计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2024,33(5):195−202 [doi: 10.15888/j.cnki.csa.009496]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于蒙特卡洛树搜索的数值目标子群发现算法①

关承彬, 何振峰

(福州大学 计算机与大数据学院, 福州 350108)
通信作者: 关承彬, E-mail: 864673559@qq.com

摘　要: MonteCloPi 算法是一种基于蒙特卡洛树搜索 (Monte Carlo tree search, MCTS) 的任意时间子群发现算法, 旨在使用 MCTS 策略构建非对称的最佳优先搜索树来发现高质量的多样性模式集, 但是限制了目标为二值变量. 为此, 本文结合了数值目标的特点, 通过为置信度上界 (upper confidence bound, *UCB*) 公式选取合适的 *C* 值、动态调整各个样本的拓展权重并对搜索树进行剪枝、使用自适应 top-*k* 均值更新策略, 将 MonteCloPi 算法拓展到了数值目标. 最后, 在 UCI 数据集、全国健康与营养调查 (national health and nutrition examination survey, NHANES) 听力测试数据集上的实验结果表明本文的算法相比其他算法可以发现更高质量的多样性模式集, 并且最优子群的可解释性也更好.

关键词: 蒙特卡洛树搜索; 子群发现; 数值目标; 任意时间算法

引用格式: 关承彬,何振峰.基于蒙特卡洛树搜索的数值目标子群发现算法.计算机系统应用,2024,33(5):195–202. http://www.c-s-a.org.cn/1003-3254/9496.html

Subgroup Discovery Algorithm for Numerical Target Based on Monte Carlo Tree Search

GUAN Cheng-Bin, HE Zhen-Feng

(College of Computer and Data Science, Fuzhou University, Fuzhou 350108, China)

Abstract: MonteCloPi is an anytime subgroup discovery algorithm based on Monte Carlo tree search (MCTS). It aims to build an asymmetric best-first search tree to discover a diverse pattern set with high quality by MCTS policies, while it is limited to a binary target. To this end, this study combines the characteristics of the numerical target to extend the MonteCloPi algorithm to the numerical target. The study selects the appropriate *C* value for the upper confidence bound (*UCB*) formula, adjusts the expansion weight of each sample dynamically as well as prunes the search tree, and uses the adaptive top-*k*-mean-update policy. Finally, the experimental results on the UCI datasets and the National Health and Nutrition Examination Survey (NHANES) audiometry datasets show that the proposed algorithm outperforms other algorithms in terms of discovering diverse pattern sets with high quality and the interpretability of the best subgroup.

Key words: Monte Carlo tree search (MCTS); subgroup discovery; numerical target; anytime algorithm

子群发现是一种广泛适用的数据挖掘技术, 旨在发现集合的不同属性或变量 (描述变量) 之间关于用户感兴趣的特定属性 (目标变量) 的有趣关系[1], 提取的模式通常以规则的形式表示. 这项任务的一个重要特点是从数据集中提取的模式在目标变量上的统计分布显著不同于目标变量上的总体统计分布, 这样的模式可以认为是有趣的[2]. 例如, 模式: (年龄≥65∧吸烟=true)→患有肺癌=true 代表了年龄大于等于 65 岁且作为吸烟者的一个子群, 相对于其他人群成为肺癌患者的可能异常的高. 通过子群发现算法可以使用户更好地了解数据特征之间的关系, 并且已经应用于许多特定领域中[1], 例如, 在医疗和生物信息学领域中, 子群发

① 基金项目: 福建省自然科学基金 (2022J01574)
收稿时间: 2023-11-15; 修改时间: 2023-12-20; 采用时间: 2024-01-05; csa 在线出版时间: 2024-04-01
CNKI 网络首发时间: 2024-04-03

现可以帮助研究人员识别患有特定疾病的人群, 并进一步深入探索疾病的原因和治疗方法; 在人口统计领域, 子群发现可以帮助专家找到高收入或者低收入人群的内部和外部影响因素, 从而为接下来的研究以及决策提供方向.

数据通常分为许多类型, 包含二值变量、名义变量、数值变量, 其中数值变量由于取值是非离散的, 大大增加了搜索空间的大小, 从中提取有趣的子群也变得更加困难. 此外, 数据集分为两个部分: 描述变量以及目标变量. 最近的子群发现主要是针对数值型描述变量和二值或名义型目标变量进行研究[1,2]. 但是随着数值型目标变量数据集以及应用需求的增加, 数值型目标变量的子群发现受到越来越多的关注, 传统的方法只是将数值目标变量进行离散化, 而忽略了数值目标的原有信息. 为解决这个问题, Lemmerich 等人[3]提出 SD-MAP*算法, 基于 SD-MAP 算法[4]所使用的基于最小支持阈值的 FP 树增长的策略, 并增加了严格乐观估计[5]来对树进行剪枝, 但是该算法对数值描述变量进行离散化, 从而导致信息的损失, 只能发现次优的子群. Nguyen 等人[6]提出了 FLEXI 算法, 通过对每个属性的不同取值区间进行装箱, 可以适用于任何目标变量, 并且通过最大化所有箱的平均质量来找到较优的子群, 在数据量大的时候可以通过较少的采样来保证算法运行效率, 但是需要微调超参数以取得更好的结果. Millot 等人[7]提出了 OSMIND 算法, 不用事先进行离散化, 并且利用在正例上闭合 (close on the positive, *COTP*) 的间隔模式[8]、最小边界改变 (MinIntChange)[9]、严格乐观估计[5]等策略来修剪搜索空间, 保证可以搜索到最优子群, 但结果缺少多样性, 并且对大的模式搜索空间进行穷举搜索需要大量的时间甚至难以进行.

为了在不进行离散化的情况下高效挖掘大数据集中的多样性模式集, Bosc 等人[10]和 Mathonat 等人[11]提出了基于蒙特卡洛树搜索 (MCTS) 的子群发现方法 MCTS4DM 和 MonteCloPi 算法. MCTS 是一种基于模拟采样的最佳优先搜索算法, 每次迭代根据先前的采样结果来构建非对称的搜索树, 将搜索集中在最有希望的区域. 在子群发现中利用 MCTS 处理大的模式搜索空间时可以取得比其他传统算法更好的结果[11], 并且可以随时输出多样性模式集, 随着可用时间的增加最终会收敛到穷举搜索. 但是目前基于 MCTS 的子群发现算法限制了目标为二值属性, 因此研究如何将 MCTS 应用至数值目标子群发现任务中具有重要意义.

将 MCTS 应用于特定领域中需要根据问题背景对原始 MCTS 策略进行相应的调整. 例如文献[12,13]中汇总了 MCTS 在棋盘游戏和电子游戏等领域中的应用以及改进方法, 主要包括: 针对原始 MCTS 算法中分支因子过多、不同场景下 *UCB* 公式中 *C* 值的选取等问题的解决方法. 在其他领域中, 文献[14]使用 MCTS 来求解问题规模较大的混合整数线性规划模型, 根据问题的约束条件修剪没有希望的分支, 在更短的时间内可以取得更好的效果.

因此, 受到上述文献中将 MCTS 应用于特定领域所做工作的启发, 再结合数值目标子群发现任务的特点, 本文分别从 *UCB* 公式中 *C* 值的调整、动态调整待拓展结点权重和对搜索树应用剪枝、采用自适应 top-*k* 均值更新策略等方面来改进 MonteCloPi 算法的选择、拓展和更新策略, 将其拓展到了数值目标数据集上, 可以在不进行离散化的情况下随时获得高质量的多样性模式集. 本文第 1 节介绍 MonteCloPi 算法. 第 2 节介绍改进后适用于数值目标的 MonteCloPi 算法. 第 3 节对改进的 MonteCloPi 算法进行实验并分析实验结果. 第 4 节总结本文的工作和未来的研究方向.

1 MonteCloPi 算法

在 MonteCloPi 算法中, $D=(O, A, C)$ 表示数据集, O 表示一组样本, A 表示一组描述属性, C 表示一个目标属性, O^+ 表示 C 取值为+的样本, 即正例样本. 间隔模式 p 提供了在各个描述属性上的限制, p 的覆盖为满足 p 中每个限制的样本集合, 记作 $ext(p)$, 支持度 $supp(p)=|ext(p)|$, 即覆盖样本的个数. *int* 将样本集合映射到最严格的间隔模式上, 例如 $int(ext(p))$ 表示 p 的闭合间隔模式, 任何比它更小的间隔模式将会至少失去一个样本. $ext^+(p)$ 表示满足 p 中每个限制的正例样本集合, p 的在正例上闭合 (*COTP*) 间隔模式为 $COTP(p)=int(ext^+(p))$. 对于间隔模式 p 和 q, MEET 运算 $\prod$ 定义为两者各个间隔的并集, 例如 $\langle[1,2],[6,7]\rangle \prod \langle[1,1],[8,9]\rangle = \langle[1,2],[6,9]\rangle$, 并且两个 *COTP* 间隔模式的 MEET 也为 *COTP* 间隔模式[11].

MonteCloPi 算法中使用的子群质量度量 $\varphi(p)$ 为:

$$WRAcc(p,D,D_c)=\frac{supp(p,D)}{|D|}\times\left(\frac{supp(p,D_c)}{supp(p,D)}-\frac{|D_c|}{|D|}\right) \tag{1}$$

其中, D_c 为目标值为 c 的数据集, $|D|$和$|D_c|$分别为数据集 D 和 D_c 中样本个数.

MonteCloPi 算法的模式集挖掘目标为找到高质量的非 θ-冗余模式集 R, 即对于给定的相似度阈值 $\theta\in[0, 1]$, $\forall p, q\in R$ 且 $p\neq q$, 满足 $sim(p, q)\leqslant\theta$, 其中 sim 是相似性函数, 例如使用 *Jaccard* 系数, sim 越高表示 p 和 q 越相似, 反之亦然. sim 的定义如下:

$$sim(p,q)=\frac{|ext(p)\cap ext(q)|}{|ext(p)\cup ext(q)|} \tag{2}$$

MonteCloPi 算法先从数据集 D 中挖掘最佳模式集 S, 再对 S 进行后处理移除冗余模式, 获得非冗余结果模式集 R. 后处理流程[10]如下: 将模式集 S 中所有模式按质量递减排序, 先将最优的模式存入 R, 之后遍历模式集 S, 对遍历到的模式与结果模式集 R 中每个模式按式 (2) 计算相似程度, 若小于阈值 θ 则将其加入 R, 否则丢弃该模式, 重复上述过程直至 R 包含 k 个模式, 此时 R 是一个非 θ-冗余模式集并且其中包含 top-k 个非冗余子群. R 的形式化定义如下:

$$R=\mathrm{argmax}_{R\subseteq S}\frac{1}{k}\sum_{r\in R}\varphi(r),\ \forall p,q\in R且p\neq q, sim(p,q)\leqslant\theta \tag{3}$$

MonteCloPi 算法通过不断迭代构建搜索树来挖掘最佳模式集 S, 直至达到给定的时间预算或者探索了整个搜索空间, 每次迭代分为 4 个步骤: 选择、拓展、模拟、更新.

MonteCloPi 算法的步骤如算法 1 所示.

算法 1. MonteCloPi 算法

输入: 数据集 D, 时间预算 *timeBudget*, 质量度量 φ.
输出: 含 top-k 个非冗余子群的模式集 R.

1) 初始化: S=优先级队列, 根结点⊥为空
2) while 没有达到 *timeBudget* do
3) p_{sel}←从根结点⊥递归选择具有最大 *UCB* 值且还未完全拓展的子结点//选择
4) o^+←从 p_{sel} 的候选正例随机选择一个 //拓展
5) p^+←o^+转化为对应的间隔模式
6) p_{exp}←MEET(p_{sel}, p^+)
7) *S.add*(p_{exp}, φ_{exp}) //将模式及其质量加入队列中
8) p_{roll}←p_{exp} 的随机一个属性设为该属性的随机值, 其他属性设为$[-\infty, +\infty]$ //模拟
9) *S.add*(p_{roll}, φ_{roll}) //将模式及其质量加入队列中
10) for each $p\in p_{exp}$ 到⊥路径上的所有结点 //更新
11) $\overline{\varphi}(p)\leftarrow\frac{N(p)\times\overline{\varphi}(p)+\varphi_{roll}}{N(p)+1}$, $N(p)\leftarrow N(p)+1$
12) end for
13) end while
14) R←*S.topK*() //使用后处理获得非冗余模式集 R

在算法 1 中, 第 1 行的 S 用于保存当前最优模式集合, 根结点⊥不含任何样本; 第 3–12 行对应于一次迭代过程, 不断重复直至达到给定的时间预算; 其中第 3 行对应于选择步骤, *UCB* 公式由式 (4) 给出, 通过 *UCB* 可以选择出具有较优质量的结点或者很少被访问的结点, 未完全拓展的结点为没有覆盖所有正例的结点; 第 4–6 行对应于拓展步骤, 这一步将拓展结点加入搜索树中来构建搜索树, 候选正例为还未被当前结点的 *COTP* 间隔模式所覆盖的正样本, 正例对应的间隔模式为将正例的各个描述属性值作为对应间隔上下限的间隔模式, 例如 o=(2, 4) 对应的间隔模式为〈[2, 2], [4, 4]〉; 第 8 行对应于模拟步骤, 通过快速将拓展结点模拟至终点状态 (覆盖所有样本) 附近来估计该结点下分支的好坏; 第 10–12 行对应于更新步骤, 根据模拟的质量φ_{roll}反向更新 p_{exp} 至⊥路径上结点的平均质量和访问次数, 用于引导后续迭代中结点的选择; 第 14 行对应于后处理步骤, 从最优模式集合 S 中筛选出非冗余模式集 R, 其中包含 top-k 个非冗余子群.

$$UCB(i)=\overline{\varphi}_i+C\sqrt{\frac{\ln(N)}{N_i}} \tag{4}$$

其中, $\overline{\varphi}_i$为所有更新到第 i 个子结点的质量的平均值, C 为常数, 用于探索和利用之间的平衡, 通常取 $\sqrt{2}$, N 为当前结点访问次数, N_i 为第 i 个子结点访问次数.

2 MCTSIND 算法

将 MonteCloPi 算法应用于数值目标, 还存在一些问题: 1) 使用针对数值目标的质量度量时, 计算的质量可能会很大, 式 (4) 的第 1 项会过大而弱化第 2 项的效果, 从而导致更少去探索很少被探索的空间; 2) 拓展时以相同概率随机选取样本, 算法缺乏稳定性, 并且拓展了过多无意义分支; 3) 在局部最优值点较少且比较分散的情况下, 结点的平均质量可能会因为访问到大量低质量模式而接近 0, 从而忽略了该结点下有趣的模式. 为此, 本文提出了 MCTSIND (Monte Carlo tree search in numerical data) 算法, 根据数值目标子群发现任务的特点分别改进 MCTS 的选择、拓展和更新策略来解决以上问题.

2.1 MCTS 策略的改进

在本文中使用适用于数值目标的基于平均值的质量度量:

$$\varphi(p)=|ext(p)|^a\times\left(\mu_{ext(p)}-\mu_{ext(p_D)}\right),a\in[0,1] \quad (5)$$

其中, $\mu_{ext(p)}$是 p 中样本的目标属性值的平均值, $\mu_{ext(p_D)}$为整个数据集中目标属性值的平均值, 也可以设置为我们所感兴趣的阈值, a 用于控制间隔模式中所覆盖样本的个数.

对于选择策略, 为了平衡搜索树的探索和利用, 需要对 *UCB* 中的 C 值进行调整. 考虑到传统 MCTS 算法中通常限制 *UCB* 的第 1 项在[0, 1]之间[13], 因此本文中调整 C 值为归一化式 (5) 所需的分母部分:

$$C=|D|^a\times\left(T_{\max}-\mu_{ext(p_D)}\right) \quad (6)$$

其中, $T_{\max}$ 为目标变量最大值. 本文采用调整后的 C 值来计算 *UCB* 值.

对于拓展策略, 考虑拓展大于等于$\mu_{ext(p_D)}$的样本, 按照每个样本 o_i 的目标值大小设置其初始权重$w(o_i)=\varphi(o_i)=T_{o_i}-\mu_{ext(p_D)}$, T_{o_i}为样本 o_i 的目标值. 每个样本在拓展时被选中的概率为$\dfrac{w(o_i)}{\sum_{o_i\in O}w(o_i)}$, 具有较高目标值的样本有更高概率被选中拓展. 此外, 拓展过程中可能会出现以下情况, 如图 1 所示.

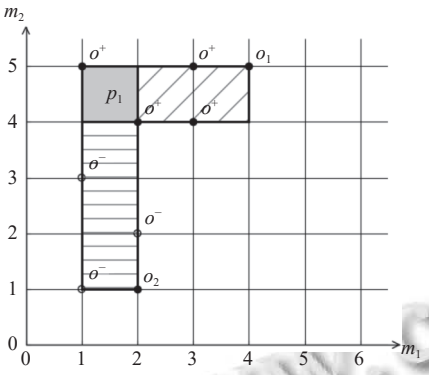

图 1 可能的拓展情况

图 1 中 m_1 和 m_2 为样本的两个属性, 实心点 o^+为目标值大于等于$\mu_{ext(p_D)}$的样本, 空心点 o^-为目标值小于$\mu_{ext(p_D)}$的样本. 例如当前结点的间隔模式为 p_1 (浅灰区域), 有两个候选拓展样本 o_1 和 o_2, 此时应该优先拓展 o_1 而不是 o_2. 因为 p_1 与 o_1 进行 MEET 运算后的间隔模式为浅灰加上斜线区域, 比拓展前多覆盖了 3 个高目标值样本; 而 p_1 与 o_2 进行 MEET 运算后的间隔模式为浅灰加上横线区域, 比拓展前只多覆盖了 1 个高目标值样本, 但是额外覆盖了 3 个低目标值样本.

因此, 本文定义了质量增益比$\dfrac{\varphi_{\exp}}{\varphi_{\text{sel}}}$来适应以上可能的拓展情况, 根据质量增益比调整后样本 o_i 的权重为:

$$w(o_i)=\frac{\varphi_{\exp}}{\varphi_{\text{sel}}}w(o_i) \quad (7)$$

其中, $\varphi_{\exp}$ 表示拓展后结点的质量, φ_{sel} 表示拓展前结点的质量. 对于图 1 的情况, 可以使拓展后能获得更高质量模式的样本具有更高的权重, 从而有更高概率在后续被拓展, 因为这些样本周围可能有较密集的高目标值样本.

拓展或模拟结点后需要将该模式及其质量加入当前最优模式集合 S 中, S 有一个模式个数上限, 此时可以根据 S 中的最低质量 $\varphi_{\min}$ 对没有希望的分支进行剪枝. 步骤如下: 首先设置模式的支持度上限 $maxsupp=0.2|D|$, 即总样本数的 20%, 再根据 $\varphi_{\min}$ 来计算阈值 μ_{thresh}, 阈值定义如下:

$$\mu_{\text{thresh}}=\frac{\varphi_{\min}}{maxsupp^a}+\mu_{ext(p_D)} \quad (8)$$

只有拓展目标值大于该阈值的样本才有可能提高 S 中的模式质量.

记某个结点 p 已覆盖的样本集合为 S_1, 可拓展样本集合$S_2=\{o|(o\in D-S_1)\wedge(T_o\geqslant\mu_{\text{thresh}})\}$, T_o 为样本 o 的目标值, 如果满足以下条件:

$$\left(\frac{1}{|S_1\cup S_2|}\sum_{o\in S_1\cup S_2}T_o\right)<\mu_{\text{thresh}} \quad (9)$$

则将结点 p 下的所有分支修剪, 此时 p 完全拓展, 因为将其子结点拓展到搜索树中并进行探索对提升当前最优模式集合 S 的质量没有帮助.

对于更新策略, 文献[10]中提出了 top-k 均值更新策略来解决局部最优值点较少且比较分散的问题: 使用更新到该结点的前 k 个最优质量的平均值来更新该结点质量, 使搜索树“记住”从该结点出发可以发现高质量的模式, 从而可以更多地“利用”该结点. 但是原策略需要指定额外的超参数 k, 因此本文提出自适应 top-k 均值更新策略: 结点 p 的 top-k 队列 Q_p 的初始上限 k 为 2, 之后更新到 p 的质量若大于 Q_p 的均值才加入 Q_p, 此时 k 也随之增加. 在本文中使用自适应 top-k 均值更新策略来对 MonteCloPi 算法进行改进.

2.2 算法描述

基于第 2.1 节中对 MonteCloPi 算法中 *UCB* 公式 C 值的选取、样本拓展权重的调整、对搜索树进行剪

枝和使用自适应 top-k 均值更新策略的改进, 本文提出的 MCTSIND 算法如算法 2 所示.

算法 2. MCTSIND 算法

输入: 数据集 D, 时间预算 *timeBudget*, 质量度量 φ.
输出: 含 top-k 个非冗余子群的模式集 R.

1) 初始化: S=优先级队列, 根结点⊥为空
2) while 没有达到 *timeBudget* do
3) p_{sel}←从根结点⊥递归选择具有最大 *UCB* 值且还未完全拓展的子结点 (使用式 (6) 中新的 C 值) //选择
4) o^+←按权重概率 $\frac{w(o_i)}{\sum_{o_i \in O} w(o_i)}$ 从 p_{sel} 的待拓展样本中随机选取一个 //拓展
5) p^+←o^+转化为对应的间隔模式
6) p_{exp}←MEET(p_{sel}, p^+)
7) 按式 (7) 更新 o^+的权重
8) $S.add(p_{exp}, \varphi_{exp})$ //将模式及其质量加入队列中
9) 若 S 已满, 按式 (8) 和式 (9) 对搜索树中的结点进行剪枝
10) p_{roll}←p_{exp} 的随机一个属性设为该属性的随机值, 其他属性设为[−∞, +∞]//模拟
11) $S.add(p_{roll}, \varphi_{roll})$ //将模式及其质量加入队列中
12) 若 S 已满, 按式 (8) 和式 (9) 对搜索树中的结点进行剪枝
13) for each $p \in p_{exp}$ 到⊥路径上的所有结点 //更新
14) if p 的 top-k 质量队列 Q_p 未满 then
15) 将 φ_{roll} 加入 Q_p
16) else 当 $\varphi_{roll} > Q_p$ 中质量的均值时才将 φ_{roll} 加入 Q_p
17) end if
18) $\bar{\varphi}(p)$←Q_p 中质量的均值, $N(p)$←$N(p)$+1
19) end for
20) end while
21) R←$S.topK()$ //使用后处理获得非冗余模式集 R

算法 2 与算法 1 不同之处在于: 第 3 行选择步骤使用式 (6) 中新的 C 值; 第 4 行按权重概率从待拓展样本中进行选取, 待拓展样本为目标值大于等于$\mu_{ext(p_D)}$的样本; 第 7 行在拓展后按式 (7) 更新被选中样本 o^+的权重; 第 9 和 12 行在 S 更新后且已满时按照式 (8) 和式 (9) 计算阈值对搜索树中的结点进行剪枝, 被剪枝结点接下来不会被选择并拓展; 第 14–18 行的 Q_p 选择性地保存较高的模拟质量并使用 Q_p 中质量的均值来更新而不是使用所有经过 p 的模拟质量.

3 实验分析

3.1 实验数据集

本文使用的数值目标数据集是 UCI 提供的 auto MPG (AMPG)、concrete compressive strength (CCS)、airfoil self-noise (ASN)、wine quality-red (WQR)、wine quality-white (WQW)、air quality (AQ) 这 6 个数据集. 此外还使用 national health and nutrition examination survey (NHANES) 提供的听力测试数据集 Audiometry 2011–2012, 2015–2016 (AUD11–16) 和 Audiometry 2017–2018 (AUD17–18), 如表 1 所示.

表 1 数据集信息

数据集	样本数	属性数 (名义/数值)	$\mu_{ext(p_D)}$ (目标变量均值或感兴趣阈值)
AMPG	398	0/7	23.515
CCS	1 030	0/8	35.818
ASN	1 503	0/5	124.836
WQR	1 599	0/11	5.636
WQW	4 898	0/11	5.878
AQ	9 357	0/8	2.153
AUD11–16	8179	6/9	低频25, 高频35
AUD17–18	2 667	6/10	低频25, 高频35

NHANES 是一项基于人群的横断面调查, 旨在收集美国成人和儿童的健康和营养状况的信息. 其听力测试的对象是 20–69 岁的人群, 数据集的描述变量对应多种可能导致听力损失的因素, 并含有多个数值类型的目标变量, 分别对应左右耳在不同频率 (0.5、1、2、3、4、6、8 kHz) 下的听力阈值 (最小听清的分贝数), 阈值越大说明听力越差, 本文中将 0.5–2 kHz 归为低频, 3–8 kHz 归为高频, 低频中听力阈值>25 dB 和高频中听力阈值>35 dB 定义为听力损失, 以此作为感兴趣的阈值, 将表 1 和式 (5) 中的$\mu_{ext(p_D)}$分别设为 25 和 35. 原始数据集中有多个描述变量含义类似, 本文将类似的描述变量合并为一个变量, 例如将耳朵有过多耵聍和有严重影响的耵聍合并为耳朵耵聍程度: 0-正常、1-过多、2-严重. 此外排除了缺失属性值过多的受试者数据和属性 (超过 60%), 缺失不多的属性进行均值或众数填充, 经过预处理之后保留的属性有: AUQ020 (过去 24 h 是否感冒、鼻窦炎或者耳痛); AUQ040 (距上次听到噪音过去多少小时); AUXOTSP(是否进行常规耳镜检查); AUXROC (耳朵耵聍程度); AUXTMEP (中耳压力, 单位: daPa); AUXTPV (外耳道容积, 单位: cc); AUXTWID (鼓室图宽度, 单位: daPa); AUXTCOM (声顺值, 单位: cc). AUD17–18 还有一个额外属性: AUQ610 (距上次使用耳机过去多少小时), 除了 AUQ040 和 AUQ610, 其他属性都包含左右耳对应的数据.

在实验中, 使用式 (5) 作为质量度量 φ, 其中 a=0.5, 并且设置时间预算为 60 s, 模式集相似度阈值 θ=0.5, 最小支持度 *minsupp* 为样本总数的 10%, FLEXI 算法的初始分箱数设为 20.

3.2 UCI 数据集实验结果分析

首先, 使用表 1 的 UCI 数据集部分, 将 MCTSIND 与 MonteCloPi (直接使用式 (5) 的质量度量 φ)、SD-MAP*、FLEXI 和 OSMIND 这些子群发现算法挖掘的非冗余模式集基于以下两个方面进行比较.

(1) 基于给定的时间预算, 比较各个算法在数据集上使用后处理步骤所提取出的 top-5 非冗余子群关于质量度量 φ 的平均性能的相对值, 如图 2 所示.

从图 2 中可以看出, 在 UCI 数据集中, MCTSIND 发现的 top-5 非冗余子群的平均质量在大多数情况下都要优于其他 4 个算法, 仅在 WQR 数据集中比 FLEXI 要略低一些. 这得益于 MCTS 的探索与利用的折中, 可以尽可能多地发现不同的局部最优值点以提高最终结果的平均质量, 如果能给出更多的时间预算, MCTSIND 在 WQR 数据集上的表现可能会超过 FLEXI. 同时还可以看出将 MonteCloPi 直接应用于数值目标数据集难以发现高质量的多样性模式集.

(2) 使用不同的时间预算来研究各个算法随着时间的增加对算法性能的影响, 时间预算分别设置为 1 s、5 s、10 s、20 s、40 s、60 s、80 s、100 s、120 s, 图 3 描述了各个算法随着时间预算的变化所给出 top-5 非冗余子群的平均质量.

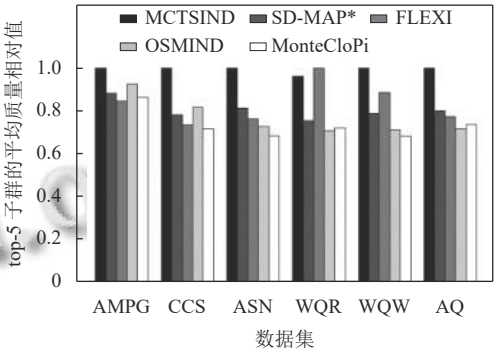

图 2 top-5 非冗余子群的平均质量相对值对比

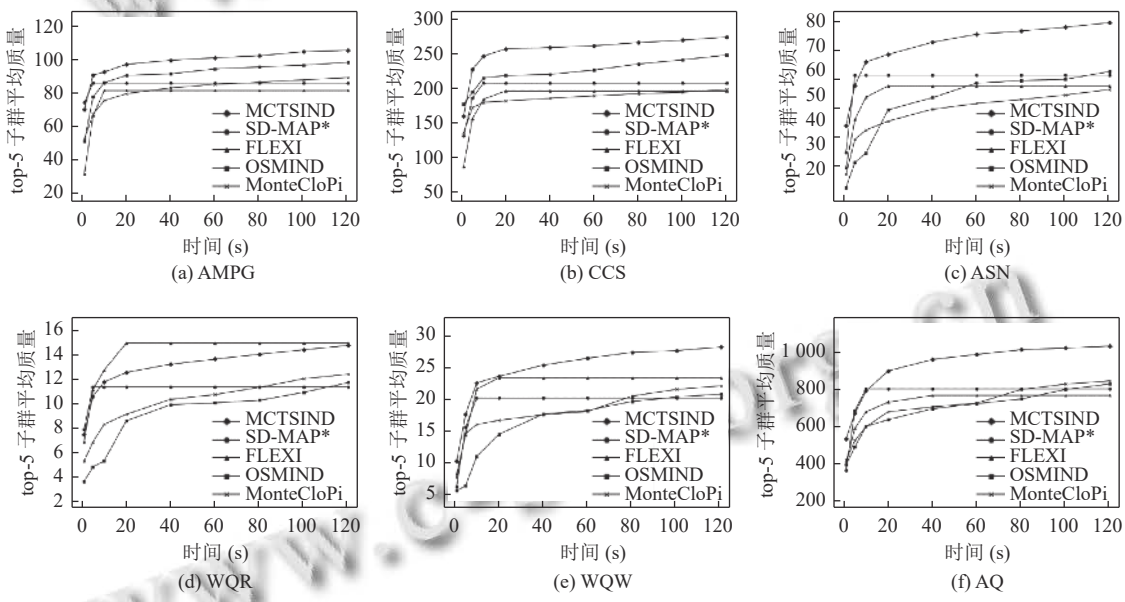

图 3 top-5 非冗余子群平均质量随时间变化对比

从图 3 中可以看出, SD-MAP*和 FLEXI 很快就因为探索完其修剪后的搜索空间而收敛, 由于使用了离散化导致较差的结果. 此外, MCTSIND 的总体表现比 MonteCloPi 更好, 虽然刚开始模式集质量增长较慢, 但是随着时间的增加不断给出更好的结果, 并且在不同样本个数的数据集上面表现稳定. OSMIND 的可伸缩性较差, 在 WQR、WQW、AQ 等大样本数据集上需要更多时间才能达到与其他算法相同的水平, 在 AMPG、CCS 这些样本个数略少的数据集才会获得较好的表现.

3.3 NHANES 数据集应用研究

最后, 使用表 1 中的 NHANES 听力测试数据集来研究 20–69 岁人群中低频和高频听力损失的可能因素. 分别将左右耳在各个频率下的听力阈值作为目标变量, 运行 MCTSIND 和 OSMIND 算法, 后者在文献[7]的实验中可以获得比 SD-MAP*更有解释性的子群. 下文将时间预算延长至 300 s, 并且使用所得到的 top-1 子群质量的平均值和各个间隔模式区间的并集来进行对比, 如表 2 所示.

表 2 MCTSIND 与 OSMIND 提取的间隔模式对比

数据集	低频/高频	算法	AUQ020	AUQ040	AUQ610	AUXOTSP	AUXROC	AUXTMEP	AUXTPV	AUXTWID	AUXTCOM	φ
AUD11–16	低频	OSMIND	[1, 1]	[2, +∞]	×	—	[1, 2]	[−229, 6]	[0.683, 1.795]	[83, 216]	[0.229, 1.434]	132.942
		MCTSIND	[1, 1]	—	×	—	[1, 2]	[−252, −84]	[0.609, 1.556]	[70, 252]	[0.172, 1.386]	**157.184**
	高频	OSMIND	[1, 1]	[5, 17]	×	—	[1, 2]	[−180, 54]	[1.475, 2.228]	[31, 136]	[0.533, 2.733]	209.937
		MCTSIND	—	[1, 22]	×	—	—	[−144, 24]	[1.547, 2.373]	[42, 120]	[0.792, 2.932]	**235.148**
AUD17–18	低频	OSMIND	[1, 1]	[1, +∞]	[3, +∞]	—	[1, 2]	[−298, −46]	[0.716, 1.771]	[77, 235]	[0.246, 1.427]	134.386
		MCTSIND	[1, 1]	—	—	—	[1, 2]	[−283, −162]	[0.628, 1.522]	[91, 277]	[0.162, 1.214]	**149.306**
	高频	OSMIND	[1, 1]	[3, 20]	[1, 21]	—	[1, 2]	[−169, 32]	[1.219, 2.382]	[38, 121]	[0.832, 2.302]	267.851
		MCTSIND	—	[1, 23]	[1, 24]	—	—	[−140, 16]	[1.401, 2.854]	[45, 107]	[0.954, 2.367]	**286.692**

表 2 的每列对应数据集的一个描述属性, 值表示间隔模式中对该描述属性的限制, ×代表不存在该属性, —代表该属性没有提供限制. 可以看出在 4 个听力测试数据集中, MCTSIND 比 OSMIND 发现的 top-1 子群质量都更优. 此外, OSMIND 在描述高频和低频听力损失的可能因素时使用了相同的描述属性, 不能很好地区分出两者之间的差异. 而 MCTSIND 对于高频和低频听力损失所用的描述属性不同, 可以很容易地看出两者的不同: 例如 AUQ020 (过去 24 h 是否感冒、鼻窦炎或者耳痛) 和 AUXROC (耳朵耵聍程度) 不是高频听力损失的影响因素, AUQ040 (距上次听到噪音过去多少小时) 和 AUQ610 (距上次使用耳机过去多少小时) 不是低频听力损失的影响因素. 并且 MCTSIND 使用了更少的描述属性, 相比于 OSMIND 提取出的结果可以被我们更好地理解和利用.

接下来分析 MCTSIND 所提取出的间隔模式含义. 对于非声阻抗测定因素 (表 2 的前 5 列描述属性), MCTSIND 认为高频听力损失与在过去的 24 h 内暴露于噪声之中或者使用耳机听音乐有关; 低频听力损失与在过去的 24 h 内有感冒、鼻窦炎或者耳痛症状、耳道中有过多的耵聍有关. 文献[15]中验证了过多暴露于噪声之中会影响高频听力, 感冒、耳痛会影响低频听力, 这可以说明 MCTSIND 所提取模式的有效性.

对于声阻抗测定因素 (表 2 的后 4 列描述属性), 结合文献[16]给出的成年人的正常值范围: AUXTMEP (中耳压力) [−150, 50] daPa, AUXTPV (外耳道容积) [0.6, 1.5] cc, AUXTWID (鼓室图宽度) [50, 100] daPa, AUXTCOM (声顺值) [0.3, 1.4] cc, 可以分析出 MCTSIND 认为高频听力损失与外耳道容积过大和声顺值过高有关; 低频听力损失与中耳压力过低、鼓室图宽度过大和较低的声顺值有关. 对于声阻抗测定因素对高低频听力损失的影响方面, 目前还找不到相关的文献, 基于 MCTSIND 提取出的结果可以辅助相关人员对听力损失原因进行分析.

4 结论与展望

针对 MonteCloPi 算法只适用于二值目标变量的问题, 本文提出了 MCTSIND 算法. 该算法结合了数值目标子群发现任务的特点, 改进了 MonteCloPi 算法的 MCTS 策略: 为 *UCB* 公式选取合适的 *C* 值、在拓展阶段动态调整各个样本的权重并且定义目标值阈值对搜索树进行剪枝、使用自适应 top-k 均值更新策略, 将 MonteCloPi 算法拓展到了数值目标. 最后, 在 UCI 和 NHANES 数据集上的实验结果表明 MCTSIND 算法可以有效处理数值目标数据集, 其发现的 top-5 非冗余子群的平均质量比其他算法更优; 对高低频听力损失因素进行研究时可以发现比 OSMIND 算法更高质量和更具有可解释性的子群. 未来的工作中将研究如何将 MCTSIND 算法拓展到多个目标变量的情况.

参考文献

1 Herrera F, Carmona CJ, González P, *et al*. An overview on subgroup discovery: Foundations and applications. Knowledge and Information Systems, 2011, 29(3): 495–525. [doi: 10.1007/s10115-010-0356-2]

2 Meeng M, Knobbe A. For real: A thorough look at numeric attributes in subgroup discovery. Data Mining and Knowledge Discovery, 2021, 35(1): 158–212. [doi: 10.1007/s10618-020-00703-x]

3 Lemmerich F, Atzmueller M, Puppe F. Fast exhaustive subgroup discovery with numerical target concepts. Data Mining and Knowledge Discovery, 2016, 30(3): 711–762. [doi: 10.1007/s10618-015-0436-8]

4 Atzmueller M, Puppe F. SD-Map —A fast algorithm for exhaustive subgroup discovery. Proceedings of the 10th European Conference on Principles of Data Mining and

Knowledge Discovery. Berlin: Springer, 2006. 6–17.
5 Grosskreutz H, Rüping S, Wrobel S. Tight optimistic estimates for fast subgroup discovery. Proceedings of the 2008 Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Antwerp: Springer, 2008. 440–456.
6 Nguyen HV, Vreeken J. Flexibly mining better subgroups. Proceedings of the 2016 SIAM International Conference on Data Mining (SDM). Miami: The Society for Industrial and Applied Mathematics, 2016. 585–593. [doi: 10.1137/1.9781611974348.66]
7 Millot A, Cazabet R, Boulicaut JF. Optimal subgroup discovery in purely numerical data. Proceedings of the 24th Pacific-Asia Conference on Knowledge Discovery and Data Mining. Singapore: Springer, 2020. 112–124.
8 Belfodil A, Belfodil A, Kaytoue M. Anytime subgroup discovery in numerical domains with guarantees. Proceedings of the 2019 Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Dublin: Springer, 2019. 500–516. [doi: 10.1007/978-3-030-10928-8_30]
9 Kaytoue M, Kuznetsov SO, Napoli A. Revisiting numerical pattern mining with formal concept analysis. Proceedings of the 22nd International Joint Conference on Artificial Intelligence. Barcelona: AAAI Press, 2011. 1342–1347. [doi: 10.5591/978-1-57735-516-8/IJCAI11-227]
10 Bosc G, Boulicaut JF, Raïssi C, *et al*. Anytime discovery of a diverse set of patterns with Monte Carlo tree search. Data Mining and Knowledge Discovery, 2018, 32(3): 604–650. [doi: 10.1007/s10618-017-0547-5]
11 Mathonat R, Nurbakova D, Boulicaut JF, *et al*. Anytime subgroup discovery in high dimensional numerical data. Proceedings of the 8th IEEE International Conference on Data Science and Advanced Analytics (DSAA). Porto: IEEE, 2021. 1–10. [doi: 10.1109/DSAA53316.2021.9564223]
12 Browne CB, Powley E, Whitehouse D, *et al*. A survey of Monte Carlo tree search methods. IEEE Transactions on Computational Intelligence and AI in Games, 2012, 4(1): 1–43. [doi: 10.1109/TCIAIG.2012.2186810]
13 Świechowski M, Godlewski K, Sawicki B, *et al*. Monte Carlo tree search: A review of recent modifications and applications. Artificial Intelligence Review, 2023, 56(3): 2497–2562. [doi: 10.1007/s10462-022-10228-y]
14 邱云飞, 于智龙, 郭羽含, 等. 蒙特卡洛树搜索下的整合多目标可持续闭环供应链网络优化. 计算机集成制造系统, 2022, 28(1): 269–293. [doi: 10.13196/j.cims.2022.01.025]
15 Shargorodsky J, Curhan GS, Curhan CG, *et al*. Change in prevalence of hearing loss in US adolescents. JAMA, 2010, 304(7): 772–778. [doi: 10.1001/jama.2010.1124]
16 Roup CM, Wiley TL, Safady SH, *et al*. Tympanometric screening norms for adults. American Journal of Audiology, 1998, 7(2): 55–60. [doi: 10.1044/1059-0889(1998/014)]

(校对责编: 孙君艳)