软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2022,33(7):2599−2617 [doi: 10.13328/j.cnki.jos.006286]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

粗糙集多目标并行属性约简算法*

危前进[1], 魏继鹏[2], 古天龙[1], 常 亮[1], 文益民[1]

[1](广西可信软件重点实验室(桂林电子科技大学), 广西 桂林 541004)

[2](桂林电子科技大学 计算机与信息安全学院, 广西 桂林 541004)

通信作者: 危前进, E-mail: wei_qj@guet.edu.cn

摘 要: 粗糙集理论(RST)中, 求解最小属性约简 MAR (minimal attribute reduction)是一种 NP-难(non-deterministic polynomial hard)组合优化问题. 蚁群优化算法 ACO (ant colony optimization)是进化算法中的一种启发式全局优化算法, 粗糙集理论与 ACO 相结合, 是求解属性约简的一种有效、可行的方式. 针对蚁群优化算法易于陷入局部最优解、收敛速度慢等问题, 首先以一种改进的信息增益率作为启发信息, 提出了冗余检测机制, 对每个被选属性和每代最优约简集合进行冗余检测, 并提出了概率提前计算机制, 可避免每只蚂蚁在搜索过程中相同路径上的信息反复计算; 针对大数据集的属性约简问题, 考虑到蚁群优化算法具有并行能力以及粗糙集中"等价类"计算的可并行性, 提出一种将 ACO 与云计算相结合用于求解大数据集的属性约简算法, 在此基础上, 进一步提出一种多目标并行求解方案. 该方案可以同时计算出其余属性相对于当前属性或约简集合的重要度. 实验结果表明, 该算法在处理大数据的情况下能够得到最小属性约简, 计算属性重要度的时间复杂度由 $O(n^2)$降至 $O(|n|)$.

关键词: 蚁群优化; 属性约简; 粗糙集; 云计算

中图法分类号: TP18

中文引用格式: 危前进, 魏继鹏, 古天龙, 常亮, 文益民. 粗糙集多目标并行属性约简算法. 软件学报, 2022, 33(7): 2599–2617. http://www.jos.org.cn/1000-9825/6286.htm

英文引用格式: Wei QJ, Wei JP, Gu TL, Chang L, Wen YM. Multi-objective Parallel Attribute Reduction Algorithm in Rough Set. Ruan Jian Xue Bao/Journal of Software, 2022, 33(7): 2599−2617 (in Chinese). http://www.jos.org.cn/1000-9825/6286.htm

Multi-objective Parallel Attribute Reduction Algorithm in Rough Set

WEI Qian-Jin[1], WEI Ji-Peng[2], GU Tian-Long[1], CHANG Liang[1], WEN Yi-Min[1]

[1](Guangxi Key Laboratory of Trusted Software (Guilin University of Electronic Technology), Guilin 541004, China)

[2](School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin 541004, China)

Abstract: Solving the minimal attribute reduction (MAR) in rough set theory (RST) is an NP-hard combinatorial optimization problem. Ant colony optimization algorithm (ACO) is a globally heuristic optimization algorithm in evolutionary algorithms, so combining RST with ACO is an effective and feasible way to solve attribute reduction. The ACO algorithm often fall into local optimal solution, and the convergence speed is slow. This study first uses an improved information gain rate as heuristic information, and then deduction test is performed on each selected attribute and the optimal reduction set of each generation. And the mechanism of calculating probability in advance is proposed to avoid repeated calculation of information on the same path in the search process for each ant. But the algorithm can only handle small-scale data sets. The ACO algorithm has good parallelism and the equivalent classes in rough set theory can be calculated by cloud computing. This study proposes a parallel attribute reduction algorithm based on ACO and cloud computing to solve massive data sets and further investigate a multi-objective parallel solution scheme, which can simultaneously calculate the importance of the remaining attributes relative to the current attribute or reduction set. Experiments show that the proposed algorithm can obtain the MAR in the case of processing big data, and complexity of time on calculating the importance of attribute decreases from $O(n^2)$ to $O(|n|)$.

* 基金项目: 国家自然科学基金(U1811264, U1711263, 61966009, 61866007); 广西自然科学基金(2018GXNSFDA281045); 广西可信软件重点实验室研究课题(KX202024)

收稿时间: 2020-08-05; 修改时间: 2020-09-21, 2020-11-17; 采用时间: 2020-12-17; jos 在线出版时间: 2022-03-24

Key words: ant colony optimization (ACO); attribute reduction; rough set; cloud computing

粗糙集理论(rough set theory, RST)理论由波兰科学家 Pawlak[1]于 1982 年提出, 是一种处理不精确、不一致、不完整和不确定信息系统的有效数学工具. 粗糙集理论具有成熟的数学基础, 是一种数据挖掘和知识发现的有效方法. 粗糙集理论中, 属性约简是指不需要任何先验知识或辅助信息, 在不改变原始决策系统中条件属性和决策属性之间依赖关系的前提下, 即不改变原始决策信息中的决策或分类能力, 删除信息系统中所存在的冗余属性. 求解最小属性约简问题是一种 NP-难的非线性组合优化问题[2]. 属性约简是粗糙集理论中最重要的研究内容之一, 也是知识获取的关键步骤. 近年来出现了一些新的约简算法[3]. Liu 等人[4]提出一种利用属性区分度, 即属性之间的差异性进行约简的方法. Wang 等人[5]提出了邻域辨别指数用于表示邻域关系的区分信息, 它反映出特征子集的区分能力, 从而进行特征提取. Jing 等人[6]针对当数据动态变化时, 提出一种具有多粒度视图的动态增量式属性约简算法. 林冰雁等人[7]针对现实生活中不同的需求导致许多信息系统的属性值是基于直觉模糊数这一现象, 在加权得分函数的基础上建立了一种直觉模糊序关系, 给出一种不协调带偏好度量的直觉模糊序决策信息系统, 通过部分一致可辨识矩阵研究求解部分一致约简方法. 邓大勇[8]提出一种全粒度的粗糙集模型, 定义了全粒度属性约简, 探索各种全粒度属性约简的性质和各种全粒度属性约简之间的关系, 有助于实际应用和启发式算法的产生. 熊熙等人[9]利用最新的模糊集和粗糙集理论构造出一种数据预处理方法, 以减小模糊选项对属性提取算法性能的影响. Li 等人[10]在决策形式语境中, 从保持极大规则不变的角度出发, 研究了属性约简方法, 将属性约简应用于规则提取. Zhang 等人[11]基于模糊粗糙集信息熵, 提出一种面向增量特征属性选择的方法.

传统求解属性约简的算法诸如“盲目式增删法”或基于贪心策略的启发式方法, 在求解得到的属性约简集合中, 通常存在冗余信息[12], 且需要花费大量时间. 随着信息技术的发展和数据量的增大, 传统求解属性约简算法[13]面临数据规模和计算能力的挑战. 由于数据集太大而无法一次性装入到计算机内存以及单个计算机节点的计算空间有限, 因此在处理大数据的同时, 能够求得最小属性约简显得尤为重要, 这也是当今该领域研究热点之一. 为求解最小属性约简, 如何将属性约简算法与智能方法相结合, 是属性约简的研究方向之一[14]. 目前已有多种智能算法, 如粒子群算法、遗传算法、蚁群优化算法等, 应用于求解属性约简[15−17]. 考虑到蚁群优化是一种启发式全局优化算法, 具有分布式计算、信息正反馈和启发式搜索等特征. Jensen[18]首次提出将蚁群优化算法用于属性约简研究. Chen 等人[19]采用信息增益作为启发信息, 提出了基于蚁群优化的属性约简算法. 但容易将冗余属性作为被选属性添加到约简集合中, 难以求得最小属性约简.基于智能算法的属性约简面临智能算法本身所存在的一些缺陷, 如易于陷入局部最优解、收敛速度慢等.

传统数据挖掘方法通常只在单个计算机节点上求解问题, 很难构建足够的内存来适应大数据集, 很难在巨大的搜索空间中得到有效的解决方案. 基于增量的属性约简方法是处理大数据一种思路, Xie 等人[20]在特征选择即属性约简过程中, 基于图的方法提出一种动态式的增量属性选择方案, 用于处理大数据问题. 张钧波等人[21]结合云计算技术与增量方法, 提出一种基于粗糙集的并行增量知识更新算法.

近年来, 随着大数据和人工智能的兴起, 云计算技术是一种用于处理大数据的新兴计算模型. 并行计算是云计算技术的核心之一, 并且有一种简单的框架——MapReduce[22], 已经得到了广泛的关注. 它是一种用于在机器集群上并行计算大规模数据的编程模型. 因此, 钱进等人[23−25]在 MapReduce 框架下提出了基于正域、区分矩阵和信息熵等多种属性约简模型, 这些方法仅在大数据中对算法进行了性能测试. Hu 等人[26]构造出适用于多模糊态分类的一种大数据属性约简. 陆玉佳等人[27]提出了基于 MapReduce 改进的离散萤火虫算法用于属性约简, 采用多重分形作为确定最小属性约简的个数. Yang 等人[28]考虑到约简长度和重要度之间的关系, 基于粒子群提出一种属性约简算法. Dagdia[29]在属性的区分度基础上, 提出一种基于 Spark 面向大数据的属性约简方法. 吴信东等人[30]介绍并比较了 MapReduce 和 Spark 分布式平台. 李佳[31]等人针对多属性决策排序结果中“并列”决策现象的存在, 提出了基于属性权重的优势度排序方法. 张清华[32]等人对直接利用 X 的现有知识来构建近似集进行了探讨, 提出了 X 的近似集构建方法. 这些求解属性约简的并行方法是基于贪心策

略, 因此求解得到的最小属性约简集合中往往仍存在冗余属性. 考虑到多数现有算法很难求解最小属性约简, 约简集合中仍有冗余属性, 且当数据集规模增大时无法处理大数据集, 因此, 本文结合智能算法与并行技术, 提出一种新的方法用于处理大数据下的属性约简问题.

- 首先, 本文利用蚁群优化算法是一种启发式全局优化算法, 用于求解最小属性约简问题. 在此基础上, 采用冗余检测机制, 能够有效避免约简集合中存在冗余属性.
- 其次, 考虑到蚁群优化算法具有并行性以及在云计算平台下, 粗糙集理论中"等价类"计算的可并行实现, 本文以云计算 MapReduce 为编程模型, 结合蚁群优化算法, 提出一种适用于大数据属性约简研究.
- 最后, 在深入分析并行原理的前提下, 进一步提出一种多目标并行求解的新策略, 从而降低属性重要度的计算时间复杂度. 并在算法实现过程中采用概率提前计算机制, 避免了蚁群在求解过程中相同路径的反复计算问题.

本文通过选取UCI数据集、大规模人工数据集和真实数据集, 以验证算法在大数据下的属性约简效率. 通过与其他智能算法约简结果以及收敛速率进行对比验证, 证明本算法在可求解得到最小约简的同时, 能够快速收敛. 约简结果通过 C4.5 和朴素贝叶斯分类方法进行验证, 分别对约简前后的数据集进行分类精度检测, 以验证约简结果是否有效.

本文的贡献如下:

(1) 提出了蚁群优化算法与 MapReduce 并行机制相结合的方法, 用于求解大数据下属性约简问题.

(2) 冗余检测机制的设计. 在属性选择过程中, 对每个被选属性进行冗余检测, 在此基础上, 对每代最优解再次进行冗余检测.

(3) 引入被选属性概率提前计算机制. 将计算概率所需的信息素和启发信息值在每代蚂蚁求解之前提前计算, 可避免每只蚂蚁在搜索过程中相同路径上信息重复计算.

(4) 提出一种多目标并行求解新策略, 可同时计算出其余属性相对当前属性的重要度. 与改进前算法运行结果一致的前提下, 属性重要度计算时间随属性个数呈线性增长, 时间复杂度由 $O(n^2)$降至 $O(|n|)$.

本文通过实验验证了提出的算法能够有效处理大数据属性约简, 并能求得最小属性约简.

本文第 1 节介绍粗糙集理论、蚁群优化和 MapReduce 框架的基本概念. 第 2 节提出并行策略和多目标并行方案. 第 3 节给出实验结果以及算法的并行性. 第 4 节给出本文的结论.

1 相关基本概念

本节将回顾粗糙集理论中用于求解属性约简的基本定义和概念[1], 云计算中 MapReduce 编程模型[22]和蚁群优化算法的基本概念.

1.1 粗糙集理论

定义 1(知识系统概念)[1]. 粗糙集理论中, 一个决策信息系统 S 定义为 $S=(U,C\cup D,V,f)$, 其中, $U=\{x_1, x_2,\ldots,x_n\}$是一个有限的非空对象集合, 即论域. C 和 D 分别为非空有限的条件属性集和决策属性集, 交集为空; 对于任意属性子集 $B\subseteq C\cup D$ 决定了一个二元等价关系 $IND(B)$:

$$IND(B)=\{(x,y)\in U\times U|\forall a\in B,f(x,a)=f(y,a)\} \tag{1}$$

$U/IND(B)$或者 U/B 称为 U 的一个划分, 是论域 U 上的一个知识, 该等价划分包括多个等价类, 每个等价类称为一个知识粒.

定义 2[19]. 设 $S=(U,C\cup D,V,f)$为一个信息决策系统, 对任意属性子集 $B\subseteq C\cup D$, 令 $U/IND(B)=\{X_1,X_2,\ldots,X_n\}$表示由等价关系 $IND(B)$确定的划分, 则属性子集 B 的信息熵 $H(B)$定义为

$$H(B)=-\sum_{i=1}^{n} p(X_i)\log_2 p(X_i) \tag{2}$$

其中, $p(X_i)=|X_i|/|U|$, $1\leqslant i\leqslant n$, $|X_i|$为 X_i 的基.

定义 3[19]. 设 $S=(U,C\cup D,V,f)$为一个信息决策系统, $U/IND(C)=\{X_1,X_2,\ldots,X_n\}$和 $U/IND(D)=\{Y_1,Y_2,\ldots,Y_m\}$分别表示等价关系 $IND(C)$和 $IND(D)$确定的划分, 则条件属性 C 相对于决策属性 D 的条件熵定义为

$$H(D\mid C)=-\sum_{i=1}^{n}p(X_i)\sum_{j=1}^{m}p(Y_j)\log_2 p(Y_j\mid X_i) \tag{3}$$

其中, $p(X_i)=|X_i|/|U|$, $p(Y_j|X_i)=|X_i\cap Y_j|/|X_i|$, $1\leqslant i\leqslant n$, $1\leqslant j\leqslant m$.

定义 4[19]. 设 $S=(U,C\cup D,V,f)$为一个信息决策系统, $U/IND(C)=\{X_1,X_2,\ldots,X_n\}$和 $U/IND(D)=\{Y_1,Y_2,\ldots,Y_m\}$表示等价关系 $IND(C)$和等价关系 $IND(D)$确定的划分, 则 C 和 D 之间的互信息定义为

$$I(C;D)=H(D)-H(D|C) \tag{4}$$

互信息可用来衡量两个特征集合间的相关程度, 互信息值较大时, 说明两个特征集合具有较强的相关度.

定义 5[19]. 设 $S=(U,C\cup D,V,f)$为一个信息决策系统, 对于任意条件属性子集 $B\subseteq C$, 如果 $I(B;D)=I(C;D)$, 且对于任意 $b\in B$, $I(B-\{b\};D)<I(C;D)$, 则 B 称为在决策信息系统 S 中, 条件属性 C 相对于决策属性 D 的一个约简集合.

定义 6[33]. 设 $S=(U,C\cup D,V,f)$为一个信息决策系统, 对于任意条件属性子集 $B\subseteq C$ 和任意属性 $a\in C-B$, 属性 a 相对于 B 和 D 重要度定义为

$$\mathrm{sgn}(a,B,D)=\frac{I(B\cup\{a\};D)-I(B;D)}{H(D\mid\{a\})} \tag{5}$$

属性的重要度可作为贪心策略中的启发信息, 用于求解属性约简问题. 以信息增益作为属性重要度时, 对可取值数目较多的属性有所偏好; 而信息增益率作为重要度时, 对可取值数目较少的属性有所偏好. 公式(5)采用一种改进的信息增益率作为属性重要度, 不仅考虑了约简集合 B 添加被选属性, $a\in C-B$ 后的互信息增量, 还考虑了所选属性自身所包含的信息熵.

1.2 MapReduce编程模型

为了解决海量数据的存储和计算问题, Google 率先提供了 GFS 和 MapReduce 等云计算技术. MapReduce 是一种处理海量数据的并行编程模式. 用户不必关注 MapReduce 如何进行数据分割、负载均衡、容错处理等细节, 只需要将实际应用问题分解成若干可并行操作的子问题, 设计相应的 *Map* 和 *Reduce* 两个函数, 就能将自己的应用程序运行在分布式系统上. 其形式如下:

$$Map(key_1,value)\rightarrow list(key_2,value_2) \tag{6}$$

$$Reduce(key_2,list(value_2))\rightarrow list(value_3) \tag{7}$$

Map 函数是接收一组输入键值对$\langle key_1;value_1\rangle$作为输入数据, 对于每个分片, 以 key_2 为关键字产生多个列表. Reduce 函数将这些具有相同 key_2 值的列表进行合并, 对具有相同 key_2 的一组 *value* 值进行归并处理, 最终形成 $value_3$ 列表. 由于 MapReduce 提供了一种灵活且具有鲁棒性的框架, 能够自动将数据分成多个数据分片, 我们只需关注如何设计合适的 *key-value* 键值适用于决策信息表, 并将其应用于大数据情况下的并行知识约简.

1.3 云计算环境下并行计算

定义 7(数据分片)[21]. 对于一个决策信息系统 $S=(U,C\cup D,V,f)$, 令 $S_i\{i=1,2,\ldots,L\}$为一个子决策表, 如果满足: (1) $S=\bigcup_{1\leqslant i\leqslant L}S_i$; (2) $S_i\cap S_j=\varnothing$, $S_i\neq\varnothing$, $S_j\neq\varnothing$, 其中, $i,j=1,2,\ldots,L$ 且 $i\neq j$, 则 S_i 称为一个数据分片.

定理 1[23]. 对于一个决策信息系统 $S=(U,C\cup D,V,f)$, 令 $B\subseteq C$, $U/B=\{X_1,X_2,\ldots,X_k,\ldots,X_n\}$, $S_i\{i=1,2,\ldots,L\}$为决策信息系统 S 的一个数据分片, $S_i/B=\{X_{i1},X_{i2},\ldots,X_{ik},\ldots,X_{in}\}$, 则 $X_k=\bigcup_{i=1}^{L}X_{ik}(k=1,2,...,n)$.

证明: 如果关于 B 的任意两个对象相同, 则它们可以合并成一个等价类. 在不同数据分片中, 相同的等价类可合并为一个更大的等价类, 因此, 对于 $U/B=\{X_1,X_2,\ldots,X_k,\ldots,X_n\}$和 $S_i/B=\{X_{i1},X_{i2},\ldots,X_{ik},\ldots,X_{in}\}$, 有:

$$X_k=\bigcup_{i=1}^{L}X_{ik}(k=1,2,...,n).$$

由定义 2 和定义 3 可知, 信息熵和条件熵的计算需要求解等价类的数量. 此外, 等价类和等价类所包含对

象的个数类似于⟨*key*;*value*⟩形式的键值对. 等价类的计算可以独立地在每个数据分片中执行, 在并行计算中, 将每个数据分片中相同的等价类合并后得到的结果与直接在所有数据集中计算得到的结果一致. 因此, 可以在 MapReduce 框架下实现等价类的并行计算. □

1.4 基于蚁群优化的属性约简算法

蚁群优化是一种用来寻找优化路径的概率型算法, 具有分布式、正反馈等特点, 本质上是进化算法中的一种启发式全局优化算法. 而求解属性约简的本质是NP-难的组合优化问题. 因此, 结合蚁群优化算法用于求解属性约简是一个很好的选择.

1.4.1 局部解与信息素的更新

以蚁群优化算法为框架, 求解属性约简过程中, 每只蚂蚁随机选取某个起始属性节点, 根据信息素与启发信息所构建的概率公式选择下一个被选概率最大的属性, 被选属性概率公式定义为

$$p_{ij}^{k}(t)=\frac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}(t)}{\sum_{l\in allowed_k}\tau_{il}^{\alpha}(t)\eta_{il}^{\beta}(t)},\ j\in allowed_k \tag{8}$$

其中, t 表示第 t 代, k 表示第 k 只蚂蚁. $p_{ij}^{k}(t)$ 表示属性节点间转移概率, 即第 t 代蚂蚁 k 从属性 i 转移到属性 j 的概率. $\alpha>0$ 和 $\beta>0$ 两个参数代表路径上信息素浓度与启发信息的重要度, 取值范围在 0–1 之间. $allowed_k$ 表示蚂蚁 k 待选取的条件属性集. $\tau_{ij}(t)$表示第 t 代属性节点 i 和 j 之间的信息素浓度, 启发信息 $\eta_{ij}(t)$表示第 t 代蚂蚁从属性节点 i 转移到属性节点 j 的期望值. 信息素初始值 $\tau_{ij}(0)$由经验设定为某一常数, 如 0.5.

蚂蚁在行走过程中会在走过的路径上留下一定的信息素, 且随着时间的推移, 信息素会逐渐挥发, 浓度降低. 因此, 蚂蚁走过路径多的地方信息素浓度较大, 而蚂蚁在路径选择时, 会倾向于信息素浓度大的地方前进, 形成正反馈效果, 以达到属性节点间的最优路径, 即最小约简集合. 当每一代中所有蚂蚁完成搜索后, 属性节点之间的信息素浓度根据公式(9)进行更新:

$$\tau_{ij}(t+1)=\rho\tau_{ij}(t)+\Delta\tau_{ij}(t) \tag{9}$$

参数 $\rho(0\leqslant\rho\leqslant1)$为常量, 表示信息素的挥发率. $\Delta\tau_{ij}(t)$表示所有蚂蚁求解完约简集合时, 属性节点间信息素浓度的累积增加量, 定义为

$$\Delta\tau_{ij}(t)=\begin{cases}\sum\dfrac{q}{|R(t)|}, & \text{如果属性}i\text{到属性}j\text{被遍历过}\\ 0, & \text{否则}\end{cases} \tag{10}$$

其中, q 为给定常量参数, $|R(t)|$表示第 t 代最小属性约简集合的基. 当满足任意以下条件时, 则停止搜索:

(1) $I(C;D)=I(R_k;D)$, 其中, R_k 为第 k 只蚂蚁所构建的局部解, 即约简集合;

(2) 当约简集合所包含属性的个数大于当前最优解所包含属性的个数;

第 1 条停止搜索的条件代表当前约简集合与决策属性的互信息, 等价于所有条件属性与决策属性的互信息, 即已经满足约简条件; 第 2 条表示如果求解约简集合的所包含属性的个数大于当前全局最优解的所包含属性的个数时, 则无须再进行下一步搜索.

以信息增益作为启发信息会对取值数目较多的属性有所偏好, 而信息增益率作为启发信息则对取值数目较少的属性有所偏好. 本文以一种改进的信息增益率, 如公式(5)所示, 作为启发信息 η_{ij}. 这里令

$$\eta_{ij}=\mathrm{sgn}(j,i,D)=\frac{H(D|\{i\})-H(D|\{i\}\cup\{j\})}{H(D|\{j\})} \tag{11}$$

表示蚂蚁在属性 i 时, 需要计算其余未被选取属性 j, $j\in C-\{i\}$, 相对于当前属性 i 的重要度. 以改进信息增益率作为启发信息不仅考虑了添加被选属性后的互信息增量, 还考虑了所选属性自身所包含的信息熵. 在求解约简集合中, 有助于选取更加合理的属性.

为解决蚁群优化算法中自身所存在的易于陷于局部最优解、收敛速度慢等问题, 本文在每只蚂蚁求解约简集合过程中, 对每个被选属性进行冗余检测. 根据信息素和启发信息构成的概率公式, 如公式(8)所示, 选

取一个被选概率最大的属性, 设为 $b_k(b_k\in C-R_k)$, 通过判断当前约简集合 R_k 添加被选属性 b_k 前后与决策属性的互信息是否发生改变, 判断被选属性 b_k 是否冗余, 若 $I(R_k\cup\{b_k\};D)$等于 $I(R_k;D)$, 则说明属性 b_k 冗余.

证明: 首先, 通过公式(12)进行被选属性的冗余判断:

$$I(R_k\cup\{b_k\};D)-I(R_k;D)=H(D)-H(D|R_k\cup\{b_k\})-(H(D)-H(D|\{b_k\}))=H(D|\{b_k\})-H(D|R_k\cup\{b_k\}) \tag{12}$$

上述公式(12), 互信息最终转化为条件熵的计算. 以当前约简集合添加备选属性前后的互信息是否发生改变, 其本质是判断两者的条件熵是否相同或发生改变. 根据条件熵的定义, $H(Y|X)$表示在已知随机变量 X 的条件下, 随机变量 Y 的不确定性. 将条件熵应用于求解属性约间中, 则如定义(3)所示, 表示在已知条件属性集下决策属性熵的大小, 即不确定性大小. 熵越大, 说明该决策值越多、越混乱、不确定性高.

设 $H(D|\{b_k\})=A$, $H(D|R_k\cup\{b_k\})=B$, 若存在 $A=B$ 的现象, 则说明约简集合 R_k 添加备选属性 b_k 后, 决策属性集 D 的不确定性没有发生改变, 即备选属性 b_k 对决策结果没有产生影响, 说明备选属性 b_k 冗余. 证毕. □

在此基础上, 每代最优解中依次删除每个属性, 若与决策属性的互信息发生未发生变化, 说明最优解中该属性为冗余属性, 继续测试下一个属性; 否则, 将其添加回约简集合中. 通过对每个被选属性以及每代最优解进行冗余检测, 进一步消除了由于每只蚂蚁起始属性随机选取所带来的冗余可能, 有助于算法快速收敛于全局最优解, 即最小属性约简.

1.4.2 概率提前计算机制

在每一代中, 存在多只蚂蚁进行约简集合的求解. 不同的蚂蚁在求解过程中, 可能面临相同“路径”重复计算.

例如第 k 只蚂蚁, 其起始随机节点为 $c_{i=1}$ 属性, 为搜索下一个备选被选概率最大的条件属性, 需要依次计算当前属性 $c_{i=1}$ 与其余被选属性 c_j, j=2,3,4,5,6 之间由信息素 τ_{ij} 和启发信息 $\eta_{ij}(t)$构成的概率公式, 如公式(8)所示. 将被选概率最大的属性设为 $c_{j=2}$, 添加到约简集合中判断是否满足约简条件; 否则, 以 $c_{i=2}$ 为当前所在属性, 继续对其余被选属性 c_j, j=3,4,5,6 搜索, 直至满足约简条件. 如图 1 的虚线搜索路径所示.

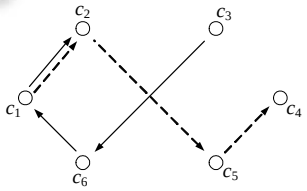


图 1 路径搜索

同代的第 k+1 只蚂蚁在求解约简集合过程中, 设当前搜索的约简集合为$\{c_3,c_6,c_1\}$, 此时需要计算当前属性 $c_{i=1}$ 其余被选属性 c_j, j=2,4,5 之间的信息素和启发信息, 选择被选概率最大的属性进行约简判断. 如图 1 的实线搜索路径所示.

通过上述描述过程可发现, 同代不同蚂蚁在搜索过程中, 当出现“交叉”节点时如 c_1 属性, 会对其余相同的备选属性进行信息素的重复计算.

需要说明的是, 信息素值会随着代数的变化不断更新, 但在每代求解过程中不会发生变化; 而启发信息则是由属性值本身所影响, 并不会随着代数的变化而更新. 为避免每一代不同蚂蚁在求解约简集合过程中面对相同“路径”的重复计算, 本文通过采取被选属性概率提前计算策略, 将启发信息提前至算法初始化阶段, 直接完成启发信息的计算. 从而在每代蚁群搜索被选属性求解约简集合时, 可直接使用启发信息值, 根据被选属性概率公式(8)得到下一个被选概率最大的属性, 避免了大量不必要的重复计算.

综上所述, 基于蚁群优化的属性约简算法流程图如图2所示. 在算法初始化阶段时, 完成了启发信息的计算和相关参数的初始化, 如蚁群个数、迭代次数等, 接下来对每代中各个蚂蚁进行属性搜索, 在搜索过程中, 同时进行冗余属性判断, 加快求解约简集合的效率, 每代蚁群求解之后进行信息素的更新, 直到最终求得最小属性约简. 该算法能够快速收敛于最小属性约简集.

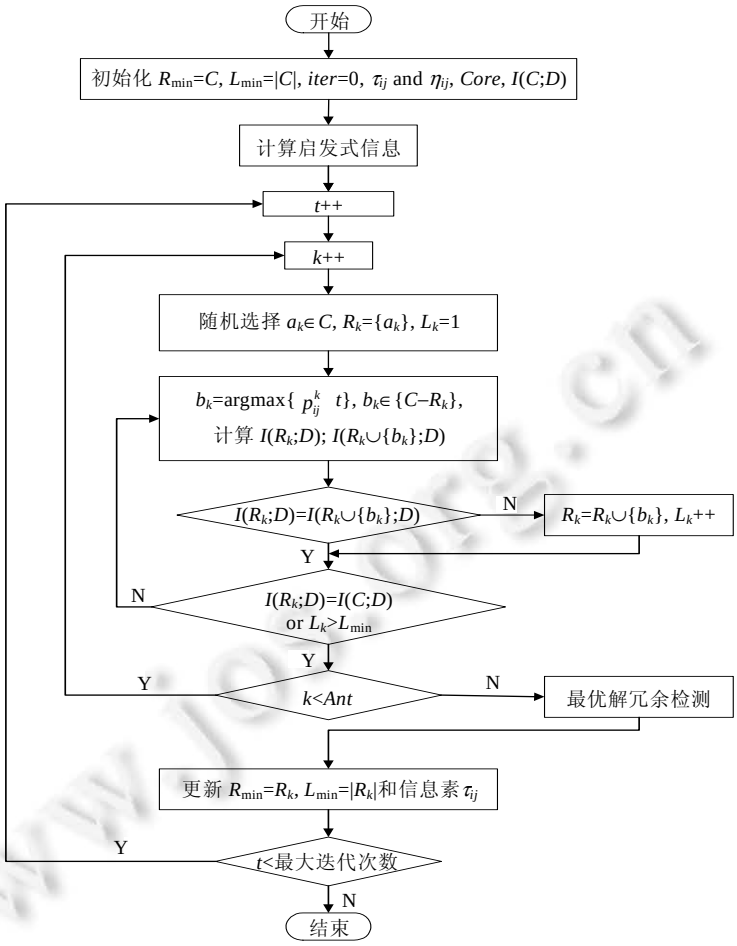


图 2 基于蚁群优化的属性约简算法流程图

2 基于云计算和蚁群优化的属性约简算法

由于蚁群优化算法具有并行能力以及粗糙集理论中等价类计算的可并行性, 通过云计算 MapReduce 并行技术, 用于解决大数据环境下的属性约简是一个很好的选择. 因此, 本文提出将蚁群优化与并行计算相结合, 用于处理大数据下的属性约简. 首先提出一种基于 MapReduce 与 ACO 的属性约简算法 ARMA (attribute reduction based on ACO and MapReduce). ARMA 以蚁群优化算法为框架求解属性约简, 具体计算方式采用云环境下的 MapReduce 分布式并行计算. 在求解属性约简过程中, 需要计算信息熵、条件熵等基本信息, 且由图 2 可知, 主要分为启发信息的计算和通过互信息判断属性是否冗余, 即互信息(条件熵)的计算, 通过一定迭代次数得到一个最优解. 根据公式(11), 启发信息同样可转换为条件熵的计算. 对此, 在处理大数据时, MapReduce 系统会自动将数据集进行“分块”, 发送至各个计算机节点上. 如何设计合适的 *Map*, *Reduce* 函数对各个数据“分块”进行操作, 实现熵或条件熵的计算, 是并行计算的难点.

2.1 云计算环境下算法设计

将启发信息计算的公式(11)展开, 可发现最终转化为条件属性相对于决策属性的条件熵计算. 同时判断是否满足约简条件 $I(C;D)=I(R_k;D)$, 即约简集合相对于决策属性的互信息是否等价于所有条件属性与决策属性的互信息. 同样可转为条件熵的计算, 即 $H(D|C)=H(D|R_k)$. 因此, 根据 MapReduce 编程模型设计出如下相应

的 *Map* 和 *Reduce* 函数, 用于求解等价类以便计算条件熵.

算法 1. Map_1.

Input: $S=(U,C\cup D,V,f)$, $S=\bigcup_{1\leqslant i\leqslant L}S_i$.

Output: (*key′*;*value′*), 其中, *key′*为条件属性和决策属性, *value′*=1.

Step 1. **for each** object x in S_i **do**

key′=(c_i,c_j,d); //c 和 d 表示条件属性及其取值和决策属性及其取值

value′=1;

output (*key′*;*value′*);

end for

算法 2. $Reduce_1$.

Input: ((c_i,c_j,d),*list*()).

Output: (*key′*;*value′*). //*key′*为条件属性, *value′*为 *list* 的和

Step 1. *sum*=0;

Step 2. **for** *i* in *list*() **do**

sum+=*list*[*i*];

end for

Step 3. *key′*=(c_i,c_j);

Step 4. *value′*=*sum*;

Step 5. output (*key′*;*value′*);

算法 3. Map_2.

Input: 读取 $Reduce_1$ 结果.

Output: (*key′*;*value′*).

Step 1. *key′*=(c_i,c_j), *value′*=*sum*;

Step 2. output (*key′*;*value′*);

算法 4. $Reduce_2$.

Input: ((c_i,c_j); *sum_list*()).

Output: (*key′*;*value′*). //*key′*表示条件属性, *value′*代表条件熵

Step 1. *sum*=0, *sig*=0, *val*[];

Step 2. **for** *i* in *sum_list*() **do**

val[*i*]=*sum_list*[*i*];

sum+=*sum_list*[*i*];

end for

Step 3. **for** *i*=0 To *val.size*() **do**

sig+=(*val*[*i*]/*sum*)*(log(*val*[*i*]/*sum*)/log(2));

end for

Step 4. Set *numofE=NumberOfElements*; //样本总数

Step 5. *sig=sig**(*sum*/*numofE*);

Step 6. *key′*=c_j, *value′*=*sum*;

Step 7. output (*key′*;*value′*);

算法 5. $Reduce_3$.

Input: (c_j;*list*()).

Output: (*key′*;*value′*). //*key′*为条件属性, *value′*代表 $H(D|\{c_j\})$

Step 1. *sum*=0;

Step 2. **for** *i* in *list*() **do**

sum+=*list*[*i*];

end for

Step 3. *key*′=c_j, *value*′=*sum*;

Step 4. output (*key*′;*value*′);

由算法 1 和算法 2 可知, Map_1 函数使用条件属性和决策属性作为 *key*, *value* 初始化值设为 1. 我们可计算每个数据分片中条件属性和决策属性所构成的等价类, 并把它们存储在本地文件中. MapReduce 框架将来自不同 Mapper 节点的相同等价类复制到相应的 *Reduce* 节点, 并对这些等价类进行求和, 如算法 2 所示. 算法 3 的 Map_2 读取算法 2 的输出结果, 并使用条件属性作为新的 *key*, *value* 为 *key* 的数量, 保持不变. 通过算法 4, 可以计算条件属性相对于决策属性的条件熵. 算法 4 中, 信息熵 $H(B)$和条件熵 $H(D|C)$的计算如公式(2)、公式(3)所示. 算法 5 是在多目标并行求解中, 同时输出多个条件熵结果.

2.2 多目标并行计算

在蚁群优化算法中, 需要依次计算每个属性相对于其余所有属性的相对重要度作为启发信息. 求解属性约简过程中, 重要度的计算占据了大部分时间. 假设当前所在的属性为节点 *i*, 根据公式(11), η_{ij}=sgn(*j*,*i*,*D*), 主要步骤为计算每个条件属性 *i* 并 *j* 相对于决策属性 *D* 的条件熵, 其中, *j*(*j*∈|*C*|–*i*), 需要计算|*C*|–1 次, 时间复杂度为 O(n^2). 通常在求解属性约简算法中[25−27], 需要计算每个属性的重要度, *Red* 表示约简集合, 选择重要度最大的属性 *c*, 将其添加到约简集合 *Red*=*Red*∪{*c*}. 重复该步骤, 直到达到约简效果, 其时间复杂度为 O(n^2).

基于数据分片机制和并行原理, 本文提出一种多目标并行求解新策略 M-ARMA (multi-objective parallel attribute reduction algorithm based on ACO and MapReduce). 该方法可以同时计算出其余属性 *j* 相对于当前属性 *i* 的条件熵.

假设存在决策信息系统 *S*, 见表 1.

表 1 决策信息系统

U	c_1	c_2	c_3	*d*
x_1	1	1	1	1
x_2	1	2	1	2
x_3	1	1	1	0
x_4	2	1	1	1
x_5	1	2	1	0
x_6	1	1	1	1

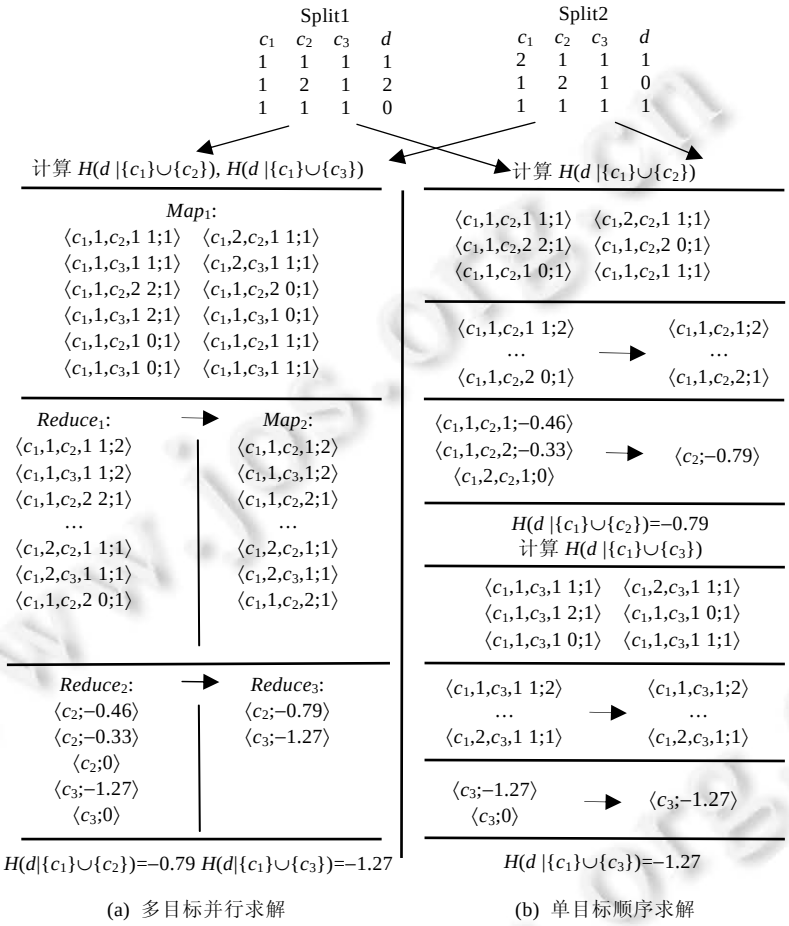
将数据集即决策信息系统 *S* 上传至分布式文件系统 HDFS (hadoop distributed file system)中, HDFS 会自动将决策信息系统切分为多个数据分片, 分布至各计算机节点, 计算机节点中一个为主节点, 其余为从节点. 通过定义的 Map_1,$Reduce_1$,…,$Reduce_3$ 函数, 同时求解出多个条件属性集合与决策属性的条件熵. 以表 1 为例, 其中, c_1, c_2 和 c_3 为条件属性, *d* 是决策属性. 假设 c_1 为蚂蚁当前所在属性, c_2 和 c_3 分别表示被选属性, 根据公式(3), 需要分别计算条件属性 c_1 和 c_2 或 c_3 与决策属性的 *d* 的条件熵, 即 $H(\{d\}|\{c_1\}\cup\{c_2\})$和 $H(\{d\}|\{c_1\}\cup\{c_3\})$. 单目标顺序求解条件熵与多目标并行求解过程对比如图 3 所示, 多目标并行求解过程如下.

Map_1 函数读取各节点中的数据分片, 并以⟨*key*;*value*⟩的形式进行存储. 以表 1 中第 2 行, 即对象 x_2 所包含的数据为例, *key*=“c_1,1,c_2,2 2”, 其中, “c_1,1”和“c_2,2”分别表示当前属性 c_1 及其取值和被选属性 c_2 及其取值, 以字符串方式写入, 最后把决策属性 *d* 的取值加入. 同理, 当被选属性为 c_3 时, *key*=“c_1,1,c_3,1 2”. Map_1 中所有 *key* 所对应的 *value* 初始化值设置为 1. 表 1 中其余对象所得结果如图 3 中 Map_1 阶段所示.

$Reduce_1$ 函数将各从节点按 Map_1 阶段所定义的键值对上传至主节点. 主节点对所有键值对进行等价类求和操作, 具体表现为将这些键值对中相同 *key* 值所对应的 *value* 进行求和操作. 例如在图 3 中, Map_1 中键值对⟨c_1,1,c_3,1 1;1⟩存在两个, 则 *key* 值保持不变, *value* 进行累加求和, 所得结果作为新的 *value*, 即⟨c_1,1,c_3,1 1;2⟩. 其

余结果以此类推.

Map_2 函数将 $Reduce_1$ 结果以新的键值对形式重定义，将 *key* 包含的决策属性值去除，其余保持不变. 如将⟨c_1,1,c_3,1 1;2⟩键值对 *key* 中决策属性值去除，得到新的键值对⟨c_1,1,c_3,1;2⟩. 根据算法 4 中 $Reduce_2$ 函数的定义，将上步结果中相同 *key* 值所对应的 *value* 以列表形式保存，当被选属性为 c_3 时，保存结果为⟨c_1,1,c_3,1;[2,1,2]⟩和⟨c_1,2,c_3,1;[1]⟩. 根据公式(2)和公式(3)，由算法 4 中第 1 行–第 10 行计算出不同属性值所对应的条件熵. 以被选属性为 *key*，所得条件熵为 *value*，如图 3 中 $Reduce_2$ 结果所示.



(a) 多目标并行求解 (b) 单目标顺序求解

图 3 MapReduce 中条件熵计算对比

最后，通过 $Reduce_3$ 函数对具有相同 *key* 值(属性)所对应的 *value* 进行求和操作，即将同一属性所包含的不同属性值得到的条件熵进行相加，最终可同时计算出所有被选属性与当前属性相对于决策属性的条件熵.以计算 $H(\{d\}|\{c_1\}\cup\{c_2\})$为例，求解过程如下所示.

(1) 在表 1 决策信息系统中，有 x_1–x_6 这 6 条决策过程，条件属性集$\{c_1,c_2\}$组成的决策条件存在 d={1,2,0};

(2) 当 d=1 时，有 3 条$\{c_1,c_2\}$组成的条件集满足，分别是 x_1:$\{c_1=1,c_2=1\}$, x_4:$\{c_1=2,c_2=1\}$, x_6:$\{c_1=1,c_2=1\}$，且存在 $x_1=x_6$. 根据公式(3), $p(d=1)=|3|/|6|$, $p(d=1|X_1=card(\{x_1,x_6\}))=|2|/|3|$, $p(d=1|X_2=card(\{x_4\}))=|1|/|3|$. 所以，满足 d=1 时，条件属性集$\{c_1,c_2\}$相对于决策属性$\{d\}$的条件熵为

$$H(\{d\}=1|\{c_1\}\cup\{c_2\})=-3/6(2/3\log_2 2/3+1/3\log_2 1/3)=-0.46.$$

(3) 当 d=2 和 d=3 时，重复上述第 2 步，依次计算条件属性集$\{c_1,c_2\}$相对于决策属性$\{d\}$的条件熵，即 $H(\{d\}|\{c_1\}\cup\{c_2\})=-(3/6(\log_2 2/3+1/3\log_2 1/3)+2/6(1/2\log_2 1/2+1/2\log_2 1/2)+1/6(1/1\log_2 1/1))=-(0.46+0.33+0)=-0.79$.

同理, $H(\{d\}|\{c_1\}\cup\{c_3\})=-1.27$.

与本文提出的算法 M-ARMA 相比, 文献[25,26]无论以信息熵、正域还是属性区分度等作为重要度,都需要依次计算所有被选属性相对于当前属性重要度值, 属性数目越多, 计算时间越大. 而本文设计的多目标并行求解方案, 利用其原理, 同样可扩展至其他算法中, 同时计算出其余属性相对于当前属性或约简集合的重要度, 使得求解属性重要度的时间复杂度由 $O(n^2)$降至 $O(|n|)$.

证明: 存在条件属性集 c_i, $i\in\{1,2,\ldots,n\}$, 设当前属性为 $c_{i=1}$, 计算其余属性 c_j, $j\in\{1,2,\ldots,n\}$且 $i\neq j$, 相对于当前属性 c_i的重要度, 需要计算 $n-1$ 次. 设下一个当前属性为 $c_{i=2}$, 此时, 其余属性 c_j, $j\in\{2,\ldots,n\}$且 $i\neq j$, 则需要计算 $n-2$ 次. 设完成一次条件熵的计算时间为 t, 算法 6 描述了计算过程, 如下所示.

算法 6. *ConditionalEntropy*.

Input: $S=(U,C\cup D,V,f)$.

Output: η_{ij}.

```
Step 1. S=0;
Step 2. for i=1 to n−1 do
Step 3.     for j=i+1 to n−1 do
            Calculate(η_ij);
            T=t;
            S=S+T;
           end for
         end for
Step 4. output η_ij;
```

这两重循环执行次数为 $f(n)=\sum_{i=1}^{n-1}\sum_{j=i+1}^{n-1}t=t((n-1)+(n-2)+\ldots+1)=t\times n\times(n-1)/2$, 即时间复杂度为 $O(n^2)$. 采用多个目标并行求解的策略, 在当前属性为 $c_{i=1}$时, 通过 Map/Reduce 计算组合, 仅需要 1 次计算, 就可同时得到其余属性 c_j 相对于当前属性 $c_{i=1}$ 的重要度, 因此总共只需要计算次数为 $f(n)=\sum_{i=1}^{n-1}t=t\times(n-1)$, 即时间复杂度为 $O(|n|)$. 证毕. □

3 实验评估

本节将对所提算法进行实验评估. 首先, 使用来自 UCI 和 MOA[34]的数据集(包括大数据集), 以验证算法的最小属性约简结果. 本文采用文献[19]以信息增益(RSFSACO)作为启发信息的蚁群优化算法作为对比, 同时与其他智能算法, 如文献[16]所采用的粒子群优化算法 MIPSO、文献[17]采用的遗传算法 ARAGA 对约简效果和收敛速率进行对比分析.

其次, 在 MapReduce 并行框架中, 将本文提出的多目标并行求解(M-ARMA)与单目标顺序求解(ARMA)进行实验对比, 以及将其他基于 MapReduce 的并行属性约简算法与本文算法进行运行时间对比, 以及各种影响因素分析验证. 最后给出了 M-ARMA 的在人工数据集下的加速比、扩展比以及磁盘读写分析实验[35].

3.1 实验设置

本文实验环境由 4 个节点组成的计算机集群构成, 其中一个设置为 master 主节点, 其余配置为 slaves 从节点. 每个节点具有 4 GB 主内存, 使用 Inter Core i7 CPU 和 Ubantu16.04, Hadoop 版本为 2.7.1, Java 1.8.0_ 121, 节点之间通信速率为 100 Mb/s. 在每个节点中设置 4 个 *map* 和 2 个 *reduce* 任务, 并将备份因子设置为 1.对于混合类型的数据集, 连续型数据则需要进行离散化处理; 而对于 Categorical data, 即描述对象的性质类型数据, 则可以将其映射为“1”“2”“3”等数字进行表示.

实验采用数据挖掘中常用的数据集进行实验. 从 UCI 中选取的一些数据集以验证算法约简效果, 本文使用 MOA 软件生成 LED 数据集, 以验证算法在大数据情况下的约简效果. 4 个人工数据集(DS1–DS6)及复制

Mushroom 数据集 5 000 次, 用于测试并行算法的有效性和正确性. 本文采用数据集 KDDCup-99, 该数据集包含近 500 万条记录、41 个条件属性和 1 个决策属性, 其中 6 个为离散型属性, 35 个为数值型属性. 需对 35 个数值型属性采用“等宽离散化”方法处理, 将其离散为 0–9 这 10 个数字, 用于对并行算法的性能进行测试.以及从 UCI 中选取的 HIGGS 数据集是用于区分产生希格斯玻色子的信号过程和不产生希格斯玻色子的背景过程, 该数据集使用蒙特卡洛模拟生成, 共 28 个属性. 通过属性约简方法可降低数据维度以减少冗余属性对粒子产生的影响. 数据集描述由表 2 列出.

表 2 数据集描述

Data sets	$\|U\|$	$\|C\|$	Data sets	$\|U\|$	$\|C\|$
Breastcancer	699	10	Wine	178	14
Chessking	3 196	37	Zoo	101	17
Audiology	200	70	Glass	214	10
Mushroom	8 124	23	Vote	435	17
DS1	10×10^6	31	DS2	20×10^6	31
DS3	30×10^6	31	DS4	40×10^6	31
Mushroom5000	40.62×10^6	23	KDD99	4.8×10^6	42
DS5	20×10^6	21	DS6	50×10^6	11
LED1	10×10^6	25	HIGGS	11×10^6	28

3.2 约简效果

为了验证所提算法有效性以及在大数据集下的约简效果, 首先在 8 个 UCI 数据集上进行实验. 同时, 我们分别通过 MOA 数据分析软件生成 1 000 万的 LED 数据集[34]. 每组数据集测试 10 次.

使用 WEKA 软件[36]上的 C4.5 分类器和朴素贝叶斯分类器测试原始数据集和属性约简后的数据集, 用以检测所提算法约简前后的分类效果, 分类准确性采用 10 折交叉验证方法进行检验. 各算法约简对比结果见表 3.

表 3 实验对比结果

Data sets	$\|U\|$	$\|C\|$	M-Redu.	M-ARMA	IEACO	RSFSACO	ARAGA	MIPSO
				Redu.	Redu.	Redu.	Redu.	Redu.
Breastcancer	699	10	4	4	$4^6 5^4$	4	4	4
Chessking	3 196	37	29	29	36	$29^4 30^6$	29	29
Mushroom	8 124	23	4	$4^7 5^3$	$4^5 5^5$	5	5	$4^6 5^4$
Vote	435	17	9	9	$12^3 13^7$	9	$9^7 10^3$	9
Wine	178	14	4	4	4	4	4	4
Zoo	101	17	5	5	$6^1 7^5 8^4$	$5^1 6^9$	$5^8 6^2$	5
Glass	214	10	5	5	$5^2 6^6 7^2$	$5^7 6^3$	$5^4 6^6$	5
Audiology	307	70	13	13	>20	$14^7 15^3$	14	>20

从表 3 中对不同数据集的约简效果可发现, 本文所提的 M-ARMA 算法求解最小属性约简集合的效果上均优于其他对比算法. 以第 2 行约简结果 $4^6 5^4$ 为例, 表示在每组数据集测试 10 次中, 有 6 次约简结果为 4 个条件属性, 有 4 次约简结果为 5 个条件属性. 不带上标则表示 10 次约简结果的条件属性个数均相同. 在 Audiology, Zoo 和 Glass 数据集, 对比算法中往往不能求得最小约简集合, 说明所求结果中仍存有冗余属性. 即使对比算法对于一些数据集能够求得最小约简集合, 如 Breastcancer, Glass 等, 但其通常需要迭代多次才能收敛与最优解, 具体实验通过第 3.3 节的收敛速度分析可证明.

从表 4 中的实验结果可发现, M-ARMA 算法对大多数数据集约简后在分类效果上几乎没有差异, 可以确保原始数据的分类有效, 即约简结果有效. 同时需要指出的是, MOA 会生成 LED 数据集包含 25 维属性特征, 7 个有效属性用以生成 7 段 LED 显示屏上的数字. 换句话说, 有 7 个相关的二进制属性来表示 0–9 这 10 个阿拉伯数字. 然而, 利用所提出的算法, 在约简之后, 仅需要 5 个属性就可区分 10 个数字, 效果如图 4 所示.

从图 4 中可以看出, 仅需要 5 段(属性)即可区分 10 个数字. 上述结果再次证明了本文设计的算法是有效的, 可求解得到 MAR.

表 4 约简前后分类精度测试

Dataset	M-ARMA	C4.5 classifier (%)		Naïve Bayes classifier (%)	
	Redu.	Before Redu.	After Redu.	Before Redu.	After Redu.
Breastcancer	4	94.849 8	95.422 0	95.708 2	94.277 5
Chessking	29	99.436 8	99.215 7	87.891 1	89.859 2
Mushroom	4	100	100	95.827 2	98.609 1
Vote	9	96.321 8	96.321 8	90.114 9	92.333 3
Wine	4	92.134 8	91.573 0	96.629 2	94.382 0
Zoo	5	92.079 2	94.059 4	95.049 5	90.099 0
Glass	5	66.822 4	69.626 2	48.598 1	47.196 3
Audiology	13	77.876 1	73.893 8	73.451 3	73.893 8

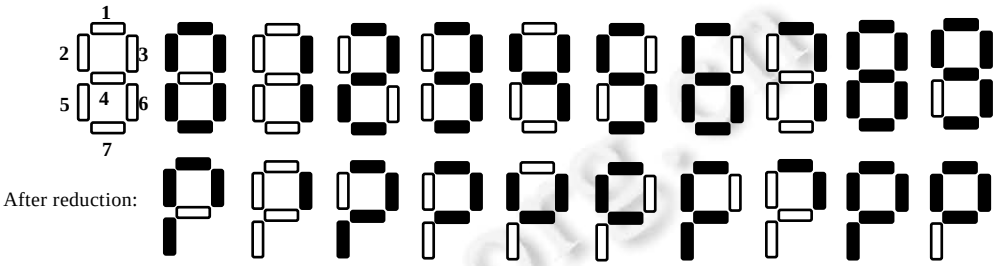


图 4 约简前后 LED 数字显示

3.3 收敛速度分析

为了比较本文算法与其余对比算法在求解过程中的收敛速度，本文采用一种适应度值计算. 该适用度值对约简效果以及约简集合长度进行加权组合，可反映出在智能算法中随着代数更新，求解最小属性约简的效率变化，定义如公式(13)所示.

$$fitness = \lambda \frac{I(R_k;D)}{I(C;D)} + (1-\lambda)\frac{|C|-|R_k|}{|C|} \tag{13}$$

实验通过选取几组 UCI 数据集对本文算法和对比算法进行收敛速度分析，效果如图 5 所示.

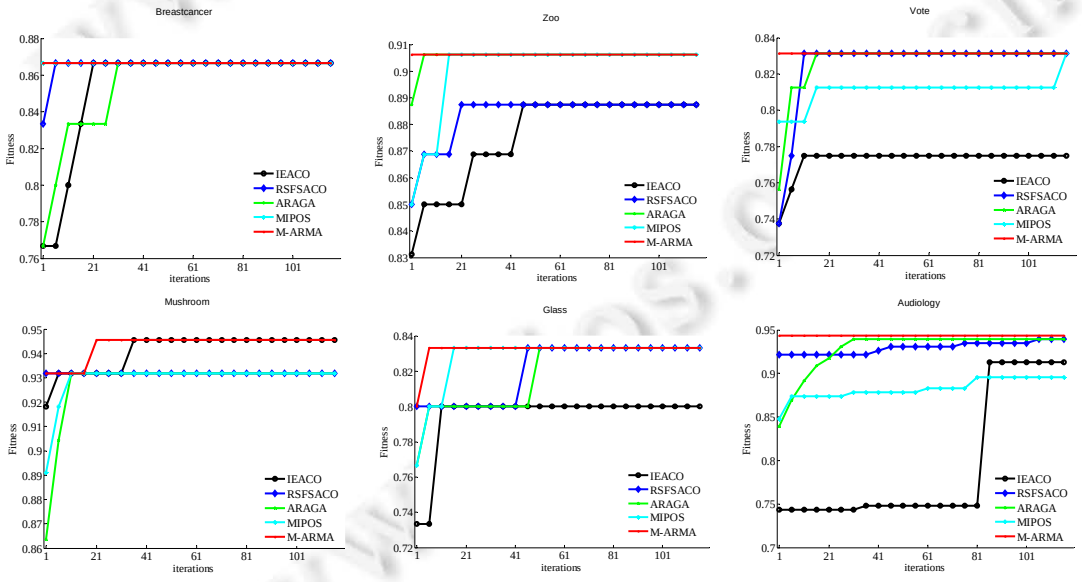


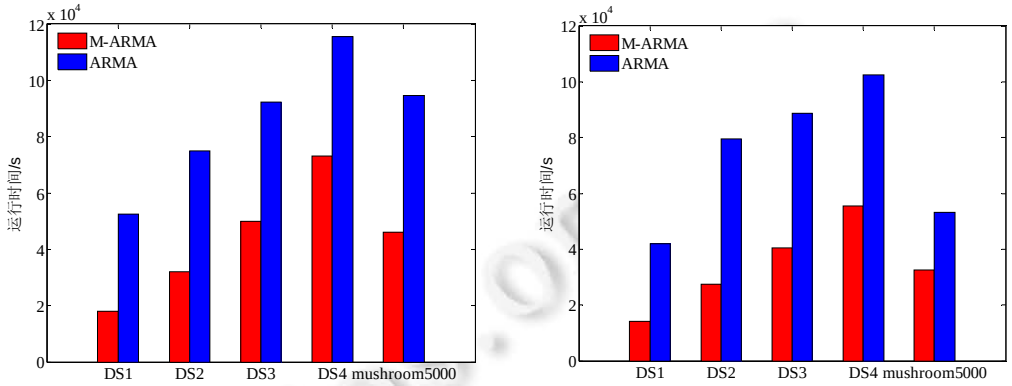
图 5 算法适应度值变化曲线对比

从图 5 可知，本文算法具有较强的收敛能力，在多数数据集下，仅需要较少的代数即可收敛于最优解. 通

过 UCI 数据集的实验可发现，无论从约简效果还是收敛速度上分析，本文所采用的策略均有助于快速求解得到最小属性约简，证明算法可行、有效.

3.4 大数据集下的运行时间

对于大数据集，本文提出的 M-ARMA 算法在基于 MapReduce 和 ACO 的属性约简过程中使用多目标并行求解策略，可以同时计算出当前属性与其余未被选取属性相对于决策属性的重要度. ARMA 算法只能依次计算出启发信息. 图 6(a)和图 6(b)显示了本文所设计的 M-ARMA 算法和 ARMA 算法分别在 2 个计算机节点和 4 个计算机节点、5 个数据集(DS1–DS4 和 Mushroom5000)的运行时间对比.



(a) node×2 (b) node×4

图 6 应用多目标并行求解策略前后运行时间对比

通过图 6 运行时间的对比，在保证运行结果一致的前提下，使用多目标并行求解的算法与原有算法相比，平均运行时间缩短一半以上，证明改进有效.

为了进一步证明 M-ARMA 算法，实验共从 3 个方面进行对比验证.

- 首先，本文通过对比文献[25]所提出的基于正域 HARAPOS 和信息熵 HARAInfo 的并行属性约简算法，以人工数据集 DS6 对各算法在面对不同被选属性数目时所消耗的时间，实验结果如图 7 所示.
- 其次，由于计算属性重要度的时间复杂度受条件属性个数影响，实验数据分别采用 5, 10, 20, 30 个条件属性和一个决策属性，每个数据集样本总数都为 2 000 万，用以验证不同条件属性个数下对所提算法以及其余对比算法运行时间的影响. 实验结果如图 8 所示.
- 最后，以入侵检测数据集 KDD99 和高能粒子碰撞实验数据集 HIGGS，对本文所提算法以及其他对比算法的总共运行时间进行对比验证，结果如图 8 所示.

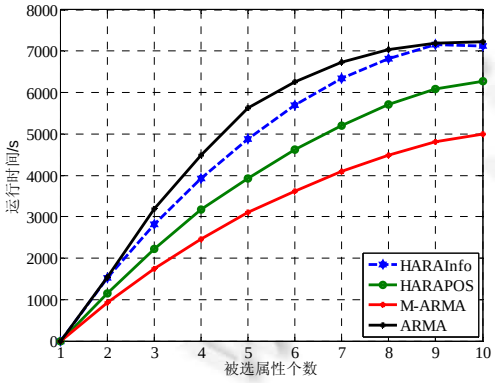


图 7 属性选择运行时间对比

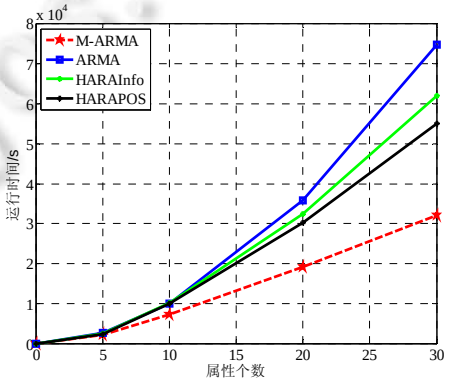


图 8 不同条件属性个数下运行时间对比

从图 7 结果可以分析出，本文所设计的 M-ARMA 算法在每个属性选择过程中采用多目标并行求解策略，在约简集合求解过程，面对不同数目被选属性，消耗时间以稳定速率增长. 说明被选属性的个数对本文所提 M-ARMA 算法的运行时间影响低于需要依次计算属性重要度的现有并行属性约简方法.

图 8 展示出，当条件属性的数量较少时，各算法运行时间较为接近. 由于计算属性重要度的时间复杂度受属性个数影响，且随属性数量的增加而增大，M-ARMA 算法的运行时间以稳定速率且呈线性增长，ARMA 算法的运行时间近乎成指数级增长, HARAPOS 和 HARAInfo 算法增长速率明显高于线性级. 说明现有并行约简算法在处理大数据时，计算属性重要度过程中，运行时间受条件属性数目影响较大. 本文所设计的多目标并行求解策略，将计算属性重要度的时间复杂度降低至线性级.

最后，以实际应用数据集 KDD 和 HIGGS 对各类算法在 3 个计算机节点下求得约简结果时运行时间进行对比，如图 9 所示. 从 KDD 数据集运行结果可知，该数据集只有约 500 万条数据，但其属性数目较多，共有 41 个条件属性，因此现有并行属性约简方法以及改进前的 ARMA 算法，需依次计算当前属性相对于其余被选属性的重要度，需要消耗大量时间；相比之下，M-ARMA 算法所需时间远低于对比算法. 粗糙集理论属性约简方法可应用于高能物理研究中的粒子碰撞实验，通过对数据集 HIGGS 的属性约简，可降低数据维度和复杂性. 结果显示，M-ARMA 算法在进行对每个被选属性以及每代最优解进行冗余检测的前提下，求解得到约简结果时，基于正域的并行属性约简算法 HARAPOS 时间略高于本文所提算法，而基于信息熵的并行属性约简算法 HARAInfo 和 ARMA 算法需较长时间才能得到一个约简结果.

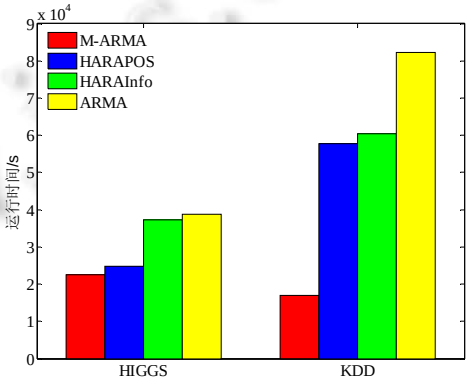


图 9　KDD 和 HIGGS 运行时间对比

3.5　算法性能测试

为了进一步评估基于 MapReduce 和蚁群优化的多目标并行属性约简算法，实验测试了算法在任务并行实验环境下的加速比、扩展比和单 Job 下磁盘 I/O 开销分析.

3.5.1　加速比

加速比是并行算法性能的重要衡量标准. 加速比测试是保持数据集不变而依次增加系统中的节点数，即集群中计算机的个数. 使用 m 台计算机在并行算法中的加速比定义如公式(14)所示.

$$Speedup(m)=\frac{\text{running time on 1 computer}}{\text{running time on } m \text{ computer}} \tag{14}$$

理想条件下，并行算法中可以获得线性加速比：具有 m 个计算节点的并行系统获得的加速比为 m. 由于集群中存在数据传输、通信成本、故障以及作业调度、监控和控制开销，很难实现理想条件下的线性加速比. 为测定加速比，保持数据规模不变，将计算机节点数量从 1 增加到 4. 图 10 显示了 M-ARMA 算法在数据集 DS1–DS5 中的测试结果，可以看出，实验结果很难接近于线性比. 原因是在计算条件熵的过程中产生了大量的中间文件，且从图 3 中可发现，与传统方法相比，即依次计算当前属性与其余被选属性的重要度, M-ARMA 产成中间文件明显多于对比算法，见表 5. HDFS 对中间缓存数据的 I/O 操作占用了大量时间，使得加速比难以

接近理想效果. 另外, 较小数据集加速比比值较低, 原因是部分节点处于空闲状态.

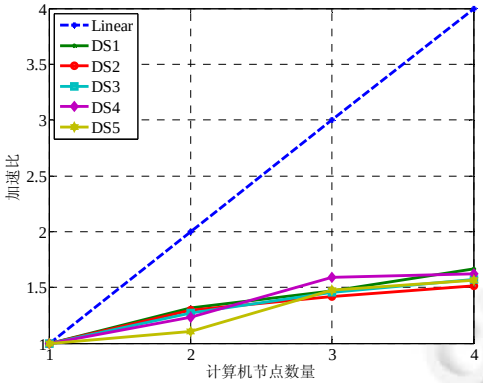


图 10 M-ARMA 算法加速比

表 5 单 Job 运行状态分析

	Map input records	Map output records/ Reduce input records	Reduce output records	Node×1 Total time spent by all map tasks (ms)	Node×3 Total time spent by all map tasks (ms)
M-ARMA	20 000 000	580 000 000	29 000	1 963 374	1 623 753
ARMA	20 000 000	20 000 000	1 000	109 103	89 510

3.5.2 可扩展性

算法的可扩展性又称扩展比, 是指数据规模与节点数成比例增大时算法的性能. 为了测试 M-ARMA 算法的可扩展性, 将与 ARMA, HARAPOS 和 HARAInfo 算法在处理更大规模数据集时进行对比. 数据集 DS1 分别被复制 2、3 和 4 倍, 并分别在 2、3、4 个计算机节点上运行. 图 11 所显示的扩展比结果表明, 本文所提 M-ARMA 算法扩展比效果优于对比算法, 具有良好的扩展性.

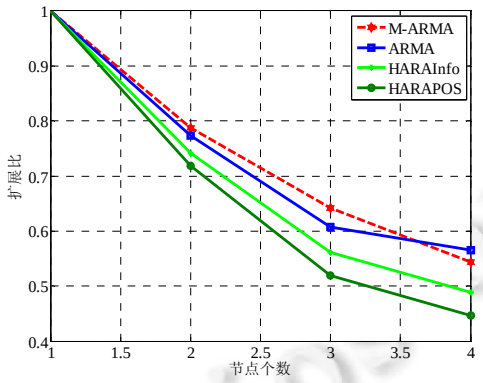


图 11 扩展比实验

3.5.3 磁盘 I/O 分析

为了确保所提算法的可行性, 以 DS2 数据集实验对象, 包括 2 000 万条样本以及 30 个条件属性和一个决策属性, 假设当前约简集合只有 1 个条件属性, 其余 29 个为未被选取的条件属性, 对比算法设置相同数量的 *mapper* 和 *reducer*. 对比实验仅获取 1 个 *map* 和 *reduce* 任务的磁盘读写情况来分析具体 I/O 开销大小. 从表 5 测试结果可知, 本文所提算法 M-ARMA 由于可同时计算其余未被选取的属性, 在 *map* 读取数据相同的前提下产生更多的中间缓存记录, 即等价类对象, 在串行统计所选属性对象个数时运行时间较长, 使得缓存文件在磁盘中的 I/O 读写及通信消耗了大量时间. 通过算法在 1 个节点和 3 个节点下 *map* 消耗时间对比, 3 个节点下 *map* 运行时间仅比 1 个节点略有下降, 间接证明所加速比难以接近理想效果. 同时需要指出的是, ARMA 算法

map 运行时间为选取 1 个被选属性, 在一个节点下时间为 1 090 103 ms, 完成所有属性重要度的计算, *map* 消耗时间需×29, 约为 3 163 987 ms. 同理, 3 个节点下完成重要度计算所需时间约为 89510×29=2595790 ms. 而在同样输入数据的状态下, M-ARMA 算法无论在单节点还是多个节点, *map* 阶段所需运行时间远低于 ARMA 算法.

4 总 结

本文针对传统的粗糙集属性约简算法求得最小属性约简集合中通常存在冗余属性的不足, 以及无法有效处理大数据, 提出利用蚁群优化算法和 MapReduce 相结合的方式用于解决最小属性约简问题. 本文在此基础上进一步提出了一种新型的多目标并行求解的策略, 与原并行算法以及其他同类型云计算环境求解属性约简的算法相比, 在保持结果一致的前提下, 计算属性重要度的时间复杂度由平方级降低至线性级. 通过真实数据以及大数据集的实验结果表明, 本文所提的 M-ARMA 算法在能够求解得到最小属性约简的同时, 能够在云计算平台下处理海量数据. 最后, 本文对提出的算法进行加速比与扩展比的性能测试.

同时, 本文所提算法也存在一些不足, 如多目标并行求解策略, 是以空间换取时间方式将时间复杂度降至线性级, 在计算过程中会产生中间缓存文件. 文中各参数的设置选取多为经验值, 需要通过大量测试调参得到一个合适值. 未来将进一步考虑如何在云计算环境下提高算法的性能和效率, 同时面对离散后数据的信息丢失, 如何针对大数下连续型数据的属性约简进行研究探讨.

References:

[1] Pawlak Z, Skowron A. Rudiments of rough sets. Information Sciences, 2006, 177(1): 3−27. [doi: 10.1016/j.ins.2006.06.003]

[2] Wei JP, Wei QJ, Wen YM. Attribute reduction algorithm based on improved information gain rate and ant colony optimization. In: Proc. of the Pacific-Asia Conf. on Knowledge Discovery and Data Mining. Melbourne, 2018. 139−150. [doi: 10.1007/978-3-319-93040-4_12]

[3] Fan J, Jiang YL, Liu Y. Quick attribute reduction with generalized indiscernibility models. Information Sciences, 2017, 397-398: 15−36. [doi: 10.1016/j.ins.2017.02.032]

[4] Liu JH, Lin YJ, Lin ML, *et al*. Feature selection based on quality of information. Neurocomputing, 2017, 225: 11−22. [doi: 10.1016/j.neucom.2016.11.001]

[5] Wang CZ, Hu QH, Wang XZ, *et al*. Feature selection based on neighborhood discrimination index. IEEE Trans. on Neural Networks and Learning Systems, 2017, 29(7): 2986−2999. [doi: 10.1109/TNNLS.2017.2710422]

[6] Jing YG, Li TR, Huang JF, *et al*. An incremental attribute reduction approach based on knowledge granularity under the attribute generalization. Int'l Journal of Approximate Reasoning, 2016, 76: 80−95. [doi: 10.1016/j.ijar.2016.05.001]

[7] Lin BY, Xu WH, Yang Q. Partially consistent reduction in intuitionistic fuzzy ordered decision information systems with preference measure. Computer Science, 2018, 45(1): 148−151, 187 (in Chinese with English abstract).

[8] Deng DY. Attribute reduction for entire-granulation rough sets. Pattern Recognition and Artificial Intelligence, 2018, 31(3): 230−235 (in Chinese with English abstract). [doi: 10.16451/j.cnki.issn1003-6059.201810008]

[9] Xiong X, Qiao SJ, Han N, Yuan CA, Zhang HQ, Li BY. A fuzzy-option based attribute discriminant method. Chinese Journal of Computers, 2019, 42(1): 190−202 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2019.00190]

[10] Li LJ, Mi JS, Xie B. Attribute reduction based on maximal rules in decision formal context. Int'l Journal of Computational Intelligence Systems, 2014, 7(6): 1044−1053. [doi: 10.1080/18756891.2014.963972]

[11] Zhang X, Mei CL, Chen DG, *et al*. Active incremental feature selection using a fuzzy rough set-based information entropy. IEEE Trans. on Fuzzy Systems, 2019, 28(5): 901−915. [doi: 10.1109/TFUZZ.2019.2959995]

[12] Wang CZ, Qi YL, Shao MW, *et al*. A fitting model for feature selection with fuzzy rough sets. IEEE Trans. on Fuzzy Systems, 2017, 25(4): 741−753. [doi: 10.1109/TFUZZ.2016.2574918]

[13] Chen H, Yang JA, Zhuang ZQ. The core of attributes and minimal attributes reduction in variable precision rough set. Chinese Journal of Computers, 2012, 35(5): 1011−1017 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2012.01011]

[14] Yu H, Wang GY, Yao YY. Current research and future perspectives on decision-theoretic rough sets. Chinese Journal of Computers, 2015, 38(8): 1628−1639 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2015.01628]

[15] Min F, Zhang ZH, Dong J. Ant colony optimization with partial complete searching for attribute reduction. Journal of Computational Science, 2017, 25: 170−182. [doi: 10.1016/j.jocs.2017.05.007]

[16] Xu XY, Zhang K, Xie J, Xie G. An attribute reduction algorithm based on mutual information of particle swarm optimization. Acta Electronica Sinica, 2017, 45(11): 129−138 (in Chinese with English abstract). [doi: 10.3969/j.issn.0372-2112.2017.11.017]

[17] Cheng Y, Zheng ZR, Wang J, *et al*. Attribute reduction based on genetic algorithm for the coevolution of meteorological data in the industrial Internet of things. Wireless Communications and Mobile Computing, 2019, 2019(1): 1−8. [doi: 10.1155/2019/3525347]

[18] Jensen R, Shen Q. Finding rough set reducts with ant colony optimization. In: Proc. of the UK Workshop Conf. on Computational Intelligence. Bristol, 2003. 15−22.

[19] Chen YM, Miao DQ, Wang RZ. A rough set approach to feature selection based on ant colony optimization. Pattern Recognition Letters, 2010, 31(3): 226−233. [doi: 10.1016/j.patrec.2009.10.013]

[20] Xie XJ, Qin XL. Dynamic feature selection algorithm based on minimum vertex cover of hypergraph. In: Proc. of the Pacific-Asia Conf. on Knowledge Discovery and Data Mining. Melbourne, 2018. 40−51. [doi: 10.1007/978-3-319-93040-4_4]

[21] Zhang JB, Li TR, Pan Y, Luo C, Teng F. Parallel and incremental algorithm for knowledge update based on rough sets in cloud platform. Ruan Jian Xue Bao/Journal of Software, 2015, 26(6): 1064−1078 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4590.htm [doi: 10.13328/j.cnki.jos.004590]

[22] Smith C, Albarghouthi A. MapReduce program synthesis. ACM SIGPLAN Notices, 2016, 51(6): 326−340. [doi: 10.1145/2908080.2908102]

[23] Qian J, Miao DQ, Zhang ZH, *et al*. Parallel attribute reduction algorithms using MapReduce. Information Sciences, 2014, 279: 671−690. [doi: 10.1016/j.ins.2014.04.019]

[24] Qian J, Miao DQ, Zhang ZH. Knowledge reduction algorithms in cloud computing. Chinese Journal of Computers, 2011, 34(12): 2332−2343 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2011.02332]

[25] Qian J, Lv P, Yue XD, *et al*. Hierarchical attribute reduction algorithms for big data using MapReduce. Knowledge-based Systems, 2015, 73(1): 18−31. [doi: 10.1016/j.knosys.2014.09.001]

[26] Hu QH, Zhang LJ, Zhou YC, *et al*. Large-scale multi-modality attribute reduction with multi-kernel fuzzy rough sets. IEEE Transactions on Fuzzy Systems, 2018, 26(1): 226−238. [doi: 10.1109/TFUZZ.2017.2647966]

[27] Lu YJ, Ni ZW, Zhu XH, Xu LF, Wu ZJ. Attribute reduction method based on MapReduce-based improved discrete glowworm swarm algorithm and multi-fractal dimension. Pattern Recognition and Artificial Intelligence, 2018, 31(6): 537−547 (in Chinese with English abstract).

[28] Yang J, Fong S, Li T. Attribute reduction based on multi-objective decomposition-ensemble optimizer with rough set and entropy. In: Proc. of the 2019 Int'l Conf. on Data Mining Workshops (ICDMW). 2019. 673−680. [doi: 10.1109/ICDMW.2019.00102]

[29] Dagdia Z, Zarges C, Beck G, *et al*. A distributed rough set theory based algorithm for an efficient big data pre-processing under the spark framework. In: Proc. of the 2017 IEEE Int'l Conf. on Big Data (Big Data). 2017. 911−916. [doi: 10.1109/BigData.2017.8258008]

[30] Wu XD, Ji SW. Comparative study on MapReduce and Spark for big data analytics. Ruan Jian Xue Bao/Journal of Software, 2018, 29(6): 1770−1791 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5557.htm [doi: 10.13328/j.cnki.jos.005557]

[31] Li J, Liang JY, Pang TJ. A sorting method of multi-attribute decision making based on dominance rough set theory. Journal of Nanjing University (Natural Sciences), 2016, 52(5): 844−852 (in Chinese with English abstract).

[32] Zhang QH, Xue YB, Wang GY. Optimal approximation sets of rough sets. Ruan Jian Xue Bao/Journal of Software, 2016, 27(2): 295−308 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4854.htm [doi: 10.13328/j.cnki.jos.004854]

[33] Yan Y, Yang HZ. Knowledge reduction algorithm based on mutual information. Journal of Tsinghua University (Science and Technology), 2007, 47(2): 1903−1906 (in Chinese with English abstract). [doi: 10.3321/j.issn:1000-0054.2007.z2.042]

[34] Bifet A, Holmes G, Kirkby R, *et al*. MOA: Massive online analysis. Journal of Machine Learning Research, 2010, 11(2): 1601−1604. [doi: 10.1007/s10846-009-9361-7]

[35] Xi DC, Wang GY, Zhang XR, *et al*. Parallel attribute reduction based on MapReduce. In: Proc. of the 9th Int'l Conf. on Rough Sets and Knowledge Technology. Shanghai, 2014. 631−641. [doi: 10.1007/978-3-319-11740-9_58]

[36] Hall M, Mark E, Frank G, *et al*. The WEKA data mining software: An update. ACM SIGKDD Explorations Newsletter, 2009, 11(1): 10−18. [doi: 10.1145/1656274.1656278]

附中文参考文献:

[7] 林冰雁, 徐伟华, 杨倩. 带偏好度量的直觉模糊序决策信息系统的部分一致约简. 计算机科学, 2018, 45(1): 148−151, 187.

[8] 邓大勇. 全粒度粗糙集属性约简. 模式识别与人工智能, 2018, 31(3): 230−235.

[9] 熊熙, 乔少杰, 韩楠, 元昌安, 张海清, 李斌勇. 一种基于模糊选项关系的关键属性提取方法. 计算机学报, 2019, 42(1): 190−202.

[13] 陈昊, 杨俊安, 庄镇泉. 变精度粗糙集的属性核和最小属性约简算法. 计算机学报, 2012, 35(5): 1011−1017. [doi: 10.3724/SP.J.1016.2012.01011]

[14] 于洪, 王国胤, 姚一豫. 决策粗糙集理论研究现状与展望. 计算机学报, 2015, 38(8): 1628−1639. [doi: 10.11897/SP.J.1016.2015.01628]

[16] 续欣莹, 张扩, 谢珺, 谢刚. 基于互信息下粒子群优化的属性约简算法. 电子学报, 2017, 45(11): 129−138. [doi: 10.3969/j.issn.0372-2112.2017.11.017]

[21] 张钧波, 李天瑞, 潘毅, 罗川, 滕飞. 云平台下基于粗糙集的并行增量知识更新算法. 软件学报, 2015, 26(5): 1064−1078. http://www.jos.org.cn/1000-9825/4590.htm [doi: 10.13328/j.cnki.jos.004590]

[24] 钱进, 苗夺谦, 张泽华. 云计算环境下知识约简算法. 计算机学报, 2011, 34(12): 2332−2343. [doi: 10.3724/SP.J.1016.2011.02332]

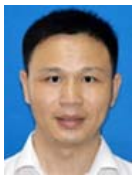
[27] 陆玉佳, 倪志伟, 朱旭辉, 许力分, 伍章俊. 基于 MapReduce 改进离散型萤火虫算法和多重分形的属性约简方法. 模式识别与人工智能, 2018, 31(6): 537−547.

[30] 吴信东, 嵇圣硙. MapReduce 与 Spark 用于大数据分析之比较. 软件学报, 2018, 29(6): 1770−1791. http://www.jos.org.cn/1000-9825/5557.htm [doi: 10.13328/j.cnki.jos.005557]

[31] 李佳, 梁吉业, 庞天杰. 一种基于优势粗糙集的多属性决策排序方法. 南京大学学报(自然科学), 2016, 52(5): 844−852.

[32] 张清华, 薛玉斌, 王国胤. 粗糙集的最优近似集. 软件学报, 2016, 27(2): 295−308. http://www.jos.org.cn/1000-9825/4854.htm [doi: 10.13328/j.cnki.jos.004854]

[33] 颜艳, 杨惠中. 一种基于互信息的粗糙集知识约简算法. 清华大学学报(自然科学版), 2007, 47(2): 1903−1906. [doi: 10.3321/j.issn:1000-0054.2007.z2.042]



危前进(1978－), 男, 博士, 副教授, 主要研究领域为粗糙集, 数据挖掘, 知识发现, 机器学习.

魏继鹏(1994－), 男, 硕士, 主要研究领域为粗糙集, 数据挖掘.

古天龙(1964－), 男, 博士, 教授, CCF 高级会员, 主要研究领域为形式化方法, 人工智能伦理, 数字政府.

常亮(1980－), 男, 博士, 教授, CCF 高级会员, 主要研究领域为知识表示与推理, 智能系统, 形式化方法.

文益民(1969－), 男, 博士, 教授, CCF 杰出会员, 主要研究领域为计算机视觉、推荐, 定位服务, 社会计算, 教育数据挖掘.