计 算 机 研 究 与 发 展
Journal of Computer Research and Development
DOI: 10.7544/issn1000-1239.202220769
60(7): 1533−1550, 2023

# 基于多目标混合蚁狮优化的算法选择方法

李庚松[1] 刘 艺[2] 郑奇斌[2] 李 翔[2] 刘 坤[2] 秦 伟[2] 王 强[2] 杨长虹[3]
[1](国防科技创新研究院 北京 100071)
[2](军事科学院 北京 100091)
[3](天津(滨海)人工智能创新中心 天津 300457)
(gus_li@163.com)

# Algorithm Selection Method Based on Multi-Objective Hybrid Ant Lion Optimizer

Li Gengsong[1], Liu Yi[2], Zheng Qibin[2], Li Xiang[2], Liu Kun[2], Qin Wei[2], Wang Qiang[2], and Yang Changhong[3]
[1](*National Innovation Institute of Defense Technology*, *Beijing* 100071)
[2](*Academy of Military Sciences*, *Beijing* 100091)
[3](*Tianjin Artificial Intelligence Innovation Center*, *Tianjin* 300457)

**Abstract** Algorithm selection refers to selecting an algorithm that satisfies the requirements for a given problem from feasible algorithms, and algorithm selection based on meta-learning is a widely used method, in which the key components are meta-features and meta-learners. However, existing research is difficult to make full use of the complementarity of meta-features and the diversity of meta-learners, which are not conducive to further improving the method performance. To solve the above problems, a selective ensemble algorithm selection method based on multi-objective hybrid ant lion optimizer (SAMO) is proposed. It designs an algorithm selection model, which sets the accuracy and diversity of the ensemble meta-learners as the optimization objectives, introduces meta-feature selection and selective ensemble, and chooses meta-features and heterogeneous meta-learners simultaneously to construct ensemble meta-learners; it proposes a multi-objective hybrid ant lion optimizer to optimize the model, which uses discrete code to select meta-feature subsets and constructs ensemble meta-learners by continuous code, and applies the enhanced walk strategy and the preference elite selection mechanism to improve the optimization performance. We utilize 260 datasets, 150 meta-features, and 9 candidate algorithms to construct classification algorithm selection problems and conduct test experiments, and the parameter sensitivity of the method is analyzed, the multi-objective hybrid ant lion optimizer is compared with four evolutionary algorithms, 8 comparative methods are compared with the proposed method, and the results verify the effectiveness and superiority of the method.

**Key words** algorithm selection; multi-objective ant lion optimizer; meta-feature selection; selective ensemble; meta-learning; classification

**摘 要** 算法选择是指从可行算法中为给定问题选择满足需求的算法,基于元学习的算法选择是应用较为广泛的方法,元特征和元算法是其中的关键内容,而现有研究难以充分利用元特征的互补性和元算法的多样性,不利于进一步提升方法性能.为了解决上述问题,提出基于多目标混合蚁狮优化的算法选择方法(SAMO),设计算法选择模型,以集成元算法的准确性和多样性作为优化目标,引入元特征选择和选

**收稿日期**: 2022-08-28; **修回日期**: 2023-02-01
**基金项目**: 科技部科技创新2030—重大项目(2020AAA0104802); 国家自然科学基金项目(91948303); 国家自然科学基金青年科学基金项目(61802426)
This work was supported by the Science and Technology Innovation 2030 Major Project of China (2020AAA0104802), the National Natural Science Foundation of China (91948303), and the National Natural Science Foundation of China for Young Scientists (61802426).
**通信作者**: 刘艺(albertliu20th@163.com)

择性集成，同时选择元特征和异构元算法以构建集成元算法；提出多目标混合蚁狮算法对模型进行优化，使用离散型编码选择元特征子集，通过连续型编码构建集成元算法，应用增强游走策略和偏好精英选择机制提升寻优性能.使用260个数据集、150种元特征和9种候选算法构建分类算法选择问题来进行测试，分析方法的参数敏感性，将多目标混合蚁狮算法与4种演化算法进行比较，通过对8种对比方法与所提方法进行对比实验，结果验证了所提方法的有效性和优越性.

**关键词** 算法选择；多目标蚁狮优化；元特征选择；选择性集成；元学习；分类

**中图法分类号** TP181

在大数据时代背景下，利用数据进行分析和决策成为了不同领域的重要工作，各种人工智能算法为此提供了不可忽视的助力.然而，“没有免费午餐”定理表明，不存在一个在所有问题上具备优越性能的“最优”算法[1].如何从大量可行方法中为给定任务选择满足需求的合适算法成为了工程中的关键问题，即算法选择问题[2].算法选择问题可以通过人工选择方法和机器学习方法解决.人工选择方法包括实验试错法和专家知识法.前者通过大量的实验获得算法性能，分析结果并选择算法；后者则按照领域专家的指导进行算法选择.然而，实验试错法成本较高，专家知识法基于专家的经验知识，存在人为偏差且获取较为困难.机器学习方法提取问题的抽象特征，设计模型实现自动化的算法选择，包括基于元学习的方法和基于协同过滤的方法[3-4].其中，基于元学习的方法研究基础较为深厚，具有灵活性高、适用范围广、计算成本低等优点，成为了算法选择的主要方法[5-7].

基于元学习的算法选择利用问题的元特征(抽象特征)和历史学习经验训练元算法，构建问题元特征到合适算法的映射.元特征和元算法是其中的关键内容，而现有研究在元特征和元算法的选择和使用上存在一些缺点：在元特征方面，多数研究选择固定的元特征[8-10]，可扩展性较弱，且难以充分利用元特征的互补性；在元算法方面，研究或采用单个元算法，泛化性能不足，或采用同构集成元算法[11-14]，没有利用不同元算法的优势，导致多样性不足.

集成学习的相关研究表明，利用准确且多样的基学习器进行集成可以获得更精确、更稳定的学习性能[15-16].选择性集成方法根据基学习器的准确性和多样性从多个基学习器中选取部分进行集成，可以减少集成算法的存储空间和预测时间并提升其泛化性能，是集成学习的热点方向之一[17-19].演化算法有着鲁棒性强、适用性高、可以全局搜索等特性，在选择性集成研究中得到了广泛应用[20-22].

蚁狮优化(ant lion optimizer, ALO)[23]算法是近年来出现的演化算法，它模拟蚁狮捕食蚂蚁的行为机制对最优解进行搜索，具有寻优性能强、参数设置少、易于实现等优点.研究人员将ALO扩展应用于多目标优化问题，在污水处理、工程设计等领域取得了良好的应用效果[24-26].

为了提升基于元学习的算法选择性能，提出基于多目标混合蚁狮优化的选择性集成算法选择方法(selective ensemble algorithm selection method based on multi-objective hybrid ant lion optimizer, SAMO)，该方法包括一种算法选择模型和一种多目标混合蚁狮优化算法.算法选择模型以集成元算法的准确性和多样性作为优化目标，同时选择元特征和构建选择性集成元算法.多目标混合蚁狮优化算法对模型进行优化，通过离散型编码选择元特征子集，使用连续型编码构建集成元算法，在此基础上应用增强游走策略和偏好精英选择机制提升寻优性能.

本文的主要贡献包括3个方面：

1)提出一种算法选择模型，同时从元特征和元算法2个方面提升基于元学习的算法选择性能.模型通过元特征选择寻找互补性较强的元特征子集，通过选择性集成构建异构集成元算法.

2)提出一种多目标混合蚁狮优化算法，求解算法选择模型.通过混合编码机制使个体同时进行元特征选择和选择性集成；采用增强游走策略，在个体搜索过程中引入随机性，增加算法的种群多样性；应用偏好精英选择机制，以不同的优化目标为偏好选择精英个体，增强算法的寻优能力.

3)使用260个数据集、150种元特征和9种候选算法构建分类算法选择问题，在此基础上将所提方法与8种典型方法在4种性能指标上进行对比分析，结果证明了所提方法的有效性和优越性.

## 1 相关概念和工作

### 1.1 基于元学习的算法选择

#### 1.1.1 基于元学习的算法选择框架

基于元学习的算法选择框架[27-28]如图1所示.框

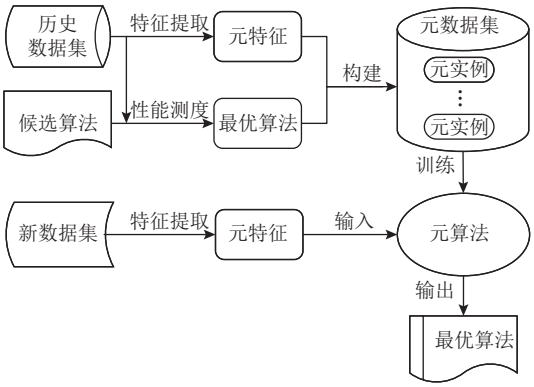


Fig. 1 Framework of algorithm selection based on meta-learning

图 1 基于元学习的算法选择框架

架首先提取历史数据集的元特征，并通过性能测度评估确定历史数据集的最优算法，在过程中采用不同的性能测度方法可能获得不同的最优算法；然后以元特征为属性，以最优算法为标签组成元实例构建元数据集，并在元数据集上训练元算法；最后，提取新数据集的元特征输入已训练的元算法，元算法即对其最优算法进行预测输出.

1.1.2 元特征相关内容

根据反映数据集特性的不同，元特征分为 4 类：

1）基于统计和信息论的元特征采用统计学和信息论的方法提取数据集的抽象特征[27]. 该类型元特征应用广泛且易于提取，然而其难以较好地刻画数据集特性.

2）基于决策树的元特征在数据集上训练决策树，使用决策树的结构信息作为元特征[28]. 其能够发掘数据集的整体特性，但提取成本高昂.

3）基于基准的元特征将基准算法在数据集上的性能指标值作为元特征[29]，其提取方法较为简单，然而对基准算法的依赖度较高.

4）基于问题复杂度的元特征从类重叠、类不平衡、数据稀疏度等方面对数据集的几何复杂度进行量化评估[30]. 该类型元特征反映了求解问题的困难程度，应用效果较好，但是计算复杂度较大.

1.1.3 元算法相关内容

按照训练过程的技术特点，将元算法分为 4 类：

1）基于规则的元算法生成候选算法的选择规则，通过将元特征与规则进行匹配，为新数据集选择合适算法，该方法具备较强的可解释性，但泛化性能较差. 针对负荷预测问题，文献 [8] 提取 18 种数据集元特征，训练分类回归树（classification and regression tree, CART）选择最优算法.

2）基于距离的元算法通过元特征测度距离，评估数据集之间的相似度，使用相似数据集的最优算法作为新数据集的合适算法. $k$ 最近邻（$k$ nearest neighbors, $k$NN）是研究中常见的元算法，其实现简单且运行较快，但关键参数 $k$ 的最优值设置较为困难. 文献 [9] 使用 32 种分类数据集元特征，基于 $k$NN 构建元特征到最优算法的映射.

3）基于回归的元算法预测各候选算法的性能指标值并进行比较，从而完成算法选择决策，使用较为广泛的元算法为支持向量回归（support vector regression, SVR）. 该类方法可以灵活地增减候选算法，但训练成本较高. 文献 [10] 选用 12 种元特征，应用 SVR 学习元特征与候选分类算法性能的映射关系.

4）基于集成的元算法通过一定策略组合多个性能较弱的元算法，得到具有较强性能的集成元算法，常见的元算法包括随机森林（random forest, RF）、极端梯度提升（extreme gradient boosting, XGB）等，这些方法的泛化性能较好，但训练速度较慢且参数设置较为复杂. 文献 [11] 提取 22 种元特征，采用 RF 学习元特征到最优分类算法的映射. 文献 [12] 使用 22 种元特征，采用随机森林回归（random forest regression, RFR）预测候选分类算法的性能. 文献 [13] 提取 98 种图像元特征，分别训练 RF 和 XGB 并预测最优图像分割算法. 文献 [14] 选择 39 种分类数据集元特征，应用轻量梯度提升机（light gradient boosting machine, LGBM）进行算法选择.

在上述研究中，元特征和元算法的选用仍存在一些问题：在元特征方面，研究通常根据需求选取固定的元特征，耦合度较高，可扩展性较弱；不同元特征从不同方面描述和提取数据集的抽象特征，具备一定的互补性，然而现有的方法难以有效利用. 在元算法方面，现有研究或使用泛化性能较弱的单个元算法，或采用同构基学习器的集成元算法，未能充分利用不同元算法的优势和多样性.

### 1.2 多目标优化问题

多目标优化是指同时优化多个目标且目标间相互矛盾的问题，该问题通常是 NP 难的，无法获取唯一最优解，而是一组次优解[31]. 不失一般性地以最小化问题为例，给出多目标优化问题的定义.

**定义 1.** 多目标优化问题. 给定一个具有 $n$ 维决策变量，$m(m\geq 2)$ 个目标的优化问题，其数学模型的定义如式（1）所示：

$$\min \boldsymbol{F}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x}))^{\mathrm{T}}, \boldsymbol{x} \in \Omega, \quad (1)$$

其中 $\boldsymbol{x}=(x_1, x_2, \cdots, x_n)$，$\boldsymbol{x}$ 为决策向量，$\Omega$ 为决策空间；

$\boldsymbol{F}$:$\Omega\to\Lambda$, $\boldsymbol{F}$ 为目标函数向量, $\Lambda$ 为由 $m$ 个优化目标构成的目标空间.

**定义 2.** 帕累托支配. 给定 2 个解 $\boldsymbol{x}$ 和 $\boldsymbol{y}$, 当满足如式(2)所示约束条件时, 则称 $\boldsymbol{x}$ 支配 $\boldsymbol{y}$, 表示为 $\boldsymbol{x}\prec\boldsymbol{y}$.

$$\begin{cases}\forall i\in\{1,2,\cdots,m\}, f_i(\boldsymbol{x})\leqslant f_i(\boldsymbol{y}),\\ \exists i\in\{1,2,\cdots,m\}, f_j(\boldsymbol{x})<f_j(\boldsymbol{y}).\end{cases}\tag{2}$$

**定义 3.** 帕累托最优解. 给定解 $\boldsymbol{x}\in\Omega$, 当且仅当不存在另一个解支配它, 即 $\nexists\boldsymbol{y}\in\Omega:\boldsymbol{y}\prec\boldsymbol{x}$ 时, $\boldsymbol{x}$ 为帕累托最优解.

**定义 4.** 帕累托解集(Pareto set, *PS*). 决策空间 $\Omega$ 中所有帕累托最优解的集合称为帕累托解集, 如式(3)所示:

$$PS=\{\boldsymbol{x}\in\Omega|\nexists\boldsymbol{y}\in\Omega,\boldsymbol{y}\prec\boldsymbol{x}\}.\tag{3}$$

**定义 5.** 帕累托前沿(Pareto front, *PF*). *PS* 对应的目标向量集合称为帕累托前沿, 如式(4)所示:

$$PF=\{\boldsymbol{F}(\boldsymbol{x})\in\Lambda|\boldsymbol{x}\in PS\}.\tag{4}$$

多目标优化旨在获取靠近真实 *PF* 的解集 *S*, 从 2 个方面评估解集 *S* 的质量: 收敛性, 即解集 *S* 与真实 *PF* 的接近程度; 分布性, 即解集 *S* 在目标空间中的散布程度.

### 1.3 ALO

ALO 通过蚂蚁的随机游走模拟蚁狮使用陷阱诱捕蚂蚁的过程, 实现个体解的搜索更新. 蚂蚁各维度的随机游走方法如式(5)所示:

$$\boldsymbol{X}(t)=(0,cusum(r_{t_1}(rand)),cusum(r_{t_2}(rand)),\cdots,cusum(r_{t_T}(rand))),\tag{5}$$

其中 $t$ 为当前迭代次数, $T$ 为最大迭代次数, $\boldsymbol{X}(t)$ 表示随机游走位置, *cusum* 表示随机游走步长的累加和, $r_t(rand)=\begin{cases}1, & rand>0.5\\ -1, & rand\leqslant 0.5\end{cases}$ 用于生成随机游走步长, *rand* 表示位于 [0,1] 之间均匀分布的随机数.

在随机游走过程中, 蚂蚁逐渐滑入陷阱, 其游走边界随之缩小, 如式(6)所示:

$$\boldsymbol{c}^t=\frac{\boldsymbol{c}}{I},\boldsymbol{d}^t=\frac{\boldsymbol{d}}{I}.\tag{6}$$

其中 $\boldsymbol{c}$ 和 $\boldsymbol{d}$ 表示个体各维度值的上界和下界, $\boldsymbol{c}^t$ 和 $\boldsymbol{d}^t$ 分别表示第 $t$ 次迭代中蚂蚁各维度值搜索范围的上界和下界, 缩小程度 $I$ 的计算如式(7)所示:

$$I=10^w\frac{t}{T},\tag{7}$$

其中倍率因子 $w$ 的值取决于当前迭代次数 $t$. $t\leqslant 0.1T$ 时, $w=0$; $t>0.1T$ 时, $w=2$; $t>0.5T$ 时, $w=3$; $t>0.75T$ 时, $w=4$; $t>0.9T$ 时, $w=5$; $t>0.95T$ 时, $w=6$. 可见随着 $t$ 的增加, $10^w$ 和 $I$ 值呈递增趋势, 而 $\boldsymbol{c}^t$ 和 $\boldsymbol{d}^t$ 呈递减趋势.

ALO 将适应度最优的蚁狮选为精英蚁狮, 每只蚂蚁通过轮盘赌方法选择 1 个蚁狮, 围绕精英蚁狮和轮盘赌蚁狮进行随机游走, 并根据式(8)更新位置:

$$\boldsymbol{A}^t=\frac{\boldsymbol{R}_e^t+\boldsymbol{R}_r^t}{2},\tag{8}$$

其中 $\boldsymbol{A}^t$ 为第 $t$ 次迭代蚂蚁的位置向量, $\boldsymbol{e}$ 和 $\boldsymbol{r}$ 分别表示精英蚁狮和轮盘赌蚁狮, $\boldsymbol{R}_e^t$ 为第 $t$ 次迭代蚂蚁围绕 $\boldsymbol{e}$ 进行随机游走得到的向量, $\boldsymbol{R}_r^t$ 为第 $t$ 次迭代蚂蚁围绕 $\boldsymbol{r}$ 进行随机游走所得向量.

## 2 SAMO

为了充分利用元特征的互补性和元算法的多样性来提升算法选择性能, SAMO 设计算法选择模型, 并提出多目标混合蚁狮优化算法对模型进行求解, 从而构建准确性和多样性较强的集成元算法.

使用 SAMO 进行算法选择的流程如图 2 所示. 将元数据集输入 SAMO 后, SAMO 利用所提优化算法对模型进行优化, 输出集成元算法集合, 然后从其中选择一个集成元算法用于预测新数据集的最优算法.

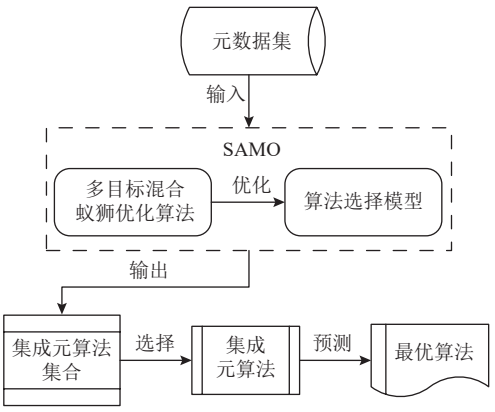


Fig. 2 Algorithm selection process of SAMO

图 2 SAMO 算法选择流程

### 2.1 算法选择模型

#### 2.1.1 模型优化目标

模型使用分类错误率(error rate, *ER*), 评估集成元算法的准确性.

设有 $m$ 个测试元实例 $X=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_m\}$, 各个元实例对应的最优算法标签为 $Y=\{y_1,y_2,\cdots,y_m\}$, 且有 $y_k\in A,1\leqslant k\leqslant m$, 其中 $A=\{a_1,a_2,\cdots,a_l\}$ 为包含 $l$ 个候选算法标签的集合, *ER* 的计算公式为

$$ER_E=\frac{1}{m}\sum_{i=1}^{m}e(\hat{y}_i,y_i),\tag{9}$$

其中 $E$ 表示集成元算法, $e(\hat{y}_i,y_i)=\begin{cases}1,\hat{y}_i\neq y_i\\ 0,\hat{y}_i=y_i\end{cases}$, $\hat{y}_i$ 和 $y_i$ 分

别表示第 $i$ 个测试元实例的预测最优算法标签和真实最优算法标签. 通过加权投票法，利用集成权重 $W=\{w_1, w_2,\cdots, w_v\}$ 组合 $v$ 个基分类器 $B=\{b_1, b_2,\cdots, b_v\}$ 构建集成元算法 $E$, $E$ 对测试元实例 $\boldsymbol{x}_k\in X$ 进行预测，如式(10)所示：

$$E(\boldsymbol{x}_k)=\arg\max_{a\in A}\left(\sum_{i=1}^{v} w_i I(b_i(\boldsymbol{x}_k)=a)\right). \tag{10}$$

其中 $E(\boldsymbol{x}_k)$ 表示预测最优算法结果；$a$ 为 $A$ 中的一个候选算法标签；$b_i$ 表示 $B$ 中的第 $i$ 个基分类器，$b_i(\boldsymbol{x}_k)$ 表示对 $\boldsymbol{x}_k$ 的预测结果；$I(\,)$ 为示性函数，当其中的表达式逻辑为真时，$I(\,)=1$，否则 $I(\,)=0$；$w_i$ 为第 $i$ 个基分类器的集成权重；函数 $\arg\max_{a\in A}(\,)$ 表示取使其内部表达式最大的算法标签 $a$. $ER$ 值越小，表示集成元算法的性能越好.

采用不同的多样性指标(diversity indicator, *DI*)，度量集成元算法的多样性，包括 Q 统计量(Q statistic, *QS*)、K 统计量(K statistic, *KS*)、相关系数(correlation coefficient, *CC*)、一致度量(agreement measure, *AM*)和双错测度(double fault, *DF*).

基分类器 $b_i$ 和 $b_j$ 对 $X$ 的预测结果列联表如表 1 所示.

**Table 1　Contingency Table of Prediction Results**

**表 1　预测结果列联表**

| $b_j$ 预测结果 | $b_i$ 预测结果 | |
|---|---|---|
| | $b_i(\boldsymbol{x}_k)=y_k$ | $b_i(\boldsymbol{x}_k)\neq y_k$ |
| $b_j(\boldsymbol{x}_k)=y_k$ | $c$ | $p$ |
| $b_j(\boldsymbol{x}_k)\neq y_k$ | $q$ | $d$ |

表 1 中，$c$ 表示 $b_i$ 和 $b_j$ 均预测正确的元实例数；$p$ 表示 $b_i$ 预测正确而 $b_j$ 预测错误的元实例数；$q$ 表示 $b_i$ 预测错误而 $b_j$ 预测正确的元实例数；$d$ 表示 $b_i$ 和 $b_j$ 均预测错误的元实例数. 根据表 1，$b_i$ 和 $b_j$ 的 *QS*, *KS*, *CC*, *AM*, *DF* 指标计算公式为

$$QS_{b_i,b_j}=\frac{c\times d-p\times q}{c\times d+p\times q}, \tag{11}$$

$$KS_{b_i,b_j}=\frac{2(c\times d-p\times q)}{(c+p)(p+d)+(c+q)(q+d)}, \tag{12}$$

$$CC_{b_i,b_j}=\frac{c\times d-p\times q}{\sqrt{(c+p)(c+q)(q+d)(p+d)}}, \tag{13}$$

$$AM_{b_i,b_j}=\frac{c+d}{m}, \tag{14}$$

$$DF_{b_i,b_j}=\frac{d}{m}. \tag{15}$$

式(11)~(15)所述的指标值越小代表基分类器多样性越强，除 *AM* 和 *DF* 指标的值域为 [0,1] 外，其余指标的值域均为 [−1,1]；此外，这 5 个指标均为成对指标，即满足 $DI_{b_i,b_j}=DI_{b_j,b_i}$. 集成元算法 $E$ 的 *DI* 值为所有成对基分类器 *DI* 值的平均值，如式(16)所示：

$$DI_E=\frac{\sum_{j=i+1}^{v}\sum_{i=1}^{v-1} DI_{b_i,b_j}}{(v^2-v)/2}. \tag{16}$$

综上，算法选择模型的目标函数向量为

$$\boldsymbol{F}_{\mathrm{SAMO}}=\min(ER_E, DI_E). \tag{17}$$

#### 2.1.2　模型设计

模型从元特征和元算法 2 个方面提升算法选择性能. 在元特征方面，选择互补性较强的元特征子集，从而有效利用元特征；在元算法方面，通过选择性集成方法，对异构基分类器进行集成，构建具有较强泛化性能和多样性的集成元算法. 为了利用不同基分类器的优势和多样性，采用 *k*NN、支持向量机(support vector machine, SVM)和 CART 作为基分类器，且本文设置每种基分类器的个数相等.

在对元数据集进行元特征选择的基础上，算法选择模型构建集成元算法的过程如图 3 所示. 首先使用自助法对训练集进行若干次抽样生成多个训练子集；然后运用基分类器在训练子集上进行训练形成基分类器集合；最后通过选择性集成方法，从集合中选择准确性和多样性较强的基分类器并基于加权投票法策略进行组合，得到集成元算法.

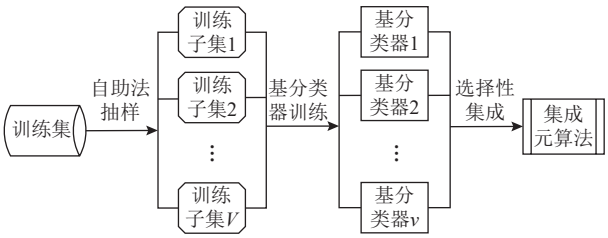


Fig. 3　Construction process of ensemble meta-learner

图 3　集成元算法构建过程

### 2.2　多目标混合蚁狮优化算法

#### 2.2.1　混合编码机制

多目标混合蚁狮优化算法针对元特征选择的离散特性(元特征选择与否)，使用离散型编码选择元特征子集；采用连续型编码构建集成元算法，该连续型编码包括 1 个用于控制基分类器选择概率的选择阈值编码，以及若干个基分类器权重编码，用于选择基分类器的训练子集并生成集成权重.

个体的混合编码方式如图 4 所示. 设元数据集含有 $M$ 个元特征，则个体的离散型编码数为 $M$；设基分类器个数为 $V$，则自助法抽样生成的训练子集个数

为 $V$，个体依次为 $k$NN，SVM，CART 使用 $V$ 个编码选择训练子集和生成集成权重，因此基分类器权重编码数为 $3V$，个体的连续型编码数为 $3V$+1.

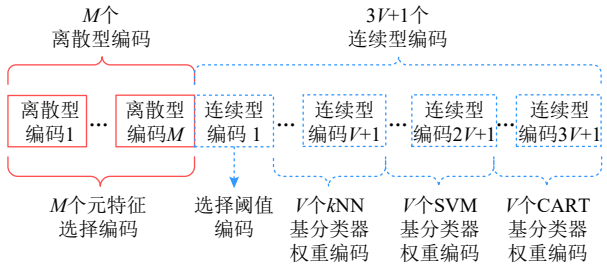


Fig. 4 Coding pattern of individuals

图 4 个体编码方式

根据混合编码机制，对个体各维度的编码值进行初始化，如式(18)所示：

$$A_i=\begin{cases}R(rand), i\leqslant M,\\ rand, i>M,\end{cases} \tag{18}$$

其中 $A_i$ 为个体第 $i$ 维的编码值，个体前 $M$ 维编码为离散型的元特征选择编码，$R(rand)=\begin{cases}1, & rand\geqslant 0.5,\\ 0, & rand<0.5,\end{cases}$ 其值为 1 表示第 $i$ 维编码对应的元特征被选中，值为 0 表示未被选中. 个体第 $M$+1 维编码为选择阈值编码，其后的 $3V$ 个编码为基分类器权重编码，前 $V$ 个基分类器权重编码为 $k$NN 选择训练子集，如式(19)所示：

$$J(A_i)=\begin{cases}1, A_i\geqslant A_{M+1},\\ 0, A_i<A_{M+1},\end{cases}\quad M+1<i\leqslant M+V+1, \tag{19}$$

其中 $A_{M+1}$ 为选择阈值编码值，$J(A_i)$= 1 表示第 $i$ 维编码对应的训练子集被选中，$J(A_i)$= 0 表示未被选中，以此类推可得 SVM 和 CART 的训练子集. 3 种基分类器分别在选择的训练子集上独立训练，得到基分类器集合 $B=\{b_1,b_2,\cdots,b_v\}$，通过基分类器权重编码值生成这 3 种基分类器集合对应的集成权重 $W=\{w_1,w_2,\cdots,w_v\}$，如式(20)所示：

$$w_i=\frac{A_i}{\sum_{j=1}^{v}A_j}, \tag{20}$$

其中 $w_i$ 为第 $i$ 个基分类器的集成权重，$A_i$ 为第 $i$ 个基分类器对应的基分类器权重编码值，$A_j$ 为第 $j$ 个基分类器对应的基分类器权重编码值. 通过式(20)的归一化方法，使得集成权重和为 1，即 $\sum_{i=1}^{v}w_i=1$.

### 2.2.2 增强游走策略

在混合编码机制的基础上，采用增强游走策略对个体进行搜索更新.

在迭代过程中，个体的离散型编码使用离散随机游走方法，如式(21)所示：

$$R_{\mathrm{dis},i}^{t}=\begin{cases}L_{\mathrm{dis},i}^{t}, & X^t/t\geqslant m(t),\\ 1-L_{\mathrm{dis},i}^{t}, & X^t/t<m(t).\end{cases} \tag{21}$$

其中 $R_{\mathrm{dis},i}^{t}$ 为第 $t$ 次迭代蚂蚁的离散型编码进行离散随机游走后所得离散向量的第 $i$ 维元素值；$L_{\mathrm{dis},i}^{t}$ 为第 $t$ 次迭代蚂蚁所围绕游走的蚁狮第 $i$ 维离散型编码值；$X^t$ 为式(5)中第 $t$ 次迭代对应的随机游走步长累加和，由式(5)可知其值在 $[-t, t]$ 之间，由此可得 $X^t/t$ 位于 $[-1,1]$ 之间，取反阈值 $m(t)$ 计算如式(22)所示：

$$m(t)=(2rand-1)\cos\frac{\pi(t-1)}{2(T-1)}. \tag{22}$$

其中 $\cos\frac{\pi(t-1)}{2(T-1)}$ 在 $[0,1]$ 之间随着迭代次数 $t$ 的增加呈现非线性递减趋势，$(2rand-1)$ 的值域为 $[-1,1]$，使得 $m(t)$ 在 $[-1,1]$ 之间随 $t$ 值增加呈现随机性的非线性递减趋势，如图 5 所示.

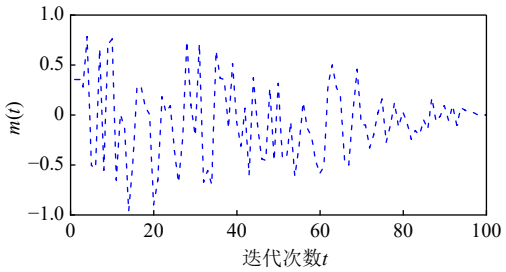


Fig. 5 Change trend of $m(t)$

图 5 $m(t)$ 变化趋势

随机游走后，蚂蚁离散型编码值更新为

$$A_{\mathrm{dis},i}^{t}=\begin{cases}R_{\mathrm{dis},\boldsymbol{e},i}^{t}, rand\geqslant 0.5,\\ R_{\mathrm{dis},\boldsymbol{r},i}^{t}, rand<0.5,\end{cases} \tag{23}$$

其中 $A_{\mathrm{dis},i}^{t}$ 为第 $t$ 次迭代蚂蚁的第 $i$ 维离散型编码值，$R_{\mathrm{dis},\boldsymbol{e},i}^{t}$ 和 $R_{\mathrm{dis},\boldsymbol{r},i}^{t}$ 分别为蚂蚁在第 $t$ 次迭代围绕精英蚁狮和轮盘赌蚁狮进行离散随机游走所得离散向量的第 $i$ 维元素值.

对于个体的连续型编码，在其游走更新过程中引入随机性. ALO 中每只蚂蚁的搜索边界 $\boldsymbol{c}^t$ 和 $\boldsymbol{d}^t$ 变化趋势相同，种群多样性不足. 为了增强所提算法的种群多样性，对式(6)进行改进，如式(24)所示：

$$\begin{aligned}\boldsymbol{c}^t&=\frac{\boldsymbol{c}}{I}\left(\cos\frac{\pi(t-1)}{3(T-1)}+rand-0.5\right),\\ \boldsymbol{d}^t&=\frac{\boldsymbol{d}}{I}\left(\cos\frac{\pi(t-1)}{3(T-1)}+rand-0.5\right),\end{aligned} \tag{24}$$

其中 $\cos\frac{\pi(t-1)}{3(T-1)}$ 在 $[0.5,1]$ 之间随着迭代次数 $t$ 的增加呈现非线性递减趋势，使得 $\left(\cos\frac{\pi(t-1)}{3(T-1)}+rand-0.5\right)$ 在 $[0,1.5]$ 之间呈现随机性的非线性递减趋势，从而在 $\boldsymbol{c}^t$ 和 $\boldsymbol{d}^t$ 的变化过程中引入一定随机性. 本文所提算法与 ALO 的搜索边界变化趋势对比如图 6 所示.

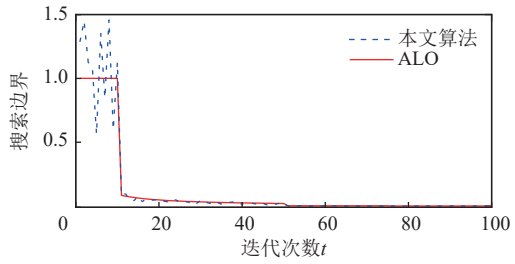


Fig. 6 Change trend of search boundary

图 6 搜索边界变化趋势

通过上述设计，增强游走策略对个体不同类型的编码进行搜索更新，保留了蚂蚁搜索边界变化的递减趋势，同时在该过程中引入了随机性，增加了算法的种群多样性.

2.2.3 偏好精英选择机制

算法将帕累托解作为蚁狮保存至外部存档 $S$，为了提升算法的寻优能力，分别以不同优化目标为偏好，从存档 $S$ 中选择在该目标上最优的精英蚁狮. 为了增强解的分布性，通过小生境技术计算蚁狮解的稀疏度从而量化其分布性，然后利用稀疏度通过轮盘赌选择蚁狮，蚁狮解 $\boldsymbol{x}_S$ 的稀疏度为

$$s(\boldsymbol{x}_S,\varphi)=\frac{|S|-1}{|\{\boldsymbol{y}_S\in S\,|\|\boldsymbol{F}(\boldsymbol{x}_S)-\boldsymbol{F}(\boldsymbol{y}_S)\|<\varphi\}|}. \tag{25}$$

其中 $s(\boldsymbol{x}_S,\varphi)$表示给定半径$\varphi$的小生境范围内解 $\boldsymbol{x}_S$ 的稀疏度，$\boldsymbol{y}_S$ 表示存档 $S$ 中的另一个解，半径$\varphi$计算为

$$\varphi=\frac{\sum_{j=i+1}^{m}\sum_{i=1}^{m-1}\|\boldsymbol{F}(\boldsymbol{e}_i)-\boldsymbol{F}(\boldsymbol{e}_j)\|}{(m^2-m)c/2},\ 0<c\leqslant|S|, \tag{26}$$

其中 $m(m\geq 2)$为优化目标个数，$\boldsymbol{e}_i$ 和 $\boldsymbol{e}_j$ 分别表示第 $i$ 个和第 $j$ 个优化目标对应的精英蚁狮，$\|\boldsymbol{F}(\boldsymbol{e}_i)-\boldsymbol{F}(\boldsymbol{e}_j)\|$ 计算二者目标函数值向量的欧氏距离，$c$ 为常数；$\|\boldsymbol{F}(\boldsymbol{x}_S)-\boldsymbol{F}(\boldsymbol{y}_S)\|$计算解 $\boldsymbol{x}_S$ 和解 $\boldsymbol{y}_S$ 目标函数值向量的欧氏距离，其值小于半径$\varphi$时表示 $\boldsymbol{y}_S$ 是 $\boldsymbol{x}_S$ 的相邻解. 给定解在存档中的相邻解越多，则该解的稀疏度越低，分布性越差. 以 2 个优化目标为例，稀疏度计算示意如图 7 所示.

在迭代过程中，对于每个优化目标，每只初始化蚂蚁根据蚁狮的稀疏度通过轮盘赌选择 1 个蚁狮，围绕该优化目标的精英蚁狮和轮盘赌蚁狮进行随机游走，产生新的个体解. 由此可得个体数是初始化种群的新蚂蚁种群的 $m$ 倍，将新蚂蚁种群加入外部存档，筛选保留存档中的帕累托解作为新一代蚁狮.

通过偏好精英选择机制，分别围绕不同优化目标上的最优个体进行迭代更新，增强算法的寻优性能；采用轮盘赌方法以较大概率选择分布性较好的个体，并围绕该个体进行搜索，使解的分布性更强.

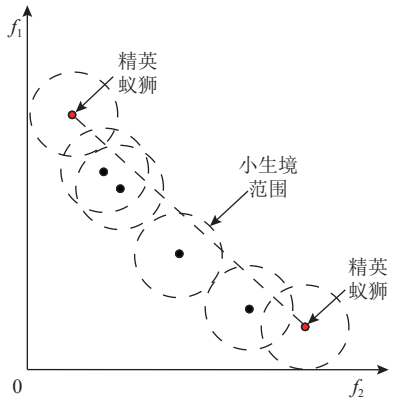


Fig. 7 Schematic diagram of sparsity calculation

图 7 稀疏度计算示意图

### 2.3 SAMO 方法整体描述

SAMO 流程如图 8 所示. 方法首先输入元数据集；然后根据混合编码机制初始化蚂蚁种群；计算蚂蚁的适应度，通过帕累托支配关系从中选出帕累托解作为蚁狮保存至外部存档；按照偏好精英选择机制，从蚁狮中选出准确性和多样性优化目标上的精英个体，分别称为准确性精英蚁狮和多样性精英蚁狮，并计算蚁狮的稀疏度；在迭代过程中，每只初始化蚂蚁根据蚁狮稀疏度进行 2 次轮盘赌选择蚁狮，围绕准确性精英蚁狮和第 1 次轮盘赌蚁狮，以及围绕多样性精英蚁狮和第 2 次轮盘赌蚁狮，基于增强游走策略进行随机游走，生成 2 只新的蚂蚁；计算新蚂蚁种群的适应度并将其加入存档，更新存档和精英蚁狮；最后，达到最大迭代时，输出蚁狮构建的集成元算法集合. 在该过程中，个体的适应度为集成元算法的准确性和多样性指标值.

SAMO 方法的伪代码如算法 1 所示.

**算法 1.** SAMO 方法.

输入：元数据集 $D$、基分类器个数 $V$、多样性指标 $DI$、最大迭代次数 $T$、种群规模 $N$；

输出：蚁狮构建的集成元算法集合 $E_S$.

① *initAnts*←*initialize*($D$,$V$,$N$)；/*获取 $D$ 的元特征数 $M$，根据 $M$ 和 $V$ 确定个体不同类型编码的维度数，基于混合编码机制，通过式(18)获取 $N$ 个蚂蚁的初始化种群 *initAnts**/

② *trSubsets*←*bootStrap*($V$)；/*对训练集使用自助法抽样生成 $V$ 个训练子集 *trSubsets**/

③ for each ant $\boldsymbol{A}$ in *initAnts*

④ *mfSubset*←*mfSelection*($\boldsymbol{A}$)；/*根据蚂蚁 $\boldsymbol{A}$ 的离散型编码选择元特征子集 *mfSubSet**/

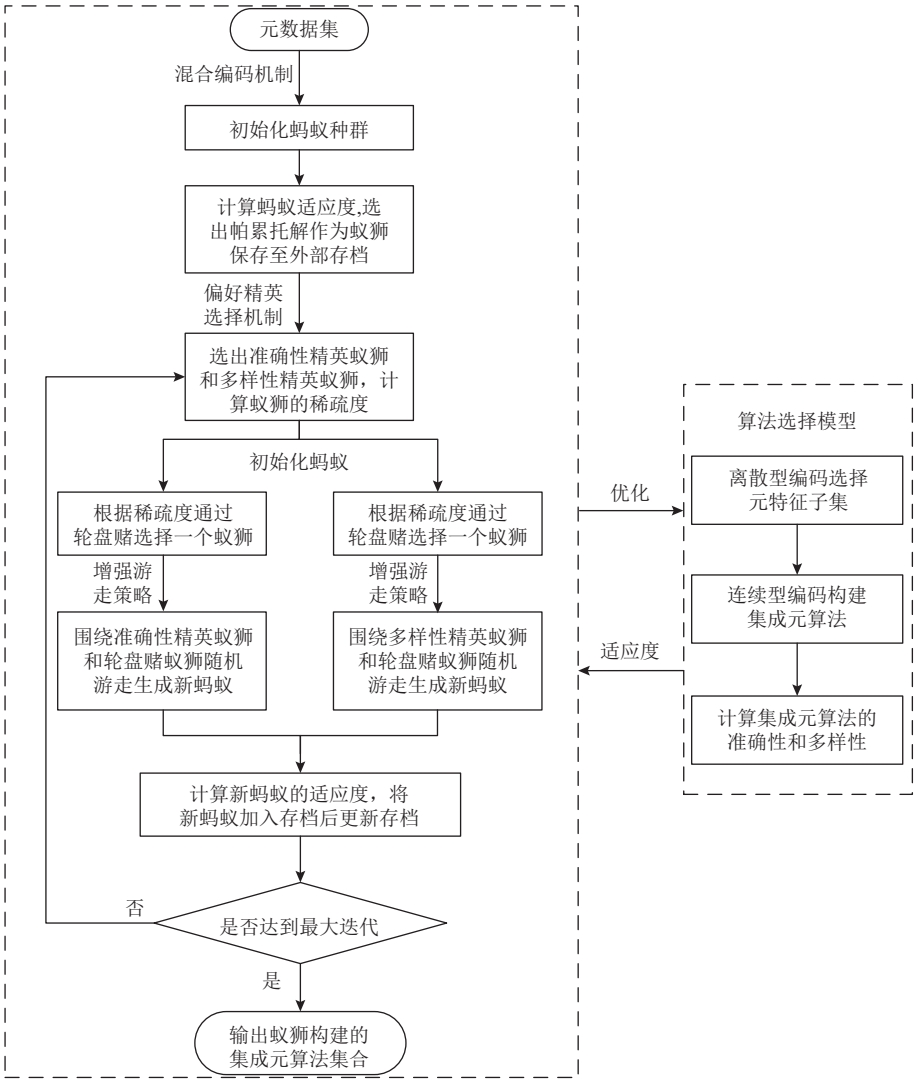


Fig. 8 Process of SAMO

图 8 SAMO 流程

⑤ $E_A$←*selEnsemble*(***A***,*mfSubset*,*trSubsets*); /*在使用 *mfSubSet* 的基础上, 根据 ***A*** 的连续型编码, 通过式(19)(20)为基分类器选择训练子集和生成集成权重, 训练基分类器并集成得到集成元算法 $E_A$*/

⑥ $\boldsymbol{F}_A$=[$ER_{EA}$,$DI_{EA}$]←*caculateFit*($E_A$, *DI*); /*通过式(9)(16)计算 $E_A$ 的 *ER* 值和 *DI* 值作为 ***A*** 的适应度值 $\boldsymbol{F}_A$*/

⑦ end for

⑧ *S*←*updateArchive*(*initAnts*); /*将 *initAnts* 加入外部存档 *S*, 根据式(2)计算解的支配关系, 保留存档中的帕累托解作为蚁狮*/

⑨ $\boldsymbol{e}_{ER}$,$\boldsymbol{e}_{DI}$←*updateElites*(*S*); /*从 *S* 中选择 *ER* 值最优的准确性精英蚁狮 $\boldsymbol{e}_{ER}$ 和 *DI* 值最优的多样性精英蚁狮 $\boldsymbol{e}_{DI}$*/

⑩ while *iter*=1,*iter*≤*T*

⑪ *sparS*←*caculateSparsity*(*S*); /*通过式(25)计算存档中蚁狮的稀疏度 *sparS**/

⑫ for each ant ***A*** in *initAnts*

⑬ ***r***←*rouletteWheel*(***A***,*sparS*); /*根据稀疏度 *sparS* 通过轮盘赌选择蚁狮 ***r****/

⑭ $\boldsymbol{R}_e$,$\boldsymbol{R}_r$←*randomWalk*(***A***,$\boldsymbol{e}_{ER}$,***r***); /*根据式(21)~(24)围绕蚁狮 $\boldsymbol{e}_{ER}$ 和 ***r*** 进行随机游走产生向量 $\boldsymbol{R}_e$ 和 $\boldsymbol{R}_r$*/

⑮ ***A′***=*updateAnt*($\boldsymbol{R}_e$,$\boldsymbol{R}_r$); /*使用 $\boldsymbol{R}_e$ 和 $\boldsymbol{R}_r$, 通过式(8)(23)生成新蚂蚁个体 ***A′****/

⑯ 将步骤⑭~⑯中的 $\boldsymbol{e}_{ER}$ 和 ***A′*** 替换为 $\boldsymbol{e}_{DI}$ 和 ***A″*** 后, 重复步骤⑭~⑯;

⑰ *newAnts*←*newAnts*∪{***A′***,***A″***}; /*将 ***A′*** 和 ***A″*** 加入新蚂蚁种群 *newAnts**/

⑱ end for

⑲ 将步骤④~⑩中的 *initAnts* 替换为 *newAnts* 后，重复步骤④~⑩；

⑳ *iter*=*iter*+1；

㉑ end while

㉒ 输出 $S$ 中蚁狮构建的集成元算法集合 $E_S$.

现对 SAMO 的时间复杂度进行分析，设元特征数为 $M$，基分类器个数为 $V$，可得个体维度数 $D=M+3V+1$；设种群规模为 $N$，最大迭代次数为 $T$，则蚂蚁随机游走进行迭代的时间复杂度为 $O(2N\times D\times T)$，计算解的支配关系和稀疏度的时间复杂度为 $O(N^2\times T)$. 综上所述，SAMO 的时间复杂度为 $O(N\times T\times(N+2D))$.

## 3 实验与结果分析

### 3.1 实验设置

#### 3.1.1 数据集

由于分类算法应用的广泛性，通过分类算法选择问题进行测试实验. 实验使用来自于 UCI[32], KEEL[33], StatLib[34], OpenML[35] 的 260 个分类数据集，这些数据集的数据来源领域各异，实例数从 13~149 332 不等，属性数从 1~1 558 不等，类数从 2~188 不等，具有一定的差异性，构成多样化的数据集，从而能够有效评估方法的性能. 实验数据集信息如表 2 所示.

**Table 2 Information of Experimental Datasets**

**表 2 实验数据集信息**

| 序号 | 数据集 | 属性数 | 实例数 | 类数 | 序号 | 数据集 | 属性数 | 实例数 | 类数 | 序号 | 数据集 | 属性数 | 实例数 | 类数 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | abalone | 8 | 4 177 | 29 | 88 | divorce | 54 | 170 | 2 | 175 | online-shoppers | 17 | 12 330 | 2 |
| 2 | absenteeism | 20 | 740 | 18 | 89 | dna | 180 | 3 186 | 3 | 176 | optdigits | 64 | 3 823 | 10 |
| 3 | ada-agnostic | 48 | 4 562 | 2 | 90 | dry-bean | 16 | 13 611 | 7 | 177 | ozone-1hr | 72 | 2 536 | 2 |
| 4 | advertisement | 1 558 | 3 279 | 2 | 91 | echocardiogram | 11 | 75 | 3 | 178 | ozone-8hr | 72 | 2 534 | 2 |
| 5 | aids | 4 | 50 | 2 | 92 | ecoli | 7 | 336 | 8 | 179 | page-blocks | 10 | 5 473 | 5 |
| 6 | allrep | 29 | 3 772 | 4 | 93 | eeg-eyestate | 14 | 14 980 | 2 | 180 | parkinson-speech | 26 | 1 040 | 2 |
| 7 | amazon-employ | 9 | 32 769 | | 94 | electricity | 8 | 45 132 | 2 | 181 | pc1 | 21 | 1 109 | 2 |
| 8 | acd-assessment | 15 | 13 | 4 | 95 | energy-eff | 9 | 768 | 37 | 182 | pc3 | 37 | 1 563 | 2 |
| 9 | acd-authorship | 70 | 841 | 4 | 96 | engine1 | 5 | 383 | 3 | 183 | pc4 | 37 | 1 458 | 2 |
| 10 | acd-bankruptcy | 6 | 50 | 2 | 97 | eucalyptus | 19 | 736 | 5 | 184 | penbased | 16 | 10 992 | 10 |
| 11 | acd-birthday | 3 | 365 | 7 | 98 | fabert | 800 | 8 237 | 7 | 185 | phishing-websites | 30 | 2 456 | 2 |
| 12 | acd-bondrate | 11 | 57 | 5 | 99 | first-order | 51 | 6 118 | 6 | 186 | phoneme | 5 | 5 404 | 2 |
| 13 | acd-boxing1 | 3 | 120 | 2 | 100 | flag | 28 | 194 | 8 | 187 | pima | 8 | 768 | 2 |
| 14 | acd-boxing2 | 3 | 132 | 2 | 101 | flare | 11 | 1 066 | 6 | 188 | polish-bankruptcy1 | 64 | 7 027 | 2 |
| 15 | acd-braziltour | 8 | 412 | 7 | 102 | gas-drift | 128 | 13 910 | 6 | 189 | polish-bankruptcy5 | 64 | 5 500 | 2 |
| 16 | acd-broadway | 9 | 95 | 5 | 103 | german | 20 | 1 000 | 2 | 190 | popularkids | 10 | 478 | 3 |
| 17 | acd-broadwaym | 7 | 285 | 7 | 104 | gesture-phase | 32 | 9 873 | 5 | 191 | post-operative | 8 | 90 | 3 |
| 18 | acd-chall101 | 2 | 138 | 2 | 105 | gina-prior2 | 784 | 3 468 | 10 | 192 | primary-tumor | 17 | 339 | 22 |
| 19 | acd-creditscore | 6 | 100 | 2 | 106 | glass | 9 | 214 | 7 | 193 | prnn-fglass | 9 | 214 | 6 |
| 20 | acd-currency | 3 | 31 | 7 | 107 | haberman | 3 | 306 | 2 | 194 | ring | 20 | 7 400 | 2 |
| 21 | acd-cyyoung8 | 10 | 97 | 2 | 108 | hayes-roth-test | 4 | 28 | 4 | 195 | risk-factors | 35 | 858 | 26 |
| 22 | acd-cyyoung9 | 10 | 92 | 2 | 109 | hayes-roth-train | 4 | 132 | 4 | 196 | rmftsa-sleep | 2 | 1 024 | 4 |
| 23 | acd-dmft | 4 | 797 | 6 | 110 | hcv-egyptian | 28 | 1 385 | 4 | 197 | robot-failures-lp4 | 90 | 117 | 3 |
| 24 | acd-draft | 4 | 365 | 12 | 111 | heart-statlog | 13 | 270 | 2 | 198 | saheart | 9 | 462 | 2 |
| 25 | acd-esr | 2 | 32 | 2 | 112 | helena | 27 | 65 196 | 100 | 199 | sat11-hand-runtime | 115 | 296 | 14 |
| 26 | acd-germangss | 5 | 400 | 4 | 113 | hepatitis | 19 | 155 | 2 | 200 | satimage | 36 | 6 435 | 7 |
| 27 | acd-halloffame | 17 | 1 340 | 3 | 114 | hill-valley | 100 | 1 212 | 2 | 201 | sat-test | 36 | 2000 | 6 |
| 28 | acd-homerun | 26 | 162 | 2 | 115 | horse-colic-test | 27 | 68 | 2 | 202 | sat-train | 36 | 4 435 | 6 |
| 29 | acd-lawsuit | 4 | 264 | 2 | 116 | horse-colic-train | 27 | 300 | 2 | 203 | seeds | 7 | 210 | 3 |

**表 2（续）**

| 序号 | 数据集 | 属性数 | 实例数 | 类数 | 序号 | 数据集 | 属性数 | 实例数 | 类数 | 序号 | 数据集 | 属性数 | 实例数 | 类数 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 30 | acd-mapleleafs | 1 | 84 | 3 | 117 | house-votes | 16 | 232 | 2 | 204 | semeion | 256 | 1 593 | 10 |
| 31 | acd-marketing | 32 | 310 | 5 | 118 | ilpd | 10 | 583 | 2 | 205 | sensor-readings-24 | 24 | 5 456 | 4 |
| 32 | acd-supreme | 7 | 4 052 | 10 | 119 | image-seg-test | 19 | 210 | 7 | 206 | sensor-readings-4 | 4 | 5 456 | 4 |
| 33 | acd-votesurvey | 4 | 48 | 4 | 120 | image-seg-train | 19 | 2 100 | 7 | 207 | servo | 4 | 167 | 2 |
| 34 | anneal | 38 | 798 | 6 | 121 | indian-pines | 220 | 9 144 | 8 | 208 | shuttle | 9 | 58 000 | 7 |
| 35 | anomalydata-5 | 4 | 1 050 | 2 | 122 | internet-usage | 70 | 10 108 | 46 | 209 | shuttle-landing | 6 | 15 | 2 |
| 36 | anomalydata-5h | 10 | 1 050 | 2 | 123 | ionosphere | 34 | 351 | 2 | 210 | smartphone-har | 66 | 180 | 6 |
| 37 | appendicitis | 7 | 106 | 2 | 124 | iris | 4 | 150 | 3 | 211 | socmob | 5 | 1 156 | 2 |
| 38 | arrhythmia | 279 | 452 | 16 | 125 | isolet1234 | 617 | 6 238 | 26 | 212 | sonar | 60 | 208 | 2 |
| 39 | artificial-charac | 7 | 10 218 | 10 | 126 | isolet5 | 617 | 1 559 | 26 | 213 | soybean-large | 35 | 307 | 19 |
| 40 | asp-potassco | 140 | 1 294 | 11 | 127 | japanese-vowels | 14 | 9 961 | 9 | 214 | soybean-small | 35 | 47 | 4 |
| 41 | audiology | 69 | 226 | 24 | 128 | jungle-chess-l-e | 46 | 4 704 | 3 | 215 | spambase | 57 | 4 597 | 2 |
| 42 | australian | 14 | 690 | 2 | 129 | jungle-chess-p-l | 46 | 4 704 | 3 | 216 | spect-test | 22 | 187 | 2 |
| 43 | autism-adult | 20 | 704 | 2 | 130 | jungle-chess-r-e | 46 | 5 880 | 3 | 217 | spect-train | 22 | 80 | 2 |
| 44 | autohorse-fixed | 68 | 201 | 186 | 131 | kc1 | 21 | 2 109 | 2 | 218 | spectf-test | 44 | 269 | 2 |
| 45 | automobile | 25 | 205 | 7 | 132 | kc2 | 21 | 522 | 2 | 219 | spectf-train | 44 | 80 | 2 |
| 46 | autouniv1-1 000 | 20 | 1 000 | 2 | 133 | kr-vs-kp | 36 | 3 196 | 2 | 220 | spectrometer | 101 | 531 | 48 |
| 47 | autouniv4-2 500 | 100 | 2 500 | 3 | 134 | kropt | 6 | 28 056 | 18 | 221 | speech | 400 | 3 686 | 2 |
| 48 | autouniv6-1 000 | 40 | 1 000 | 8 | 135 | leaf | 15 | 340 | 30 | 222 | splice | 60 | 3 190 | 3 |
| 49 | autouniv6-750 | 40 | 750 | 8 | 136 | leaves-margin | 64 | 1 600 | 100 | 223 | steel-plates-faults | 27 | 1941 | 7 |
| 50 | autouniv7-1 100 | 12 | 1 100 | 5 | 137 | leaves-shape | 64 | 1 600 | 100 | 224 | student-mat | 30 | 395 | 21 |
| 51 | autouniv7-500 | 12 | 500 | 5 | 138 | leaves-texture | 64 | 1 600 | 100 | 225 | student-por | 30 | 649 | 21 |
| 52 | bach-choral | 16 | 5 665 | 102 | 139 | led24 | 24 | 3 200 | 10 | 226 | surveillance | 7 | 15 | 3 |
| 53 | balance-scale | 4 | 625 | 3 | 140 | led7digit | 7 | 500 | 10 | 227 | synthetic-control | 60 | 600 | 6 |
| 54 | ballon | 4 | 16 | 2 | 141 | lense | 5 | 24 | 3 | 228 | tae | 5 | 151 | 3 |
| 55 | banana | 2 | 5 300 | 2 | 142 | letter | 16 | 20 000 | 26 | 229 | tamilnadu | 3 | 45 781 | 20 |
| 56 | bank-marketing | 16 | 45 211 | 2 | 143 | libras-move | 90 | 360 | 15 | 230 | texture | 40 | 5 000 | 11 |
| 57 | banknote | 4 | 1 372 | 2 | 144 | lung-cancer | 56 | 32 | 3 | 231 | thyroid | 21 | 7 200 | 3 |
| 58 | biodeg | 41 | 1 055 | 2 | 145 | lupus | 3 | 87 | 2 | 232 | thyroid-allbp | 26 | 2 800 | 5 |
| 59 | blood-trans | 4 | 748 | 2 | 146 | lymphography | 18 | 148 | 4 | 233 | thyroid-allhyper | 26 | 2 800 | 5 |
| 60 | breast-cancer | 9 | 286 | 2 | 147 | madelon | 500 | 2 600 | 2 | 234 | tic-tac-toe | 9 | 958 | 2 |
| 61 | breast-cancer-w | 9 | 699 | 2 | 148 | magic | 10 | 19 020 | 2 | 235 | titanic | 3 | 2 201 | 2 |
| 62 | bupa | 6 | 345 | 2 | 149 | marketing | 13 | 8 993 | 9 | 236 | toronto-apartment | 6 | 1 124 | 188 |
| 63 | cacao | 8 | 1 795 | 42 | 150 | mc1 | 38 | 9 466 | 2 | 237 | touch2 | 10 | 265 | 8 |
| 64 | calendar-dow | 32 | 399 | 5 | 151 | meta-all | 62 | 71 | 6 | 238 | trains | 32 | 10 | 2 |
| 65 | car | 6 | 1 728 | 4 | 152 | meta-stream | 74 | 45 164 | 13 | 239 | twonorm | 20 | 7 400 | 2 |
| 66 | car-evaluation | 21 | 1 728 | 4 | 153 | mfeat-fac | 216 | 2000 | 10 | 240 | unix-user | 2 | 9 100 | 9 |
| 67 | cardiotocograph | 35 | 2 126 | 3 | 154 | mfeat-fou | 76 | 2000 | 10 | 241 | user-knowledge | 5 | 403 | 5 |
| 68 | castmetal1 | 37 | 327 | 2 | 155 | mfeat-kar | 64 | 2000 | 10 | 242 | usps | 256 | 9 298 | 10 |
| 69 | chess | 36 | 3 196 | 2 | 156 | mfeat-mor | 6 | 2000 | 10 | 243 | vehicle | 18 | 846 | 4 |
| 70 | churn | 20 | 5 000 | 2 | 157 | mfeat-pix | 240 | 2000 | 10 | 244 | vehicle-reproduced | 18 | 846 | 4 |
| 71 | clean2 | 165 | 6 598 | 2 | 158 | mfeat-zer | 47 | 2000 | 10 | 245 | volcanoes-a1 | 3 | 3 252 | 4 |
| 72 | cleveland | 13 | 297 | 5 | 159 | miceprotein | 76 | 1 080 | 8 | 246 | volcanoes-d2 | 3 | 9 172 | 4 |

表 2（续）

| 序号 | 数据集 | 属性数 | 实例数 | 类数 | 序号 | 数据集 | 属性数 | 实例数 | 类数 | 序号 | 数据集 | 属性数 | 实例数 | 类数 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 73 | click-prediction | 9 | 39 948 | 2 | 160 | micro-a2 | 20 | 20 000 | 5 | 247 | volcanoes-e2 | 3 | 1 080 | 4 |
| 74 | climate-model | 20 | 540 | 2 | 161 | micro-mass | 1 300 | 571 | 20 | 248 | vowel | 13 | 990 | 11 |
| 75 | cmc | 9 | 1 473 | 3 | 162 | monks1-test | 6 | 122 | 2 | 249 | walking-activity | 4 | 149 332 | 22 |
| 76 | cnae9 | 856 | 1 080 | 9 | 163 | monks1-train | 6 | 124 | 2 | 250 | waveform | 21 | 5 000 | 3 |
| 77 | coil2000 | 85 | 9 822 | 2 | 164 | monks2-test | 6 | 432 | 2 | 251 | waveform-noise | 40 | 5 000 | 3 |
| 78 | colleges-aaup | 14 | 1 161 | 4 | 165 | monks2-train | 6 | 169 | 2 | 252 | wdbc | 30 | 569 | 2 |
| 79 | collins | 19 | 1 000 | 30 | 166 | monks3-test | 6 | 432 | 2 | 253 | wifi-localization | 7 | 2000 | 4 |
| 80 | contraceptive | 9 | 1 473 | 3 | 167 | monks3-train | 6 | 122 | 2 | 254 | wilt | 5 | 4 339 | 2 |
| 81 | cpmp-2015 | 24 | 527 | 4 | 168 | mozilla4 | 5 | 15 545 | 2 | 255 | wine | 13 | 178 | 3 |
| 82 | credit-card | 23 | 30 000 | 2 | 169 | mushroom | 22 | 8 124 | 2 | 256 | winequality-r | 11 | 1 599 | 10 |
| 83 | crx | 15 | 653 | 2 | 170 | newthyroid | 5 | 215 | 3 | 257 | winequality-w | 11 | 4 898 | 10 |
| 84 | cylinder-bands | 19 | 539 | 2 | 171 | nursery | 8 | 12 960 | 5 | 258 | wpbc | 32 | 198 | 2 |
| 85 | dbworld-bodies | 64 | 3 721 | 2 | 172 | obs-network | 20 | 1 075 | 4 | 259 | yeast | 8 | 1 484 | 13 |
| 86 | dermatology | 34 | 366 | 6 | 173 | oil-spill | 49 | 937 | 2 | 260 | zoo | 16 | 101 | 7 |
| 87 | diggle-table-a2 | 8 | 310 | 9 | 174 | olivetti-faces | 4 096 | 400 | 40 | | | | | |

### 3.1.2 元特征

通过元特征提取工具 MFE[36] 提取常用的 150 种分类数据集元特征，包括 77 种基于统计和信息论的元特征、24 种基于决策树的元特征、14 种基于基准的元特征和 35 种基于问题复杂度的元特征. 元特征信息如表 3 所示.

**Table 3 Information of Meta-Features**

**表 3 元特征信息**

| 元特征类型 | 元特征名称 |
|---|---|
| 基于统计和信息论的元特征 | attr_conc.mean，attr_conc.sd，attr_ent.mean，attr_ent.sd，attr_to_inst，can_cor.mean，can_cor.sd，cat_to_num，class_conc.mean，class_conc.sd，class_ent，cor.mean，cor.sd，cov.mean，cov.sd，eigenvalues.mean，eigenvalues.sd，eq_num_attr，freq_class.mean，freq_class.sd，g_mean.mean，g_mean.sd，gravity，h_mean.mean，h_mean.sd，inst_to_attr，iq_range.mean，iq_range.sd，joint_ent.mean，joint_ent.sd，kurtosis.mean，kurtosis.sd，lh_trace，mad.mean，mad.sd，max.mean，max.sd，mean.mean，mean.sd，median.mean，median.sd，min.mean，min.sd，mut_inf.mean，mut_inf.sd，nr_attr，nr_bin，nr_cat，nr_class，nr_cor_attr，nr_disc，nr_inst，nr_norm，nr_num，nr_outliers，ns_ratio，num_to_cat，one_itemset.mean，one_itemset.sd，p_trace，range.mean，range.sd，roy_root，sd.mean，sd.sd，sd_ratio，skewness.mean，skewness.sd，sparsity.mean，sparsity.sd，t_mean.mean，t_mean.sd，two_itemset.mean，two_itemset.sd，var.mean，var.sd，w_lambda |
| 基于决策树的元特征 | leaves，leaves_branch.mean，leaves_branch.sd，leaves_corrob.mean，leaves_corrob.sd，leaves_homo.mean，leaves_homo.sd，leaves_per_class.mean，leaves_per_class.sd，nodes，nodes_per_attr，nodes_per_inst，nodes_per_level.mean，nodes_per_level.sd，nodes_repeated.mean，nodes_repeated.sd，tree_depth.mean，tree_depth.sd，tree_imbalance.mean，tree_imbalance.sd，tree_shape.mean，tree_shape.sd，var_importance.mean，var_importance.sd |
| 基于基准的元特征 | best_node.mean，best_node.sd，elite_nn.mean，elite_nn.sd，linear_discr.mean，linear_discr.sd，naive_bayes.mean，naive_bayes.sd，one_nn.mean，one_nn.sd，random_node.mean，random_node.sd，worst_node.mean，worst_node.sd |
| 基于问题复杂度的元特征 | c1，c2，cls_coef，density，f1.mean，f1.sd，f1v.mean，f1v.sd，f2.mean，f2.sd，f3.mean，f3.sd，f4.mean，f4.sd，hubs.mean，hubs.sd，l1.mean，l1.sd，l2.mean，l2.sd，l3.mean，l3.sd，lsc，n1，n2.mean，n2.sd，n3.mean，n3.sd，n4.mean，n4.sd，t1.mean，t1.sd，t2，t3，t4 |

### 3.1.3 候选算法

使用 8 种 sklearn[37] 机器学习平台的分类算法，包括 *k*NN、RF、SVM、逻辑回归（logistic regression, LR）、朴素贝叶斯（naive Bayes, NB）、线性判别分析（linear discriminant analysis, LDA）、ID3 决策树和多层感知机（multi-layer perceptron, MLP），以及基于 Keras[38] 构建的卷积神经网络（convolutional neural network, CNN）作为候选算法. 基于 sklearn 实现的算法均采用其默认参数设置，对于 CNN，使用 1 个含有 32 个卷积核的 1 维卷积层、1 个全连接层以及 1 个最大池化层构成其隐藏层，并设置其卷积层和全连接层使用 ReLU（rectified linear unit）激活函数，输出层使用 softmax 激活函数.

### 3.1.4 候选算法性能测度

使用准确率（*Acc*）、查准率（*Pre*）、查全率（*Rec*）、*F*1 得分（*F*1）和 *ARR*（adjusted ratio of ratios）[39] 指标测

度并比较候选算法的性能. *ARR* 指标综合考虑算法的运行时间和准确率, 其计算如式(27)(28)所示:

$$ARR_{a_i,a_j}=\frac{Acc_{a_i}\Big/Acc_{a_j}}{1+\alpha\times\lg\left(Rt_{a_i}\Big/Rt_{a_j}\right)},\ 1\leqslant i\neq j\leqslant l, \tag{27}$$

$$ARR_{a_i}=\frac{\sum_{j=1\wedge j\neq i}^{l}ARR_{a_i,a_j}}{l-1},\ 1\leqslant i\neq j\leqslant l. \tag{28}$$

其中 $a_i$ 和 $a_j$ 表示候选算法, $l$ 为候选算法数, $Acc$ 和 $Rt$ 表示算法在数据集上的准确率和运行时间; $\alpha$为可变参数, 表示准确率和运行时间的相对重要程度. 实验中 *ARR* 的$\alpha$值取研究中常用的 0.1, 0.01, 0.001, 并各自记为 $ARR_1$, $ARR_2$, $ARR_3$ 指标, 以获得较为全面的算法性能测度结果.

通过 5 次 10 折交叉验证获取候选算法在各数据集上的性能指标值, 对指标值进行比较从而确定数据集的最优算法, 将最优算法作为标签与元特征结合, 构建相应性能指标的元数据集. 采用 $D_{Acc}$, $D_{Pre}$, $D_{Rec}$, $D_{F1}$ 表示通过准确率、查准率、查全率和 $F_1$ 得分指标构建的元数据集; 使用$D_{ARR_1}$, $D_{ARR_2}$, $D_{ARR_3}$分别表示通过 $ARR_1$, $ARR_2$, $ARR_3$ 指标生成的元数据集. 此外, 当应用基于回归的元算法时, 为各候选算法构建单独的元数据集, 其中的元实例标签为性能指标值, 训练元算法预测各候选算法的性能指标值, 比较预测值从而选出最优算法.

3.1.5 对比方法

采用 *k*NN, SVM, CART, SVR, RF, RFR, XGB, LGBM 作为元算法与 SAMO 进行对比实验. 研究表明 *k*NN 的距离度量采用欧氏距离, 邻居数 *k* 值位于元数据集实例数的 10%～15% 之间时, 其表现更好[9,40]. 经过对比择优, 本文设置 *k*=30, 距离度量采用以距离倒数为权重的加权欧氏距离. 除 *k*NN 外, 其他基分类器和元算法均采用 sklearn 中的默认参数设置.

通过 5 折交叉验证, 即将元数据集随机划为 5 份, 选择其中 4 份作为训练集, 1 份为测试集, 计算各方法的性能指标值.

### 3.2 元数据集分析

对构建的 7 个元数据集进行分析, 候选算法在各元数据集中的胜出次数如表 4 所示.

从表 4 可以看出, 在本文的测试环境中, RF 具有较为优越的分类性能, 但是其在$D_{ARR_i}$元数据集中的胜出次数较少, 表明运行时间是其短板. 与 RF 相对的是 *k*NN 和 NB, 得益于算法较快的运行速度, *k*NN 和 NB 在 *ARR* 指标上具备一定优势, 但其分类性能较差, 因此, 随着 $\alpha$ 值的减小, 运行时间的重要性降低, *k*NN 和 NB 在 *ARR* 指标元数据集中的胜出次数减少. 相较于 *k*NN 和 NB, LR 的分类性能略优但时间开销更高, 在 7 个指标上的表现较为平庸. SVM 的分类性能较优且具有合适的时间开销, 在 7 个指标上均取得了较好结果. LDA 和 ID3 展现了较好的分类性能, 另一方面, LDA 和 ID3 在 *ARR* 指标的元数据集中也取得了较多次数的胜出, 说明其在运行时间和准确率上较为均衡. MLP 的分类性能具有一定优势, 但其在各数据集上的运行时间较长, 因此 MLP 在 *ARR* 指标的元数据集中取得的胜出次数较少. CNN 的分类性能较弱且运行开销较高, 在 7 个指标上的结果较差.

**Table 4 Win Times of the Candidate Algorithms**

**表 4 候选算法胜出次数**

| 候选算法 | 元数据集 | | | | | | |
|---|---|---|---|---|---|---|---|
| | $D_{Acc}$ | $D_{Pre}$ | $D_{Rec}$ | $D_{F1}$ | $D_{ARR_1}$ | $D_{ARR_2}$ | $D_{ARR_3}$ |
| *k*NN | 10 | 15 | 13 | 16 | 45 | 21 | 8 |
| RF | **106** | **100** | **84** | **94** | 10 | **66** | **98** |
| SVM | 37 | 28 | 20 | 21 | 21 | 32 | 32 |
| LR | 27 | 26 | 25 | 22 | 2 | 21 | 26 |
| NB | 11 | 13 | 21 | 13 | 51 | 17 | 14 |
| LDA | 20 | 24 | 26 | 25 | **66** | 40 | 22 |
| ID3 | 21 | 27 | 38 | 38 | 65 | 46 | 34 |
| MLP | 23 | 18 | 23 | 24 | 0 | 14 | 21 |
| CNN | 5 | 9 | 10 | 7 | 0 | 3 | 5 |

注: 黑体数值表示最多胜出次数.

### 3.3 参数敏感性分析

为了确定合适的参数设置, 本节设置种群规模 *N*=50 和最大迭代次数 *T*=300, 对方法的关键参数, 即基分类器个数 *V* 和多样性指标 *DI* 进行敏感性分析.

为了便于说明, 以 $D_{Acc}$ 元数据集为例, $DI\in\{QS, KS, CC, AM, DF\}$, $V\in\{10, 15, 20, 25, 30\}$, 对方法独立运行 20 次的平均帕累托解数量和准确性精英蚁狮的 *ER* 平均值进行对比, 结果分别如图 9 和图 10 所示.

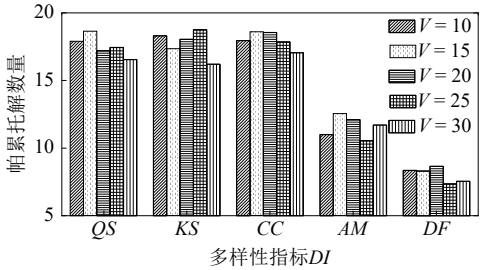


Fig. 9 Pareto solution numbers of different diversity indicators

图 9 不同多样性指标时的帕累托解数量

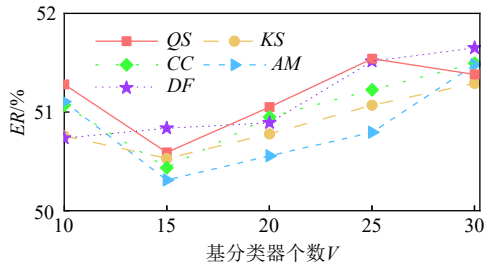


Fig. 10 Error rates of different base classifier numbers

图 10 不同基分类器个数时的错误率

观察图 9 中的帕累托解数量，其在 5 种多样性指标上均呈现相同的变化趋势，即随着基分类器个数 $V$ 的增加，帕累托解数量整体呈现出先增加后减少的趋势，当 $V$=15 时，SAMO 在 5 种 *DI* 上的帕累托解数量相对较多. 另一方面，当使用 *QS*, *KS*, *CC* 指标时，SAMO 产生的帕累托解数量较为接近，优于 *AM* 和 *DF* 指标上的结果.

分析图 10 结果可以发现，随着基分类器个数 $V$ 的增加，SAMO 的 *ER* 性能在 5 种多样性指标上均呈现出先降低后提升的趋势，当基分类器个数 $V$=15 时，5 种指标上的 *ER* 值取得最优结果. 此外，当基分类器个数 $V$ 相同时，5 种指标上的 *ER* 性能较为接近，其中 *AM* 指标上的整体表现优于其他指标.

进一步分析图 9 和图 10，随着 $V$ 的增加，基分类器的多样性得到提升，集成的效果变好；而当 $V$>15 时，基分类器多样性的提升有限，却引入了部分准确性较差的基分类器，同时方法个体的搜索空间扩大，搜索效率降低，导致了方法性能的下降.

综合上述结果，为了获取具有较好算法选择性能的集成元算法，设置基分类器个数 $V$=15 较为合适，多样性指标 *DI* 宜使用 *AM* 指标.

### 3.4 优化算法效果验证

对本文所提多目标混合蚁狮优化算法、多目标蚁狮优化(multi-objective ALO, MALO)[41]、第 2 代非支配排序遗传算法(non-dominated sorting genetic algorithm 2, NSGA2)[42]、速度约束多目标粒子群优化(speed-constrained multi-objective particle swarm optimization, SMPSO)[43] 以及第 2 代强度帕累托进化算法(strength Pareto evolutionary algorithm 2, SPEA2)[44] 的优化性能进行比较，其中 NSGA2, SMPSO, SPEA2 基于框架 jMetalPy[45] 实现. 各对比算法均采用连续型编码，设置阈值为 0.5 对个体的元特征选择编码进行离散化. 具体而言，设 $A_i$ 为个体的第 $i$ 维元特征选择编码，当 $A_i \geqslant 0.5$ 时表示第 $i$ 个元特征被选择，$A_i<0.5$ 时表示未被选择；在个体的选择性集成编码部分设置与本文算法相同的编码和解码方式.

设置基分类器个数 $V$=15，多样性指标 *DI* 为 *AM*，算法种群规模 $N$=50，最大迭代次数 $T$=300，取算法独立运行 20 次的平均结果. 对算法准确性最优个体的 *ER* 值、多样性最优个体的 *DI* 值和帕累托解数量进行比较，结果如表 5~7 所示.

**Table 5 Error Rate Results of the Algorithms**

**表 5 各算法错误率结果** %

| 元数据集 | 本文算法 | MALO | NSGA2 | SMPSO | SPEA2 |
|---|---|---|---|---|---|
| $D_{Acc}$ | **50.3** | 52.4 | 54.0 | 54.4 | 53.8 |
| $D_{Pre}$ | **54.1** | 56.7 | 57.1 | 57.7 | 57.2 |
| $D_{Rec}$ | **56.3** | 58.2 | 60.1 | 61.2 | 59.8 |
| $D_{F1}$ | **51.7** | 54.4 | 55.5 | 56.3 | 55.8 |
| $D_{ARR_1}$ | **51.6** | 54.3 | 57.8 | 59.2 | 58.9 |
| $D_{ARR_2}$ | **55.7** | 57.9 | 60.5 | 60.9 | 60.8 |
| $D_{ARR_3}$ | **50.9** | 52.2 | 53.9 | 54.2 | 53.8 |

注：黑体数值表示最优结果.

**Table 6 Diversity Indicator Results of the Algorithms**

**表 6 各算法多样性指标结果**

| 元数据集 | 本文算法 | MALO | NSGA2 | SMPSO | SPEA2 |
|---|---|---|---|---|---|
| $D_{Acc}$ | **0.561** | 0.590 | 0.654 | 0.64 | 0.651 |
| $D_{Pre}$ | **0.592** | 0.628 | 0.668 | 0.665 | 0.662 |
| $D_{Rec}$ | **0.560** | 0.601 | 0.636 | 0.631 | 0.639 |
| $D_{F1}$ | **0.549** | 0.582 | 0.634 | 0.627 | 0.637 |
| $D_{ARR_1}$ | **0.501** | 0.521 | 0.587 | 0.584 | 0.579 |
| $D_{ARR_2}$ | **0.568** | 0.601 | 0.646 | 0.646 | 0.657 |
| $D_{ARR_3}$ | **0.561** | 0.598 | 0.647 | 0.642 | 0.638 |

注：黑体数值表示最优结果.

**Table 7 Pareto Solution Number Results of the Algorithms**

**表 7 各算法帕累托解数量结果**

| 元数据集 | 本文算法 | MALO | NSGA2 | SMPSO | SPEA2 |
|---|---|---|---|---|---|
| $D_{Acc}$ | **12.6** | 5.9 | 6.2 | 7.2 | 6.6 |
| $D_{Pre}$ | **9.3** | 5.5 | 5.8 | 7.2 | 5.8 |
| $D_{Rec}$ | **10.4** | 5.4 | 6.1 | 5.9 | 6.2 |
| $D_{F1}$ | **12.2** | 5.6 | 6.1 | 5.9 | 5.7 |
| $D_{ARR_1}$ | **9.0** | 5.2 | 4.9 | 5.1 | 4.8 |
| $D_{ARR_2}$ | **10.2** | 5.7 | 5.5 | 6.0 | 5.1 |
| $D_{ARR_3}$ | **11.0** | 6.0 | 6.6 | 6.8 | 6.3 |

注：黑体数值表示最优结果.

从表 5 和表 6 可以看出，本文算法的准确性最优个体和多样性最优个体分别可以取得最低的 *ER* 值和 *DI* 值，说明其寻优性能优于其他对比算法. 对比 MALO,

NSGA2, SMPSO, SPEA2 的结果, 不难发现 MALO 的性能优于另外 3 种算法.

表 7 中, 本文算法可以产生较多的帕累托解数量, 而其他算法的帕累托解数量较之有明显的差距, 进一步验证了本文算法具有较强的寻优性能.

使用非支配比率(non-dominance ratio, *NR*)[46]、超体积(hyper volume, *HV*)[47] 和空间指标(spacing, *SP*)[48], 对各算法的解集质量进行评估. *NR* 将不同算法产生的多个解集合并为 1 个解集并从中筛选出帕累托解构成解集 $S_m$, 然后计算某一个解集中的解在 $S_m$ 中所占的比例, 比值越大说明该解集的质量较之其他解集的质量越优. *HV* 计算解集与目标空间中的最差点所构成空间的面积, *ER* 和 *DI* 最差的指标值均为 1, 因此最差点为(1,1), *HV* 值越大表示解集质量越好. *SP* 计算解集中相距最近的 2 个解距离的标准差, 其值越小代表解的分布性越好. 各算法在 *NR*, *HV*, *SP* 上的比较结果如表 8~10 所示.

**Table 8 *NR* Results of the Algorithms**

**表 8 各算法 *NR* 结果**

| 元数据集 | 本文算法 | MALO | NSGA2 | SMPSO | SPEA2 |
|---|---|---|---|---|---|
| $D_{Acc}$ | **0.743** | 0.252 | 0 | 0.010 | 0 |
| $D_{Pre}$ | **0.834** | 0.166 | 0 | 0 | 0 |
| $D_{Rec}$ | **0.713** | 0.297 | 0 | 0 | 0 |
| $D_{F1}$ | **0.778** | 0.225 | 0 | 0 | 0 |
| $D_{ARR_1}$ | **0.711** | 0.289 | 0 | 0 | 0 |
| $D_{ARR_2}$ | **0.764** | 0.229 | 0.010 | 0 | 0 |
| $D_{ARR_3}$ | **0.757** | 0.229 | 0.010 | 0 | 0 |

注: 黑体数值表示最优结果.

**Table 9 *HV* Results of the Algorithms**

**表 9 各算法 *HV* 结果**

| 元数据集 | 本文算法 | MALO | NSGA2 | SMPSO | SPEA2 |
|---|---|---|---|---|---|
| $D_{Acc}$ | **0.209** | 0.190 | 0.157 | 0.160 | 0.158 |
| $D_{Pre}$ | **0.181** | 0.158 | 0.140 | 0.139 | 0.142 |
| $D_{Rec}$ | **0.184** | 0.164 | 0.141 | 0.139 | 0.141 |
| $D_{F1}$ | **0.208** | 0.186 | 0.159 | 0.158 | 0.157 |
| $D_{ARR_1}$ | **0.235** | 0.215 | 0.171 | 0.167 | 0.170 |
| $D_{ARR_2}$ | **0.184** | 0.164 | 0.137 | 0.135 | 0.132 |
| $D_{ARR_3}$ | **0.206** | 0.186 | 0.159 | 0.160 | 0.162 |

注: 黑体数值表示最优结果.

对比表 8 中的 *NR* 结果, 相较于其他算法, 本文算法有明显的优势, 展现了更强的收敛性. MALO 可以搜索到本文算法所未能发现的帕累托解, 具备一定的寻优性能. NSGA2, SMPSO, SPEA2 在 *NR* 上的表现较差.

**Table 10 *SP* Results of the Algorithms**

**表 10 各算法 *SP* 结果**

| 元数据集 | 本文算法 | MALO | NSGA2 | SMPSO | SPEA2 |
|---|---|---|---|---|---|
| $D_{Acc}$ | **0.013** | 0.019 | 0.017 | 0.024 | 0.021 |
| $D_{Pre}$ | 0.018 | **0.014** | 0.020 | 0.016 | 0.018 |
| $D_{Rec}$ | 0.012 | **0.011** | 0.019 | 0.018 | 0.014 |
| $D_{F1}$ | **0.013** | 0.021 | 0.015 | 0.016 | 0.022 |
| $D_{ARR_1}$ | **0.015** | 0.016 | 0.017 | 0.017 | 0.022 |
| $D_{ARR_2}$ | 0.013 | **0.011** | 0.018 | 0.014 | 0.018 |
| $D_{ARR_3}$ | **0.016** | 0.023 | 0.019 | 0.017 | 0.018 |

注: 黑体数值表示最优结果.

分析表 9, 所提算法在各元数据集上取得了最大的 *HV* 值, 表明算法产生的解在目标空间中距离最差点较远, 解集的质量高于其他算法. MALO 在个别元数据集上可以取得与本文算法相近的 *HV* 值, 其性能优于 NSGA2, SMPSO, SPEA2.

表 10 的结果显示 5 种算法在不同元数据集上具有较优的 *SP* 值, 其中本文算法和 MALO 分别在部分元数据集上都取得了最优结果, NSGA2, SMPSO, SPEA2 的结果较为接近. 结合帕累托解数量结果, 本文算法产生较多的帕累托解, 同时可以保持较好的分布性.

综合上述分析, 在对算法选择模型进行优化时, 相较于其他 4 种算法, 本文算法可以在 2 个优化目标上搜索到更优解, 并且可以发现数量更多且质量更高的帕累托解, 在收敛性和分布性上有较大的优势, 说明本文算法在寻优性能上具备优越性.

### 3.5 SAMO 实验结果

采用与 3.4 节相同的方法参数设置, 取 SAMO 运行 20 次所得的准确性精英蚁狮的性能均值进行对比, 各方法的 *ER* 比较结果如表 11 所示.

**Table 11 *ER* Results of the Methods**

**表 11 各方法 *ER* 结果** %

| 方法 | 元数据集 | | | | | | | 排名 |
|---|---|---|---|---|---|---|---|---|
| | $D_{Acc}$ | $D_{Pre}$ | $D_{Rec}$ | $D_{F1}$ | $D_{ARR_1}$ | $D_{ARR_2}$ | $D_{ARR_3}$ | |
| SAMO | **50.3** | 54.1 | **56.3** | **51.7** | **51.6** | **55.7** | **50.9** | **1** |
| *k*NN | 59.2 | 66.5 | 68.5 | 63.8 | 74.6 | 73.5 | 59.6 | 7 |
| SVM | 59.6 | 61.9 | 66.9 | 63.1 | 78.1 | 75.4 | 63.1 | 8 |
| CART | 69.6 | 71.9 | 76.2 | 69.6 | 64.6 | 73.1 | 69.2 | 9 |
| SVR | 59.2 | 61.5 | 67.3 | 63.8 | 75.0 | 75.0 | 62.7 | 6 |
| RF | 55.8 | 55.4 | 58.8 | 54.2 | 54.2 | 57.7 | 54.6 | 2 |
| RFR | 60.0 | 62.3 | 63.8 | 60.8 | 66.5 | 71.2 | 66.9 | 5 |
| XGB | 59.6 | 57.7 | 56.9 | 54.2 | 54.2 | 58.1 | 56.5 | 4 |
| LGBM | 57.7 | **53.8** | 60 | 55.8 | 53.5 | 58.8 | 56.5 | 3 |

注: 黑体数值表示最优结果.

对表 11 进行分析，可以看出 SAMO 在各元数据集上有更优越的性能，其在除 $D_{Pre}$ 外的其他元数据集上均取得了最好结果. *k*NN, SVM, CART 的性能较弱，SAMO 的性能明显优于这 3 种元算法. SVR 预测各候选算法的性能指标值，比较预测值并选择最优算法，计算开销较高，且其在 *ER* 指标上不具备性能优势. 在集成方法中，除 SAMO 外，RF 具有较好的性能，在各元数据集上均有较好的表现；RFR 结合集成和回归方法，预测算法的性能指标值，其性能优于 SVR 但明显劣于其他集成方法；XGB 在 $D_{Rec}$ 元数据集上的性能强于 RF，但是方法的整体性能不如 RF；LGBM 的整体表现并不突出，但其在 $D_{Pre}$ 元数据集上取得了最优结果. 将 SAMO 与平均性能排名第 2 的 RF 进行比较，SAMO 有着 2.9% 的平均性能领先，证明了 SAMO 在 *ER* 指标上的优越性.

进一步使用查准率、查全率和 *F*1 得分指标比较方法性能，结果如表 12~14 所示.

分析表 12 查准率结果，SAMO 在 5 个元数据集上均获得了最好性能. *k*NN 和 CART 在查准率指标上的表现明显优于 SVM 和 SVR. 在除 SAMO 外的集成方法中，RFR 的性能不如其他方法. RF, XGB, LGBM 的性能较优，其中 RF 在 $D_{Pre}$ 元数据集上表现突出，XGB 在 $D_{Rec}$ 元数据集上具有优越性能. SAMO 相较于 *k*NN, SVM, CART, SVR 有较大程度的性能优势，其与 RF 相比平均性能提升了 2.5%，表明 SAMO 在查准率指标上具有优越性.

表 13 中，SAMO 在 *ARR* 指标的元数据集上取得了较好的查全率结果. CART 的查全率性能优于 *k*NN 和 SVM，SVR 的性能表现稍劣于 SVM. RFR 的性能优于 CART，在 $D_{Acc}$ 上具有优异表现，但是在其他元数据集上的表现一般. RF, XGB, LGBM 的查全率性能较好，其中 XGB 在 $D_{Rec}$, $D_{F1}$, $D_{ARR_2}$ 元数据集上以及 LGBM 在 $D_{Pre}$ 元数据集上表现优异. SAMO 的性能相较于 RF 平均领先了 2.5%，相较于 XGB 平均领先了 0.5%，说明 SAMO 在查全率指标上具备一定的有效性.

从表 14 可以看出，SAMO 在各元数据集上的 *F*1 得分性能较好，在部分元数据集上取得了最好的结果. 在 *k*NN, SVM, CART 中，CART 的性能优于 *k*NN 和 SVM. SVR 与 SVM 的性能较为接近，与其他方法的差距较大. RFR 的性能稍劣于 CART，表现较为一般. RF, XGB, LGBM 具有较好的 *F*1 得分性能，其中 XGB 和 LGBM 分别在不同元数据集上表现突出. SAMO 的 *F*1 得分优于 *k*NN, SVM, CART，另一方面，其相较于 RF 性能平均提升了 2.4%，与 XGB 相比则性能平均提

**Table 12 Precision Results of the Methods**

**表 12 各方法查准率结果** %

| 方法 | 元数据集 | | | | | | | 排名 |
|---|---|---|---|---|---|---|---|---|
| | $D_{Acc}$ | $D_{Pre}$ | $D_{Rec}$ | $D_{F1}$ | $D_{ARR_1}$ | $D_{ARR_2}$ | $D_{ARR_3}$ | |
| SAMO | **35.7** | 35.9 | 38.5 | **37.7** | **38.6** | **32.8** | **32.8** | **1** |
| *k*NN | 16.1 | 14.2 | 18.6 | 21.3 | 17.4 | 16.7 | 18.3 | 7 |
| SVM | 5.0 | 4.4 | 7.0 | 7.5 | 5.1 | 2.9 | 4.4 | 8 |
| CART | 19.9 | 18.0 | 17.3 | 19.7 | 30.1 | 21.4 | 22.3 | 6 |
| SVR | 5.0 | 4.4 | 4.7 | 5.3 | 7.3 | 3.0 | 4.4 | 9 |
| RF | 27.1 | **42.4** | 35.9 | 36.1 | 34.8 | 28.8 | 29.5 | 2 |
| RFR | 27.9 | 21.0 | 25.7 | 23.6 | 22.9 | 15.9 | 12.7 | 5 |
| XGB | 26.7 | 26.6 | **40.6** | 36.3 | 36.2 | 30.2 | 31.1 | 4 |
| LGBM | 27.7 | 37.3 | 36.9 | 37.3 | 38.1 | 27.0 | 29.1 | 3 |

注：黑体数值表示最优结果.

**Table 13 Recall Results of the Methods**

**表 13 各方法查全率结果** %

| 方法 | 元数据集 | | | | | | | 排名 |
|---|---|---|---|---|---|---|---|---|
| | $D_{Acc}$ | $D_{Pre}$ | $D_{Rec}$ | $D_{F1}$ | $D_{ARR_1}$ | $D_{ARR_2}$ | $D_{ARR_3}$ | |
| SAMO | 28.0 | 29.7 | 30.9 | 31.6 | **38.9** | **31.2** | **27.4** | **1** |
| *k*NN | 18.8 | 16.1 | 17.8 | 18.9 | 17.7 | 17.0 | 20.8 | 7 |
| SVM | 11.9 | 11.3 | 11.9 | 12.1 | 15.8 | 11.3 | 11.4 | 8 |
| CART | 19.9 | 20.6 | 20.5 | 26.1 | 31.2 | 23.1 | 22.2 | 6 |
| SVR | 12.0 | 11.4 | 11.4 | 11.6 | 15.8 | 11.5 | 11.6 | 9 |
| RF | 22.8 | 28.9 | 31.2 | 29.4 | 33.4 | 28.7 | 25.7 | 4 |
| RFR | **29.4** | 25.7 | 25.7 | 26.5 | 24.9 | 17.7 | 13.9 | 5 |
| XGB | 23.7 | 27.4 | **36.3** | **34.0** | 34.7 | **31.2** | 27.0 | 2 |
| LGBM | 24.4 | **31.9** | 31.6 | 30.6 | 34.8 | 29.2 | 25.9 | 3 |

注：黑体数值表示最优结果.

**Table 14 *F*1 Score Results of the Methods**

**表 14 各方法 *F*1 得分结果** %

| 方法 | 元数据集 | | | | | | | 排名 |
|---|---|---|---|---|---|---|---|---|
| | $D_{Acc}$ | $D_{Pre}$ | $D_{Rec}$ | $D_{F1}$ | $D_{ARR_1}$ | $D_{ARR_2}$ | $D_{ARR_3}$ | |
| SAMO | **28.0** | 28.8 | 29.5 | 30.4 | **36.6** | **29.5** | 26.4 | **1** |
| *k*NN | 16.3 | 13.8 | 15.0 | 16.5 | 16.4 | 15.0 | 18.4 | 7 |
| SVM | 7.0 | 6.2 | 6.7 | 7.2 | 6.1 | 4.6 | 6.3 | 8 |
| CART | 18.7 | 17.8 | 17.2 | 20.7 | 27.9 | 21.1 | 20.6 | 5 |
| SVR | 7.0 | 6.3 | 5.9 | 6.4 | 6.9 | 4.7 | 6.4 | 9 |
| RF | 22.5 | 30 | 29.6 | 27.5 | 32.1 | 26.5 | 24.5 | 4 |
| RFR | 25.0 | 20.9 | 22.7 | 22.1 | 22.0 | 14.8 | 11.5 | 6 |
| XGB | 23.3 | 25.4 | **34.8** | **31.6** | 33.1 | 29.3 | **26.6** | 2 |
| LGBM | 24.4 | **30.7** | 30.8 | 29.9 | 33.6 | 27.0 | 25.4 | 3 |

注：黑体数值表示最优结果.

升了 0.8%, 验证了 SAMO 在 $F1$ 得分指标上的有效性.

将 SAMO 与 RF, RFR, XGB, LGBM 这 4 种集成方法进行对比, SAMO 使用 3 种基分类器进行选择性集成, 而其他方法仅使用 CART 基学习器进行集成, SAMO 生成的集成元算法具有更强的多样性. 另一方面, 在实验测试中, SAMO 构建的集成元算法在 $D_{Acc}$, $D_{Pre}$, $D_{Rec}$, $D_{F1}$, $D_{ARR_1}$, $D_{ARR_2}$, $D_{ARR_3}$ 上平均集成的基分类器数量分别为 7.7, 6.5, 6.7, 7.2, 5.5, 7.1, 7.8, 而其他方法固定使用 100 个基学习器进行集成, 由此可见 SAMO 通过选择性集成有效减少了基学习器的数量, 从而降低集成元算法的存储和运行开销.

综上所述, SAMO 选择具有互补性的元特征子集, 并使用不同类型的基分类器进行选择性集成, 生成具有较好算法选择性能和多样性的集成元算法. 其在测试指标上的性能均明显优于 $k$NN, SVM, CART, SVR 方法, 而与 RF, XGB 等采用同构基学习器的集成方法相比, 也具有相对优越的性能表现, 证明了 SAMO 的有效性和优越性.

## 4 结 论

为了提升基于元学习的算法选择性能, 提出基于多目标混合蚁狮优化的算法选择方法 SAMO. 设计算法选择模型, 引入元特征选择和选择性集成, 同时寻找互补性较强的元特征子集和对多种元算法进行选择性集成; 提出多目标混合蚁狮优化算法求解模型, 使用混合编码机制选择元特征并构建集成元算法, 采用增强游走策略和偏好精英选择机制提高寻优性能. 构建分类算法选择问题进行实验测试, 将 SAMO 与 8 种典型的元算法进行对比, 结果显示 SAMO 具有优异的算法选择性能, 明显优于 $k$NN, SVM, CART, SVR 元算法, 较之 RF, RFR, XGB, LGBM 这 4 种集成元算法也有一定优势, 并且具备更强的多样性, 综合体现了 SAMO 的有效性和优越性.

未来的工作包括 3 个方面:

1)由于算法选择相关因素较多, 问题较为复杂, 因此目前本文所提方法 SAMO 的效果有限, 后续将深入研究数据集、元特征和元算法的内在关系, 对 SAMO 进行改进, 显著提升算法选择性能.

2)本文方法的时间复杂度较高, 未来将通过并行计算、迁移学习等手段降低方法的训练开销.

3)将 SAMO 拓展应用于实际工程, 构建算法选择系统.

**作者贡献声明:** 李庚松提出方法设计, 负责实验方案的实现和论文内容的撰写; 刘艺提出研究问题, 梳理论文结构; 郑奇斌整理论文内容, 负责实验数据集的收集和处理; 李翔参与方法的实验结果分析; 刘坤提出方法思路的改进建议; 秦伟提出论文修改和优化建议; 王强负责实验方案的实现指导; 杨长虹给出完善论文整体框架的建议.

## 参 考 文 献

[1] Adam S P, Alexandropoulos S-A N, Pardalos P M, et al. No free lunch theorem: A review[M]//Approximation and Optimization. Cham, Switzerland: Springer, 2019: 57−82

[2] Kerschke P, Hoos H H, Neumann F, et al. Automated algorithm selection: Survey and perspectives[J]. Evolutionary Computation, 2019, 27(1): 3−45

[3] Brazdil P, Giraud-Carrier C. Metalearning and algorithm selection: Progress, state of the art and introduction to the 2018 special issue[J]. Machine Learning, 2018, 107(1): 1−14

[4] Yang Chengrun, Akimoto Y, Kim D W, et al. OBOE: Collaborative filtering for AutoML model selection[C]//Proc of the 25th ACM SIGKDD Int Conf on Knowledge Discovery & Data Mining. New York: ACM, 2019: 1173−1183

[5] Dias L V, Miranda P B C, Nascimento A C A, et al. ImageDataset2Vec: An image dataset embedding for algorithm selection[J]. Expert Systems with Applications, 2021, 180: 115053

[6] Shahoud S, Winter M, Khalloof H, et al. An extended meta learning approach for automating model selection in big data environments using microservice and container virtualizationz technologies[J]. Internet of Things, 2021, 16: 100432

[7] Aguiar G J, Santana E J, De Carvalho A C P F L, et al. Using meta-learning for multi-target regression[J]. Information Sciences, 2022, 584: 665−684

[8] Arjmand A, Samizadeh R, Dehghani Saryazdi M. Meta-learning in multivariate load demand forecasting with exogenous meta-features[J]. Energy Efficiency, 2020, 13(5): 871−887

[9] Li Li, Wang Yong, Xu Ying, et al. Meta-learning based industrial intelligence of feature nearest algorithm selection framework for classification problems[J]. Journal of Manufacturing Systems, 2022, 62: 767−776

[10] Chalé M, Bastian N D, Weir J. Algorithm selection framework for cyber attack detection[C]//Proc of the 2nd ACM Workshop on Wireless Security and Machine Learning. New York: ACM, 2020: 37−42

[11] Mu Tianyu, Wang Hongzhi, Zheng Shenghe, et al. Assassin: An automatic classification system based on algorithm selection[J]. Proceedings of the VLDB Endowment, 2021, 14(12): 2751−2754

[12] Garcia L P F, Lorena A C, De Souto M C P, et al. Classifier recommendation using data complexity measures[C]//Proc of the 24th Int Conf on Pattern Recognition. Piscataway, NJ: IEEE, 2018: 874−879

[13] Aguiar G J, Mantovani R G, Mastelini S M, et al. A meta-learning approach for selecting image segmentation algorithm[J]. Pattern

Recognition Letters, 2019, 128: 480−487

[14] Aduviri R, Matos D, Villanueva E. Feature selection algorithm recommendation for gene expression data through gradient boosting and neural network metamodels[C]//Proc of the 12th IEEE Int Conf on Bioinformatics and Biomedicine. Los Alamitos, CA: IEEE Computer Society, 2018: 2726−2728

[15] Meng Jun, Zhang Jing, Jiang Dingling, et al. Selective ensemble classification integrated with affinity propagation clustering[J]. Journal of Computer Research and Development, 2018, 55(5): 986−993 (in Chinese)
(孟军, 张晶, 姜丁菱, 等. 结合近邻传播聚类的选择性集成分类方法[J]. 计算机研究与发展, 2018, 55(5): 986−993)

[16] Jan Z, Munos J C, Ali A. A novel method for creating an optimized ensemble classifier by introducing cluster size reduction and diversity[J]. IEEE Transactions on Knowledge and Data Engineering, 2020, 34(7): 3072−3081

[17] Xu Yuhong, Yu Zhiwen, Cao Wenming, et al. Adaptive classifier ensemble method based on spatial perception for high-dimensional data classification[J]. IEEE Transactions on Knowledge and Data Engineering, 2021, 33(7): 2847−2862

[18] Mohammed A M, Onieva E, Woźniak M, et al. An analysis of heuristic metrics for classifier ensemble pruning based on ordered aggregation[J]. Pattern Recognition, 2022, 124: 108493

[19] Hu Yi, Qu Boyang, Liang Jing, et al. A survey on evolutionary ensemble learning algorithm[J]. Chinese Journal of Intelligent Science and Technology, 2021, 3(1): 18−33 (in Chinese)
(胡毅, 瞿博阳, 梁静, 等. 进化集成学习算法综述[J]. 智能科学与技术学报, 2021, 3(1): 18−33)

[20] Liu Yi, Diao Xingchun, Cao Jianjun, et al. High-dimensional data entity resolution based on ensemble classifying[J]. Application Research of Computers, 2018, 35(3): 689−693 (in Chinese)
(刘艺, 刁兴春, 曹建军, 等. 基于集成分类的高维数据实体分辨[J]. 计算机应用研究, 2018, 35(3): 689−693)

[21] Qasem A, Sheikh Abdullah S N H, Sahran S, et al. An improved ensemble pruning for mammogram classification using modified bees algorithm[J]. Neural Computing and Applications, 2022, 34: 10093−10116

[22] Zhu Xuhui, Ni Zhiwei, Ni Liping, et al. Ensemble pruning of ELM via migratory binary glowworm swarm optimization and margin distance minimization[J]. Neural Processing Letters, 2020, 52(3): 2043−2067

[23] Mirjalili S. The ant lion optimizer[J]. Advances in Engineering Software, 2015, 83(C): 80−98

[24] Niu Guoqiang, Li Xiaoyong, Wan Xin, et al. Dynamic optimization of wastewater treatment process based on novel multi-objective ant lion optimization and deep learning algorithm[J]. Journal of Cleaner Production, 2022, 345: 131140

[25] Abualigah L, Shehab M, Alshinwan M, et al. Ant lion optimizer: A comprehensive survey of its variants and applications[J]. Archives of Computational Methods in Engineering, 2021, 28(3): 1397−1416

[26] Liu Yi, Qin Wei, Zhang Jinhui, et al. Multi-objective ant lion optimizer based on time weight[J]. IEICE Transactions on Information and Systems, 2021, E104.D(6): 901−904

[27] Khan I, Zhang Xianchao, Mobashar R, et al. A literature survey and empirical study of meta-learning for classifier selection[J]. IEEE Access, 2020, 8: 10262−10281

[28] Zeng Zilin, Zhang Hongjun, Zhang Rui, et al. Summary of algorithm selection problem based on meta-learning[J]. Control and Decision, 2014, 29(6): 961−968 (in Chinese)
(曾子林, 张宏军, 张睿, 等. 基于元学习思想的算法选择问题综述[J]. 控制与决策, 2014, 29(6): 961−968)

[29] Rivolli A, Garcia L P F, Soares C, et al. Meta-features for meta-learning[J]. Knowledge-Based Systems, 2022, 240: 108101

[30] Lorena A C, Garcia L P F, Lehmann J, et al. How complex is your classification problem: A survey on measuring classification complexity[J]. ACM Computing Surveys, 2019, 52(5): 1−34

[31] Diao Xingchun, Liu Yi, Cao Jianjun, et al. Reviews of multiobjective ant colony optimization[J]. Computer Science, 2017, 44(10): 7−13,25 (in Chinese)
(刁兴春, 刘艺, 曹建军, 等. 多目标蚁群优化研究综述[J]. 计算机科学, 2017, 44(10): 7−13,25)

[32] Dua D, Graff C. UCI machine learning repository[EB/OL]. 2017[2022-03-19].https://archive.ics.uci.edu/ml/index.php

[33] Alcalá-Fdez J, Fernández A, Luengo J, et al. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework[J]. Journal of Multiple-Valued Logic & Soft Computing, 2011, 17: 255−287

[34] Kooperberg C. StatLib: An archive for statistical software, datasets, and information[J]. The American Statistician, 1997, 51(1): 98−98

[35] Vanschoren J, Van Rijn J N, Bischl B, et al. OpenML: Networked science in machine learning[J]. ACM SIGKDD Explorations Newsletter, 2014, 15(2): 49−60

[36] Alcobaça E, Siqueira F, Rivolli A, et al. MFE: Towards reproducible meta-feature extraction[J]. Journal of Machine Learning Research, 2020, 21: 1−5

[37] Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: Machine learning in Python[J]. Journal of Machine Learning Research, 2011, 12: 2825−2830

[38] Chollet F. Keras[EB/OL]. 2015[2022-07-16].https://keras.io

[39] Brazdil P B, Soares C. Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results[J]. Machine Learning, 2003, 50(3): 251−277

[40] Li Hongqi, Xu Qingsong, Zhu Liping, et al. Classification algorithms recommendation based on dataset similarity[J]. Computer Applications and Software, 2016, 33(8): 62−66 (in Chinese)
(李洪奇, 徐青松, 朱丽萍, 等. 基于数据集相似性的分类算法推荐[J]. 计算机应用与软件, 2016, 33(8): 62−66)

[41] Mirjalili S, Jangir P, Saremi S. Multi-objective ant lion optimizer: A multi-objective optimization algorithm for solving engineering problems[J]. Applied Intelligence, 2017, 46(1): 79−95

[42] Deb K, Pratap A, Agarwal S, et al. A fast and elitist multiobjective genetic algorithm: NSGA2[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182−197

[43] Nebro A J, Durillo J J, Garcia-Nieto J, et al. SMPSO: A new PSO-based metaheuristic for multi-objective optimization[C]//Proc of the 2009 IEEE Symp on Computational Intelligence in Milti-Criteria Decision-Making. Piscataway, NJ: IEEE, 2009: 66−73

[44] Zitzler E, Laumanns M, Thiele L. SPEA2: Improving the strength pareto evolutionary algorithm, 103[R]. Zurich: Swiss Federal Institute of Technology, 2001

[45] Benítez-Hidalgo A, Nebro A J, García-Nieto J, et al. jMetalPy: A

Python framework for multi-objective optimization with metaheuristics[J]. Swarm and Evolutionary Computation, 2019, 51: 100598

[46] Goh C-K, Tan K C. A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization[J]. IEEE Transactions on Evolutionary Computation, 2009, 13(1): 103−127

[47] Zitzler E, Thiele L. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach[J]. IEEE Transactions on Evolutionary Computation, 1999, 3(4): 257−271

[48] Schott J R. Fault tolerant design using single and multicriteria genetic algorithm optimization[D]. Cambridge, MA: Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 1995

**Li Gengsong**, born in 1999. Master. His main research interests include algorithm selection and big data.

**李庚松**, 1999 年生. 硕士. 主要研究方向为算法选择和大数据.

**Liu Yi**, born in 1990. PhD, assistant professor. His main research interests include robot operating system, big data technologies, and evolutionary algorithms.

**刘　艺**, 1990 年生. 博士，助理研究员. 主要研究方向为机器人操作系统、大数据技术和演化算法.

**Zheng Qibin**, born in 1990. PhD, assistant professor. His main research interests include data engineering, data mining, and machine learning.

**郑奇斌**, 1990 年生. 博士，助理研究员. 主要研究方向为数据工程、数据挖掘和机器学习.

**Li Xiang**, born in 1988. PhD, assistant professor. His main research interest includes big data.

**李　翔**, 1988 年生. 博士，助理研究员. 主要研究方向为大数据.

**Liu Kun**, born in 1982. PhD, associate professor. His main research interest includes big data.

**刘　坤**, 1982 年生. 博士，副研究员. 主要研究方向为大数据.

**Qin Wei**, born in 1983. Master, assistant professor. His main research interest includes intelligent information system management.

**秦　伟**, 1983 年生. 硕士，助理研究员. 主要研究方向为智能信息系统管理.

**Wang Qiang**, born in 1972. Master, associate professor. His main research interest includes big data.

**王　强**, 1972 年生. 硕士，副研究员. 主要研究方向为大数据.

**Yang Changhong**, born in 1967. Master, senior engineer. His main research interest includes computer software.

**杨长虹**, 1967 年生. 硕士，高级工程师. 主要研究方向为计算机软件.