6. CurP = Top-PSize(FIM_1).
7. while CurP≠Ø do
8.     for each indiv_i in CurP do
9.         indiv_i' = Crossover(indiv_i)
10.        if indiv_i' not in CanP then
11.            CanP = CanP ∪ indiv_i'
12.        end if
13.    end for
14.    while |CanP| < λPSize do
15.    for each indiv_i in CurP do
16.        indiv_i'' = Mutation(indiv_i)
17.        if indiv_i'' not in CanP then
18.            CanP = CanP ∪ indiv_i''
19.        end if
20.    end for
21.    end while
22.    CurP = Selection(CanP)
23.    FIM = FIM ∪ CurP
24.    PSize = |CurP|
25. end while
26. return FIM
```

上述伪代码中，第 1－6 行执行初始化操作，主要实现了最小支持度阈值为 $min\text{-}support$、最小种群规模 $minPSize$、最大种群规模 $maxPSize$、种群规模控制系数 ξ(可控制种群规模介于最小种群规模 $minPSize$ 和最大种群规模 $maxPSize$ 之间)和候选种群规模控制系数 λ 的设置，并获取频繁 1 项集 FIM_1，设置种群规模 $PSize$，并生成当前种群 $CurP$。第 8－13 行执行交叉操作，把具有良好遗传基因的个体优先加入到新一代候选种群中。第 14－21 行执行变异操作，主要通过变异操作适当扩展候选新一代种群的规模，以使频繁项集尽可能包含在新一代候选种群中。第 22－24 行执行选择操作，从候选新一代种群生成频繁 k 项集，并动态调整种群规模大小，生成当前种群，为下一轮的频繁项集挖掘做好准备。算法 26 最终返回所有频繁项集。

4 实验及结果分析

4.1 实验设置及相关说明

接下来将对 GAA-FIM 算法在时间效率和挖掘质量两方面的性能进行验证分析。算法时间效率是指挖掘出特定数目的频繁项集所需要的 CPU 时间；挖掘质量用算法能够挖掘出的频繁项集数量占所有频繁项集数量的比例来描述。

实验平台配置为：Intel Core i7-4510U 2.6 GHz 处理器，8 GB 内存，Windows 7 64 位操作系统。实验采用的数据集为 chess，mushroom，pumbs_star，pumbs，retail，T10I4D100K，accidents 及 Kosarak 数据集。数据集的特征如表 1 所列，其中 $|D|$ 为数据集所包含的事务数，$|I|$ 为数据集所包含的项目数，$AvgLen$ 为平均每条事务包含的项目数。

表 1 数据集特征

Table 1 Characteristics of datasets

Dataset	$\|D\|$	$\|I\|$	$AvgLenG$
chess	3196	75	37
mushroom	8124	119	23
pumbs_star	49046	2088	50.5
pumbs	49046	2113	74
retail	88162	16470	10.3
T10I4D100K	100000	870	10.1
accidents	340183	468	33.8
Kosarak	990002	41270	8.1

将最小种群规模 minPSize 设置为 20，最大种群规模 maxPSize 设置为 200，对 *GAA-FIM* 算法性能随参数变化情况进行分析，并将 *GAA-FIM* 算法性能与 *Apriori* 算法、*GAFIM* 算法及 *GA-Apriori* 算法的性能进行对比。

4.2 参数对算法性能影响分析

本小节将通过 3 组实验分别分析 GAA-FIM 算法中参数设置对算法性能的影响，包括算法的时间效率和生成频繁项集的质量随最小支持度阈值 $min\text{-}support$、种群规模控制系数 ξ、候选种群规模控制系数 λ 的变化情况。

第一组实验将种群规模控制系数 ξ 设置为 0.2，候选种群规模控制系数 λ 设置为 4，分析了 GAA-FIM 算法时间效率和生成频繁项集的质量随最小支持度阈值 $min\text{-}support$ 的变化情况。实验过程中，chess，mushroom，pumbs_star，pumbs 及 accidents 数据集对应的最小支持度 $min\text{-}support$ 由 0.01 增大到 0.1；retail，T10I4D100K 及 Kosarak 数据集对应的最小支持度 $min\text{-}support$ 由 0.001 增大到 0.01。由表 2 可以看出，在所有数据集中，算法运行时间均随最小支持度阈值 $min\text{-}support$ 的增大而减小，但变化趋势不同。这表明，GAA-

FIM算法虽然是一种生物启发频繁项集挖掘策略,但是其与精确频繁项集挖掘算法一样,算法运行时间依然随最小支持度的减小而增。也就是说,为了提高频繁项集挖掘的时间效率,增大最小支持度阈值依然是最有效的方法。以pumbs数据集为例,当最小支持度由0.01增大到0.1时,算法运行时间由2 991.287 s减少到358.766 s,算法运行时间降低88.01%。由表3可以看出,所有数据集生成的频繁项集数量均随最小支持度阈值 *min-support* 的增大而减少。以pumbs数据集为例,当最小支持度由0.01增大到0.1时,生成频繁项集数量从4091个减少到797个,生成频繁项集的数量降低80.51%。综合表2和表3可以看出,算法的运行时间和生成频繁项集的数量存在近似正比例关系。

表2 运行时间($\xi=0.2,\lambda=4$)

Table 2 Runtime($\xi=0.2,\lambda=4$)

chess		mushroom		pumbs_star		pumbs		retail		T10I4D100K		accidents		Kosarak	
ms	*rt*	*ms*	*rt*	*ms*	*rt*	*ms*	*rt*	*ms*	*rt*	*ms*	*rt*	*ms*	*rt*	*ms*	*rt*
0.01	5.409	0.01	13.325	0.01	904.308	0.01	2991.287	0.001	573.785	0.001	52.722	0.01	1490.176	0.001	1354.635
0.02	5.245	0.02	11.299	0.02	564.411	0.02	1747.265	0.002	465.875	0.002	51.522	0.02	1182.469	0.002	1089.639
0.03	5.223	0.03	9.623	0.03	334.835	0.03	1152.899	0.003	422.161	0.003	41.517	0.03	817.125	0.003	737.761
0.04	4.928	0.04	8.801	0.04	249.194	0.04	933.477	0.004	380.883	0.004	33.680	0.04	689.838	0.004	446.433
0.05	4.891	0.05	7.432	0.05	210.277	0.05	692.317	0.005	329.171	0.005	30.921	0.05	672.682	0.005	320.896
0.06	4.728	0.06	6.714	0.06	192.156	0.06	648.731	0.006	295.815	0.006	28.708	0.06	555.279	0.006	225.377
0.07	3.912	0.07	5.662	0.07	159.940	0.07	482.968	0.007	218.807	0.007	27.294	0.07	468.952	0.007	184.790
0.08	3.851	0.08	4.971	0.08	144.202	0.08	467.281	0.008	165.478	0.008	25.486	0.08	402.941	0.008	151.681
0.09	3.613	0.09	4.773	0.09	133.553	0.09	369.905	0.009	133.722	0.009	22.369	0.09	388.236	0.009	132.566
0.1	3.526	0.1	4.696	0.1	130.781	0.1	358.766	0.01	111.716	0.01	20.846	0.1	272.921	0.01	103.344

表3 生成频繁项集数量($\xi=0.2,\lambda=4$)

Table 3 Number of frequent itemset($\xi=0.2,\lambda=4$)

chess		mushroom		pumbs_star		pumbs		retail		T10I4D100K		accidents		Kosarak	
ms	*fi*	*ms*	*fi*	*ms*	*fi*	*ms*	*fi*	*ms*	*fi*	*ms*	*fi*	*ms*	*fi*	*ms*	*fi*
0.01	234	0.01	366	0.01	2267	0.01	4091	0.001	2391	0.001	901	0.01	908	0.001	1766
0.02	234	0.02	296	0.02	1444	0.02	2574	0.002	1188	0.002	845	0.02	706	0.002	942
0.03	233	0.03	258	0.03	899	0.03	1897	0.003	728	0.003	765	0.03	557	0.003	536
0.04	206	0.04	233	0.04	686	0.04	1460	0.004	453	0.004	665	0.04	498	0.004	362
0.05	204	0.05	203	0.05	595	0.05	1239	0.005	318	0.005	587	0.05	485	0.005	259
0.06	178	0.06	188	0.06	561	0.06	1166	0.006	234	0.006	525	0.06	450	0.006	199
0.07	178	0.07	163	0.07	460	0.07	971	0.007	183	0.007	482	0.07	372	0.007	160
0.08	176	0.08	140	0.08	428	0.08	854	0.008	147	0.008	449	0.08	335	0.008	125
0.09	176	0.09	139	0.09	403	0.09	799	0.009	125	0.009	407	0.09	334	0.009	105
0.1	176	0.1	137	0.1	400	0.1	797	0.01	104	0.01	378	0.1	265	0.01	92

在第二组实验中,将种群规模控制系数 ξ 设置为0.2,最小支持度阈值 *min-support* 设置为0.01,分析了GAA-FIM算法性能随候选种群规模控制系数 λ(2~10)的变化情况。由表4可以看出,在所有数据集中,算法运行时间均随候选种群规模控制系数 λ 的增大而增大,但变化趋势不同。以pumbs数据集为例,当候选种群规模控制系数由2增到10时,算法运行时间由1222.6 s增至7130.6 s,近似呈线性增长;以Kosarak数据集为例,当候选种群规模控制系数由2增大到5时,算法运行时间由53.5 s增加到108.4 s,算法运行时间近似线性增长;但是当候选种群规模控制系数由5增大到10的过程中,由于候选种群规模控制系数的变化对候选频繁项集的数量不再产生影响,因此算法运行时间近似恒定不变。

表4 运行时间($\xi=0.2,min\text{-}support=0.01$)

Table 4 Runtime($\xi=0.2,min\text{-}support=0.01$)

(单位:s)

λ	chess	mushroom	pumbs_star	pumbs	retail	T10I4D100K	accidents	Kosarak
2	3.560	7.920	388.377	1222.592	70.120	11.382	728.885	53.487
3	4.591	9.504	746.312	2004.875	100.219	14.079	1154.689	85.315
4	5.379	13.406	969.913	2785.659	120.449	18.799	1582.761	99.884
5	5.494	14.713	1430.854	3455.889	150.799	24.728	1963.768	108.439
6	5.521	15.282	1710.753	4199.285	160.329	28.049	2176.726	109.934
7	5.815	16.130	2113.791	4869.291	170.559	33.228	2628.85	110.714
8	5.883	16.842	2414.812	5576.438	170.919	39.337	3006.523	111.174
9	5.889	17.399	2795.981	6305.506	180.419	42.118	3273.221	111.805
10	5.917	18.423	3200.750	7130.657	180.921	47.237	3355.829	112.494

由表5可以看出,在候选种群规模控制系数 λ 由2增大到10的过程中,数据集生成的频繁项集数量均随候选种群规模控制系数 λ 的增大而增大,但是对于不同的数据集,生成频繁项集数量的增幅不同。分别以pumbs数据集和accidents数据集为例,当候选种群规模控制系数由2增大到10时,pumbs数据集生成频繁项集的数量从3718个增大到4221个,增幅为13.53%;accidents数据集生成频繁项集数量从401个增大到1762个,增幅为3.39倍。究其原因,这是由于

不同的数据集中所包含的项目数不同，平均每条事务中所包含的项目数不同造成的。

综合表 4 和表 5 可以看出，算法的运行时间和生成频繁项集的数量存在近似正比例关系。

表 5 生成频繁项集的数量($\xi=0.2$, *min-support* $=0.01$)

Table 5 Number of frequent itemset($\xi=0.2$, *min-support* $=0.01$)

λ	chess	mushroom	pumbs_star	pumbs	retail	T10I4D100K	accidents	Kosarak
2	163	186	1713	3718	64	322	401	41
3	192	257	2043	4004	82	344	607	78
4	234	366	2267	4091	104	378	908	92
5	239	398	2339	4101	128	399	1151	105
6	241	425	2321	4177	143	418	1243	107
7	246	470	2448	4169	151	436	1417	107
8	246	521	2461	4176	151	459	1545	107
9	246	552	2508	4188	151	481	1720	107
10	246	601	2539	4221	151	498	1762	107

在第三组实验中，将候选种群规模控制系数 λ 设置为 4，最小支持度阈值 *min-support* 设置为 0.01，分析了 GAA-FIM 算法性能随种群规模控制系数 ξ(0.1～0.9)的变化情况。由表 6 可以看出，在所有数据集中，算法运行时间均随种群规模控制系数 ξ 的增大而增大；但变化趋势区别较大。以 chess 数据集为例，当种群规模控制系数由 0.1 增大到 0.9 时，算法运行时间持续由 2.8s 增加到 53.1s。由表 7 可以看出，在种群规模控制系数 ξ 由 0.1 增大到 0.9 的过程中，数据集生成的频繁项集数量在开始随种群规模控制系数 ξ 的增大而增大；但是当种群规模控制系数 ξ 增大到一定程度后，生成频繁项集数量趋于稳定，不再增大。以 pumbs 数据集为例，当种群规模控制系数 ξ 由 0.1 增大到 0.4 的过程中，生成频繁项集数量由 3365 个增大到 4168 个，增幅 23.86%；但是当种群规模控制系数 ξ 由 0.4 增大到 0.9 的过程中，生成频繁项集的数量没有变化。综合表 6 和表 7 可以看出，算法运行时间和生成频繁项集数量的变化趋势不再紧密吻合，因为种群规模的持续增大使得遗传算法单次迭代的复杂度增加，对算法运行时间产生较大影响。但是有的数据集由于所包含的项目数较多，当种群规模控制系数 ξ 增大到特定阈值时，种群规模超过最大种群规模 *maxPSize* 而不再变化，算法的运行时间也会趋于稳定，比如 pumbs_star，pumbs 和 T10I4D100K 数据集。

表 6 运行时间($\lambda=4$, *min-support* $=0.01$)

Table 6 Runtime($\lambda=4$, *min-support* $=0.01$)

(单位：s)

ξ	chess	mushroom	pumbs_star	pumbs	retail	T10I4D100K	accidents	Kosarak
0.1	1.012	2.765	388.737	781.964	40.247	9.679	309.607	15.376
0.2	4.981	12.951	1039.458	2756.695	110.150	19.209	1575.796	105.886
0.3	10.109	21.470	1102.937	3744.374	130.503	23.318	2233.710	122.574
0.4	18.879	26.595	1292.316	3805.976	160.361	24.309	2905.93	141.562
0.5	29.538	33.215	1319.710	3829.855	180.634	25.228	3586.257	254.856
0.6	38.754	37.801	1331.321	3833.346	200.388	25.949	4262.568	278.565
0.7	43.370	42.203	1364.309	3871.769	200.607	26.128	4687.303	284.341
0.8	48.872	49.153	1403.705	3890.119	230.315	26.759	4706.173	345.328
0.9	53.782	52.192	1444.505	3912.149	230.458	26.999	4707.097	356.924

表 7 生成频繁项集数量表($\lambda=4$, *min-support* $=0.01$)

Table 7 Number of frequent itemset($\lambda=4$, *min-support* $=0.01$)

ξ	chess	mushroom	pumbs_star	pumbs	retail	T10I4D100K	accidents	Kosarak
0.1	185	253	1683	3365	87	247	683	79
0.2	234	366	2267	4091	104	378	908	92
0.3	239	382	2304	4124	113	449	1055	101
0.4	239	413	2387	4168	117	457	1121	107
0.5	239	453	2387	4168	117	469	1162	107
0.6	239	453	2387	4168	117	477	1162	107
0.7	239	453	2387	4168	117	477	1162	107
0.8	239	453	2387	4168	117	477	1162	107
0.9	239	453	2387	4168	117	477	1162	107

4.3 算法性能对比分析

本小节将 GAA-FIM 算法的种群规模控制系数 ξ 设置为 0.2，候选种群规模控制系数 λ 设置为 4，最小支持度阈值 *min-support* 设置为 0.01，将 GAA-FIM 算法的时间效率和生成频繁项集的质量与 Apriori 算法、GAFIM 算法及 GA-Apriori 算法进行了对比分析。

图 1(a)给出了 pumbs_star，pumbs，retail，accidents 及 Kosarak 数据集上 GAA-FIM 算法与 Apriori 算法、GAFIM 算法及 GA-Apriori 算法的运行时间对比情况，图 1(b)给出了 chess、mushroom 及 T10I4D100K 数据集上 GAA-FIM 算法

与 Apriori 算法、GAFIM 算法及 GA-Apriori 算法的运行时间对比情况。由图 1 可以看出,在所有数据集上,GAA-FIM 算法相对于 Apriori 算法、GAFIM 算法及 GA-Apriori 算法,时间效率都得到了明显提升。以 pumbs 数据集为例,GAA-FIM 算法的运行时间为 2991.3 s,相对于 Apriori 算法的运行时间 8129.4 s 缩短了 63.2%,相对于 GAFIM 算法的运行时间 5002.8 s 缩短了 40.2%,相对于 GA-Apriori 算法的运行时间 3690.1 s 缩短了 18.9%。

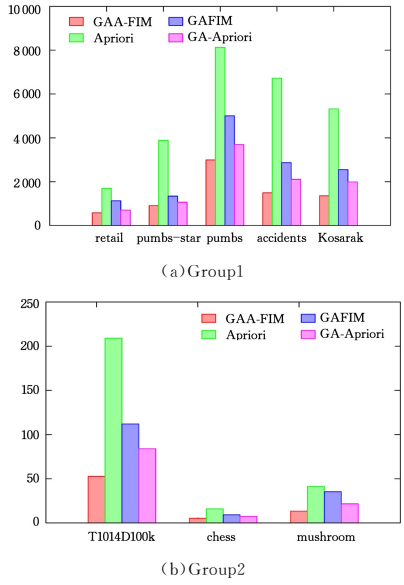

图 1 算法时间效率对比

Fig. 1 Runtime comparison

图 2 给出了 chess,mushroom,pumbs_star,pumbs,retail,T10I4D100K,accidents 及 Kosarak 数据集上 GAA-FIM 算法与 Apriori 算法、GAFIM 算法及 GA-Apriori 算法生成频繁项集比例的对比情况。由图 2 可以看出,Apriori 算法为精确频繁项集挖掘算法,可以获得数据集上的所有频繁项集。GAA-FIM 算法生成频繁项集数量占 Apriori 算法生成频繁项集数量的比例最高为 84.2%,最低为 65.3%,平均为 76.1%;GAFIM 算法生成频繁项集数量占 Apriori 算法生成频繁项集数量的比例最高为 70.2%,最低为 54.5%,平均为60.9%;GA-Apriori 算法生成频繁项集数量占 Apriori 算法生成频繁项集数量的比例最高为 79.8%,最低为 59.7%,平均为 69.4%。总而言之,GAA-FIM 算法生成频繁项集的质量相对于 GAFIM 算法和 GA-Apriori 算法均略有提升,其中相对于 GAFIM 算法生成频繁项集的比例提升约 24.9%,相对于 GA-Apriori 算法生成频繁项集的比例提升约 9.6%。

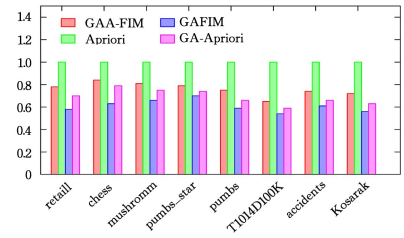

图 2 算法生成频繁项集的质量对比

Fig. 2 Quality comparison of frequent itemsets generated by algorithms

结束语 本文针对精确频繁项集挖掘算法时间效率低下,现有近似频繁项集挖掘策略虽然具有良好的时间效率,但是生成频繁项集质量有待提升的问题,提出一种基于遗传算法的生物启发频繁项集挖掘策略 GAA-FIM。所提策略将遗传算法与精确频繁项集挖掘算法的向下闭包特性融合,通过无性繁殖的交叉操作将具有良好遗传基因的个体优先加入到新一代候选种群中,并通过变异操作扩展新一代候选种群的规模,在保证算法时间效率的同时获取更佳质量的频繁项集。文章详细说明了编码操作、交叉操作、变异操作和选择操作的具体操作规则,给出了算法的伪代码,并在合成数据集和真实数据集上分析了参数对算法性能的影响,同时对比分析了所提出算法与现有算法时间效率和生成频繁项集质量情况,验证了算法的有效性。

参 考 文 献

[1] CHEE C H,JAAFAR J,AZIZ I A,et al. Algorithms for frequent itemset mining:a literature review[J]. Artificial Intelligence Review,2019,52(4):2603-2621.

[2] VALIULLIN T,HUANG Z,WEI C,et al. A new approximate method for mining frequent itemsets from big data[J]. Computer Science and Information Systems,2021,18(3):641-656.

[3] MATA J,ALVAREZ J L,RIQUELME J C. Mining numeric association rules with genetic algorithms[C]// Artificial Neural Nets and Genetic Algorithms. Springer,Vienna,2001:264-267.

[4] MATA J,ALVAREZ J L,RIQUELME J C. An evolutionary algorithm to discover numeric association rules[C]// Proceedings of the 2002 ACM Symposium on Applied Computing. 2002:590-594.

[5] ALATA B,AKIN E. An efficient genetic algorithm for automated mining of both positive and negative quantitative association rules[J]. Soft Cmputing,2006,10(3):230-237.

[6] YAN X,ZHANG C,ZHANG S. Genetic algorithm-based strategy for identifying association rules without specifying actual minimum support[J]. Expert Systems with Applications,2009,36(2):3066-3076.

[7] DJENOURI Y,NOUALI-TABOUDJEMAT N,BENDJOUDI A. Association rules mining using evolutionary algorithms[C]// The 9th International Conference on Bio-inspired Computing:Theories and Applications(BIC-TA 2014). LNCS,2014.

[8] DJENOURI Y,COMUZZI M. Combining Apriori heuristic and bio-inspired algorithms for solving the frequent itemsets mining problem[J]. Information Sciences,2017,420:1-15.

[9] BAGUI S,STANLEY P. Mining frequent itemsets from streaming transaction data using genetic algorithms[J]. Journal of Big Data,2020,7(1):54.

[10] FONG S,WONG R,VASILAKOS A V. Accelerated PSO swarm search feature selection for data stream mining big data[J]. IEEE Transactions on Services Computing,2015,9(1):33-45.

[11] KUO R J,LIN S Y,SHIH C W. Mining association rules through integration of clustering analysis and ant colony system for health insurance database in Taiwan[J]. Expert Systems with Applications,2007,33(3):794-808.

[12] WU J M T,ZHAN J,LIN J C W. An ACO-based approach to mine high-utility itemsets[J]. Knowledge-Based Systems,2017,116:102-113.

[13] KUO R J, CHAO C M, CHIU Y T. Application of particle swarm optimization to association rule mining[J]. Applied Soft Computing, 2011, 11(1): 326-336.

[14] 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.

[15] DJENOURI Y, DRIAS H, HABBAS Z. Bees swarm optimization using multiple strategies for association rule mining[J]. International Journal of Bio-Inspired Computation, 2014, 6(4): 239-249.

[16] HERAGUEMI K E, KAMEL N, DRIAS H. Multi-swarm bat algorithm for association rule mining using multiple cooperative strategies[J]. Applied Intelligence, 2016, 45(4): 1021-1033.

[17] CAO H, YANG S, WANG Q, et al. A Closed Itemset Property based Multi-objective Evolutionary Approach for Mining Frequent and High Utility Itemsets[C]// 2019 IEEE Congress on Evolutionary Computation (CEC). Wellington, New Zealand, 2019: 3356-3363.

[18] DJENOURIY, DJENOURI D, BELHADI A, et al. A Novel Parallel Framework for Metaheuristic-based Frequent Itemset Mining[C]// 2019 IEEE Congress on Evolutionary Computation (CEC). Wellington, New Zealand, 2019: 1439-1445.

[19] TIMUR V, HUANG Z J, WEI C, et al. A new approximate method for mining frequent itemsets from big data[J]. Computer Science and Information Systems, 2021, 18(3): 641-656.

[20] RAMESH D F, JEYASUTHA M. A Novel Fuzzy Frequent Itemsets Mining Approach for the Detection of Breast Cancer [J]. International Journal of Information Retrieval Research, 2021, 11(1): 36-53.

[21] FATEMI S M, HOSSEINI S M, KAMANDI A, et al. CL-MAX: a clustering-based approximation algorithm for mining maximal frequent itemsets[J]. International Journal of Machine Learning and Cybernetics, 2021, 12: 365-383.

[22] YU X, ZHAO J, WANG H, et al. A model of mining approximate frequent itemsets using rough set theory[J]. International Journal of Computational Science and Engineering (IJCSE), 2019, 19(1): 71-82.

[23] WU T, LIN J, YUN J, et al. An efficient algorithm for fuzzy frequent itemset mining[J]. Journal of Intelligent & Fuzzy Systems, 2020, 38(5): 5787-5797.

[24] VALIULLIN T, HUANG Z, WEI C, et al. A new approximate method for mining frequent itemsets from big data[J]. Computer Science and Information Systems, 2021, 18(3): 641-656.

ZHAO Xuejian, born in 1982, Ph.D, associate professor, is a member of China Computer Federation. His main research interests include data mining and wireless sensor networks.