基于预搜索和模型选择的离线数据驱动进化算法

李二超†，原万吉

(兰州理工大学 电气工程与信息工程学院，兰州 730050)

摘　要: 离线数据驱动进化算法 (DDEAs) 能够从历史数据中建立代理模型指导种群优化, 它克服了传统进化算法难以应用在计算密集型、机理复杂难以建立数学模型等昂贵优化问题的局限性, 引起了广大学者的关注. 然而, 离线 DDEAs 面临两个困难, 首先构建高质量的代理模型需要使用复杂的模型管理策略, 这虽然提高了算法的性能, 但也增加了算法的运行时间; 其次, 径向基函数网络作为一个被广泛应用在离线 DDEAs 中的模型, 少有研究会根据不同的问题来选择合适的超参数. 为此, 首先提出一种预选择策略, 该策略可以通过复杂度低的粗糙模型将种群快速地迭代到最优解附近; 其次, 提出一种基于肯德尔相关系数的模型排序置信度指标, 并利用该指标设计一种选择策略, 该策略能从几种径向基函数网络的超参数中选择出最适合当前问题的超参数. 基于以上两点并结合堆叠泛化的集成方法, 提出基于预搜索和模型选择的离线数据驱动进化算法 (DDEA-PMS). 与 6 个最新的离线 DDEAs 在 5 个基准问题上的实验结果表明, 所提出的 DDEA-PMS 能以较少的时间开销产生具有明显优势的结果.

关键词: 离线数据驱动优化；进化计算；代理模型；肯德尔相关系数；径向基函数网络

中图分类号: TP391　　**文献标志码:** A

DOI: 10.13195/j.kzyjc.2024.1380

引用格式: 李二超, 原万吉. 基于预搜索和模型选择的离线数据驱动进化算法 [J]. 控制与决策, 2025, 40(10): 3029-3041.

Offline data-driven evolutionary algorithm based on pre-search and model selection

LI Er-chao†，*YUAN Wan-ji*

(College of Electrical and Information Engineering，Lanzhou University of Technology，Lanzhou 730050，China)

Abstract: Offline data-driven evolutionary algorithms (DDEAs) can utilize historical data to build surrogate models that guide population optimization. It overcomes the limitations of traditional evolutionary algorithms, when faced with expensive and computationally intensive problems or problems with complex mechanisms that are difficult to model mathematically. Therefore, it has attracted the attention of many scholars. However, DDEAs face two challenges: first, constructing high-quality surrogate models requires sophisticated model management strategies. While these strategies enhance algorithmic performance, they increase computational time. Second, while radial basis function networks (RBFNs) are widely used in DDEAs, selecting appropriate hyperparameters for different problems remains challenging. To address these issues, this paper first proposes a preselection strategy that uses low-complexity coarse models to quickly iterate the population towards the vicinity of the optimal solution. Then, a model ranking confidence indicator based on the Kendall's rank correlation coefficient is introduced, along with a selection strategy designed to identify the most suitable hyperparameters for the current problem from several RBFN hyperparameter configurations. Based on these two components and combined with the ensemble method of stacked generalization, an offline data-driven evolutionary algorithm based on presearch and model selection (DDEA-PMS) is proposed. Experimental results on 5 benchmark problems, compared with six state-of-the-art DDEAs, demonstrate that the proposed DDEA-PMS can achieve significantly better results with extremely low computational overhead.

Keywords: offline data-driven evolutionary algorithms； evolutionary computation； surrogate models； Kendall’s Tau；radial basis-function network

收稿日期: 2024-11-27；**录用日期:** 2025-03-08.
基金项目: 国家自然科学基金项目 (62063019).
责任编委: 侯忠生.
†通信作者. E-mail: lecstarr@163.com.

0 引 言

进化算法 (EAs)[1] 已被广泛用于解决各种复杂的优化问题[2-3]. 传统进化算法进行优化的前提是可以对新生成的种群进行适应度评估 (FEs), 但当面临如采矿过程优化[4]、高炉优化[5]、融镁炉优化[6]、急救系统优化[7] 等现实生活中难以建立数学模型的复杂问题, 需要耗费大量的物理或计算资源的昂贵优化问题 (EOPs)[8-11] 时, 传统进化算法便难以进行. 为此, 学者们提出了基于数据驱动的进化算法 (DDEAs)[12-13] 来解决这类昂贵优化问题. DDEAs 通过有限的历史数据建立代理模型来取代进化算法中的真实适应度评估. 常用的代理模型包括多项式回归 (PR)[14]、人工神经网络 (ANN)[15]、径向基函数网络 (RBFN)[16]、克里金模型 (Kriging)[17]、支持向量回归 (SVR)[18-19] 等.

根据在优化过程中是否能使用真实适应度函数评估部分新个体用于更新代理模型, DDEAs 可以分为在线 DDEAs[20] 和离线 DDEAs[21] 两大类. 因为在线 DDEAs 可以评估部分新个体, 所以在线 DDEAs 更加关注的是如何选择合适的个体进行评估来更新代理模型[22], 以便代理模型能够更加有效地指导 EAs 进行搜索. 与在线 DDEAs 相反, 离线 DDEAs 无法对新个体进行真实的适应度函数评估来更新代理模型, 因此离线 DDEAs 的主要挑战在于如何利用有限的历史数据构建高质量的代理模型[23].

近年来, DDEAs 问题受到越来越多学者的关注, 针对上述离线 DDEAs 的挑战, 许多学者提出了不同的算法. 为了更加充分地利用数据, Wang 等[21] 提出了 DDEA-SE, 一种基于集成学习的算法. 该算法的模型管理策略主要是在优化之前通过对数据进行抽样来构建大量的代理模型, 在优化的过程中选择部分模型用于集成, 该算法在中低维的昂贵优化问题上有着较好的表现. 针对离线 DDEAs 中数据短缺难以建立精确代理模型的问题, Li 等[24] 提出了 BDDEA-LDG, 一种基于局部数据生成的算法. 该算法无需真实的适应度函数评估就可以合成可靠的数据, 缓解了数据短缺问题. 对于离线 DDEAs 中无法通过小数据集训练出高质量回归模型的问题, Huang 等[25] 提出了一种基于对比学习的 DDEAs 算法, 即 CL-DDEA. 该算法参考对比学习的思想, 通过二元分类来确定每两个个体之间的优劣关系, 建立起一个有向图后通过拓扑排序的方法确定当前种群所有个体的优劣排序. CL-DDEA 在数据稀疏的高维问题上有着不错的表现.

现有的离线 DDEAs 为了提高算法的性能大都采用了复杂的模型管理策略, 但这同时增加了算法的时间复杂度, 导致算法的运行时间增加. RBFN 有着良好的泛化能力、较低的计算成本、较快的训练速度, 被广泛应用于 DDEAs 算法中. 但现有离线 DDEAs 算法中的 RBFN 超参数基本上采取统一的经验值, 统一的超参数难以同时适应不同类型的问题. 针对上面两个问题, 本文在堆叠泛化 (Stacking) 框架的基础上提出一种基于预搜索和模型选择的离线 DDEAs 算法 (DDEA-PMS). 本文的主要贡献如下:

1) 提出一种预搜索策略, 该策略主要有两个作用. 首先, 可以快速地将种群迭代到最优解附近, 在减少计算资源浪费的同时, 不易使种群陷入到局部最优解中; 其次, 为基于肯德尔相关系数[26] 的模型选择策略提供评估对象.

2) 提出一种基于肯德尔相关系数的模型排序置信度指标, 该指标可以衡量不同超参数的 RBFN 模型在最优解附近正确预测个体之间优劣关系的能力. 根据该指标设计了一种模型选择策略, 该策略可以根据不同的问题从多个不同超参数的 RBFN 模型中选择出合适的模型用于集成, 提高了代理模型的精确性与泛化能力.

1 相关工作

1.1 堆叠泛化

集成学习在机器学习领域得到了广泛的应用, 该方法通过构建并组合多个基学习器得到强学习器. 相比于单一学习器, 集成学习通常有更高的准确性和鲁棒性[27]. 常见的集成方法有 Bagging (Bootstrap aggregating)[28]、Boosting[29]、Stacking[30]. 其中, Stacking 通过引入多层模型结构, 不同模型之间可以实现互

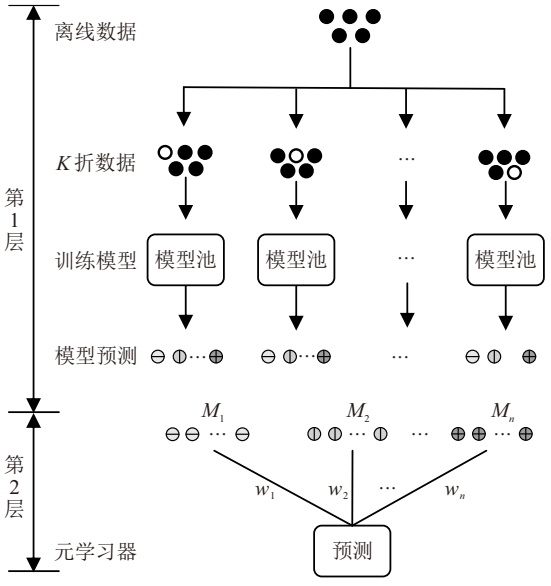


图1 Stacking 框架

补进而减少偏差和方差. Stacking 通常有两层, 第 1 层通过交叉验证法将数据集划分为多个子集, 用于训练多个基学习器并在对应的验证集上进行预测; 在第 2 层中引入元学习器 (Meta-Learner), 并根据基学习器的预测结果进行二次学习, 从而形成最终的预测输出. 在 Stacking 中, 元学习器的主要作用是学习不同基学习器的优点, 提升模型整体性能. Stacking 框架结构如图 1 所示.

1.2　肯德尔相关系数

肯德尔相关系数 (Kendall's Tau) 是一种统计学中用于衡量两个变量之间相关性强弱的非参数统计方法[31]. 相比于皮尔逊相关系数与斯皮尔曼等级相关系数, Kendall's Tau 更适合处理序数数据, 同时在小样本的情况下有着更好的稳健性与可解释性. 定义如下: 设$X=\{x_1,x_2,\ldots,x_N\}$, $Y=\{y_1,y_2,\ldots,y_N\}$是含有N个元素的变量, 分别从序列X、Y中任取下标满足$1\leqslant i<j\leqslant N$的两个元素, 组成两组元素对$(x_i,y_i)$和$(x_j,y_j)$. 如果满足$(x_i-x_j)\cdot(y_i-y_j)>0$, 则称这两个元素对是一致对; 若满足$(x_i-x_j)\cdot(y_i-y_j)<0$, 则称这两个元素对是不一致对; 当出现$x_i=x_j$或$y_i=y_j$时, 这两个元素对既不是一致对也不是不一致对. 当序列不存在这种情况时, Kendall's Tau 计算公式如下所示:

$$\tau=\frac{C-D}{\frac{1}{2}N(N-1)}. \tag{1}$$

其中: C表示X、Y中一致对的个数; D表示X、Y中不一致对的个数.

Kendall's Tau 通过计算成对数据之间的次序一致性来反映变量间的相关性, τ的取值范围位于$[-1,1]$. $\tau>0$表示正相关, $\tau<0$表示负相关. 值越接近 1 或−1, 两个变量之间的关联程度越高; 当$\tau=0$时, 表示两个变量之间是相关独立的.

1.3　研究动机

正如引言中所述, 现有的离线 DDEAs 算法大都采用复杂的模型管理策略来训练, 组合不同的模型来提高代理模型的泛化以及拟合能力, 将这些通过复杂的模型管理策略来构建的代理模型称为精细代理模型 (FM), 然后通过 FM 指导种群从第一代迭代到最后一代来得到问题的最优解. 但在进化算法中, 最初的种群是随机生成的, 此时的种群与最优解之间还有着很大差距, 若从一开始就使用 FM 来指导种群进行迭代, 不但会消耗大量的时间, 还容易使种群陷入到局部最优解中. 其次, 现有离线 DDEAs 算法中 RBFN 模型的超参数基本上都采用统一的经验值, 但采用统一的经验值构建 RBFN 模型难以在不同的问题都取得较好的效果, 特别是 RBFN 隐含层节点的个数. 若隐含层节点过少, RBFN 模型容易欠拟合, 难以捕捉数据中的非线性关系; 相反, 若隐含层节点太多, 则容易导致 RBFN 模型过拟合, 泛化能力变差, 以及提高模型的计算成本.

基于上述考虑, 本文首先提出一个预搜索策略, 该策略通过使用全部的训练数据来构建一个粗糙代理模型 (CM), CM 指导种群迭代到最优解附近后, 将该种群迁移给 FM 继续迭代搜索, 将使用精细代理指导搜索的过程称为精细搜索. 其次, 通过 Kendall's Tau 来构建一种模型选择方法, 该方法可以根据不同的问题从含有不同隐含层节点数的 RBFN 模型中去除掉较差的一些模型, 进而提高代理模型泛化能力以及准确率.

2　算法设计

2.1　DDEA-PMS 算法框架

DDEA-PMS 算法的总体框架如图 2 所示, 包括基础优化器、预搜索和代理模型 3 部分. 基础优化器可以采用遗传算法 (GA)、粒子群算法、差分进化算法等. 预搜索在基础优化器的基础上使用 CM 指导

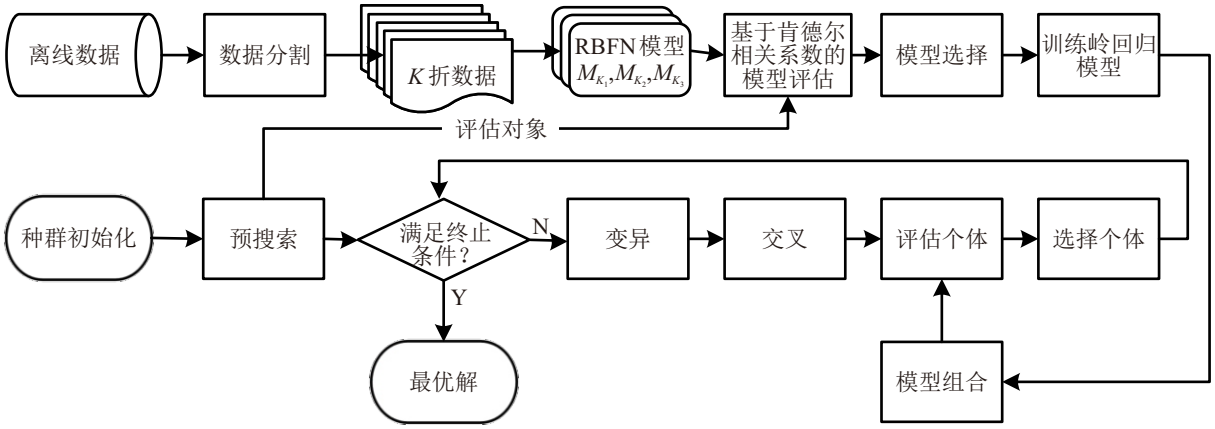


图2　DDEA-PMS 的整体框架

种群迭代一定次数, 使得种群收敛到最优解附近, 精细搜索在预搜索的基础上继续迭代搜索到满足终止条件为止, 与此同时预搜索还要生成待评估对象, 将其用于模型评估和模型选择. FM 采用了 Stacking 集成方法, 第 1 层中有 3 组模型池M_{K_1}、M_{K_2}、M_{K_3}, 每组模型池中分别使用含有K_1、K_2、K_3个隐含层节点数的 RBFN 模型作为基学习器, 并且每组模型池中 RBFN 模型的数量保持一致. 在第 1 层中, 首先通过k折交叉验证的方法将离线数据集分割成k个训练集与测试集, 每个训练集训练 3 个隐含层节点数为K_1、K_2、K_3的 RBFN 模型, 分别放入M_{K_1}、M_{K_2}、M_{K_3}三个模型池中, 每组模型池都有k个模型, 共计得到$3k$个不同的初级模型. 然后通过 Kendall's Tau 来评估 3 组模型的置信程度, 接着通过模型选择策略进行模型选择, 最终剩余的初级模型在测试集上的预测值作为第 2 层的训练数据集. 为了防止过拟合, 第 2 层采用岭回归模型 (RR)[32] 作为元学习器, 用于学习如何有效组合初级模型, 最终得到用于适应度评估的次级模型.

2.2 预搜索策略

预搜索的主要作用是通过复杂度较低的 CM 指导种群快速迭代到最优解附近, 在减少算法运行时间的同时还能避免种群陷入到局部最小值中. 这是由于 CM 不但能平滑掉一些局部最小值[33], 还能帮助种群快速地通过含有局部最小值的区域.

预搜索的具体过程如下:首先使用全部数据训练一个 RBFN 模型作为 CM, 然后使用该模型指导种群迭代ρG次, ρ是$[0,1)$区间内的一个系数, G是优化器总共需要迭代的次数. 根据此时搜索到的最优个体$I_{\rho G\text{-best}}$生成一个随机种群$\mathrm{POP}_{\mathrm{pre}}$, 将该种群作为精细搜索的初始种群. 最后继续使用 CM 指导种群迭代到G次, 根据最终搜索得到最优个体$I_{G\text{-best}}$生成一组用于模型评估的评估对象$\mathrm{Obj}_{\mathrm{eva}}$. 该部分的过程如算法 1 所示.

算法 1 预搜索过程.

输入: 系数ρ, 边界系数k_b, 种群的上下边界UB与LB, 待评估对象的个数n;

输出: 种群$\mathrm{POP}_{\mathrm{pre}}$, 待评估对象$\mathrm{Obj}_{\mathrm{eva}}$.

step 1: 使用全部数据训练一个 RBFN 模型;

step 2: 以UB与LB为上下边界初始化一个种群$\mathrm{POP}_{\mathrm{init}}$;

step 3: 训练一个 RBFN 模型指导种群迭代ρG次, 得到当前种群的最优个体$I_{\rho G\text{-best}}$;

step 4: 继续使用 RBFN 模型指导种群迭代G次, 得到最终的最优个体$I_{G\text{-best}}$;

step 5: 随机生成一个以$k_b\cdot\mathrm{UB}$和$k_b\cdot\mathrm{LB}$为上下边界的种群, 该种群的每个个体加上$I_{\rho G\text{-best}}$后得到种群$\mathrm{POP}_{\mathrm{pre}}$;

step 6: 随机生成n个以$k_b\cdot\mathrm{UB}$和$k_b\cdot\mathrm{LB}$为上下边界的个体, 每个个体加上$I_{G\text{-best}}$后得到待评估对象$\mathrm{Obj}_{\mathrm{eva}}$.

2.3 基于 Kendall's Tau 的模型评估与选择

如文献 [25] 中所述, 在 DDEAs 中更应该关注的是代理模型预测个体之间的支配关系或者排序的准确率, 而不是具体适应度值. 训练代理模型所使用的数据往往与最优解有着较大的差异, 这就导致随着进化算法迭代次数的增加, 种群中的个体逐渐远离已知的数据点, 代理模型预测个体之间正确支配关系的能力逐渐下降. 因此, 判断一个模型优劣的标准之一是要在最优解附近仍能较好地正确判断个体之间的支配关系. 但在离线 DDEAs 中无法使用真实的适应度函数得到个体之间正确的支配关系来用于模型评估, 为此本文设计一个基于 Kendall's Tau 的模型排序置信度 (RC) 指标, 用来评估采用不同超参数训练的 RBFN 模型对最优解附近个体排序的置信程度.

RC 指标的原理参考了文献 [34], 该文献提出可以认为真实的适应度值$f(x)$服从一个以$\tilde{f}(x)$为中心、$\tilde{\sigma}(x)$为方差的正态分布, 其中$\tilde{f}(x)$与$\tilde{\sigma}(x)$分别为克里金模型预测x的值与均方根误差. 即$f(x)$以99.7%的置信度位于$[\tilde{f}(x)-3\tilde{\sigma}(x),\tilde{f}(x)+3\tilde{\sigma}(x)]$区间内. 当模型对两个个体$x_1$、$x_2$预测的值与方差满足$|\tilde{f}(x_1)-\tilde{f}(x_2)|>|3\tilde{\sigma}(x_1)+3\tilde{\sigma}(x_2)|$时, 认为模型预测的$x_1$与$x_2$的支配关系是可靠的, 反之则是不可靠的. 由此可知, 越好的模型, $\tilde{\sigma}(x)$值越小, 模型预测个体之间的支配关系越不容易出错. 虽然 RBFN 模型无法根据已有数据预测未知数据的均方根误差即预测偏差, 但可以使用k折交叉验证或者多次重复采样的方法来训练多个具有相同超参数的 RNFN 模型, 用这些模型来对同一组$\mathrm{Obj}_{\mathrm{eva}}$进行预测后排序, 根据排序之间的相关程度来间接衡量该超参数 RBFN 模型的$\tilde{\sigma}(x)$, 进而从使用不同超参数训练的 RBFN 模型中选择合适的模型.

在进行模型评估之前, 需要先完成模型池的构建. 首先通过k折交叉验证的方法将原始的数据集划分为k个训练集与测试集, 然后依次使用$1-k$折的训练集训练 3 个隐含层的节点数分别为K_1、K_2、K_3的 RBFN 模型, 共计得到$3k$个 RBFN 模型. K_1、K_2、

K_3通过下式得到:

$$K_2 = \lfloor \sqrt{D} \rfloor,\ \text{gap} = \lfloor K_2/3 \rfloor; \tag{2}$$

$$K_1 = K_2 - \text{gap}; \tag{3}$$

$$K_3 = K_2 + \text{gap}. \tag{4}$$

将这些模型分别放到各自对应的模型池中得到$M^1_{K_1},\ldots,M^k_{K_1}$; $M^1_{K_2},\ldots,M^k_{K_2}$; $M^1_{K_3},\ldots,M^k_{K_3}$. 然后, 3 组模型池中的全部模型对$\text{Obj}_{\text{eva}}$进行预测. 为了方便接下来的计算, 需要将预测结果统一转换为排序序列, 根据预测值越小的个体排名越靠前的规则得到Obj_{eva}的排序序列. 将所有的排序序列记为$R^1_{K_1},\ldots,R^k_{K_1}$; $R^1_{K_2},\ldots,R^k_{K_2}$; $R^1_{K_3},\ldots,R^k_{K_3}$.

得到所有排序序列后, 对每组模型池的k个排序序列$R^1_m,\ldots,R^k_m (m = K_1, K_2, K_3)$进行两两组合, 并计算两个排序序列$R^i_m$与$R^j_m (m = K_1, K_2, K_3; i = 1,\ldots,k; j = 1,\ldots,k)$之间的肯德尔相关系数$\tau_{ij}$ $(i = 1,\ldots,k; j = i,\ldots,k)$, 再通过 T 检验计算出 p 值. 当 $p < 0.05$时拒绝原假设, 认为两个排序序列存在显著的相关性, 即两个排序序列是高度相似的. 通过上面的方式, 每组模型池都需要计算$k(k-1)/2$次的$r(\tau_{ij}, p)$, 最后计算每组模型池的 RC 指标.

$$r(\tau_{ij}, p) = \begin{cases} 1, & \tau_{ij} > 0 \text{ and } p < 0.05; \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

$$\text{RC} = \frac{\sum\limits_{i=1}^{k}\sum\limits_{j=i+1}^{k} r(\tau_{ij}, p)}{k(k-1)/2}. \tag{6}$$

接下来, 根据得到的 3 组模型池的 RC 指标进行模型池选择, 若有任意一个模型池的 RC 指标大于 th, 则将 RC 指标最小的模型池中的所有 RBFN 模型去除, 否则保留 3 组模型池中的全部模型. 再根据等级保护指标 (RP)[35] 来删除一定比例模型池中的初级模型, 用于选择出在已知数据点附近预测性能好的模型. 最后使用 RR 模型来组合初级模型, 得到用于预测的次级模型. RP 指标的计算方法如下:

$$\text{RP} = \sum_{i=1}^{n}\sum_{j=i+1}^{n} h(x_i, \boldsymbol{x}_j) \Big/ \binom{n}{2}, \tag{7}$$

$$h(\boldsymbol{x}_i, \boldsymbol{x}_j) = \begin{cases} 1, & (y_i < y_j \text{ and } M(\boldsymbol{x}_i) < M(\boldsymbol{x}_j)) \text{ or} \\ & (y_i > y_j \text{ and } M(\boldsymbol{x}_i) > M(\boldsymbol{x}_j)) \text{ or} \\ & (y_i = y_j \text{ and } M(\boldsymbol{x}_i) = M(\boldsymbol{x}_j)); \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

2.4　时间复杂度分析

DDEA-PMS 的时间复杂度主要包括预搜索与精细搜索两部分, 每部分的计算开销都包括代理模型的建立和基础优化器中的种群迭代[36]. 采用伪逆法训练M个 RBFN 模型的时间复杂度是$O(MN_{\text{data}}C^2)$, N_{data}表示训练数据集的大小, C表示 RBFN 模型隐含层的节点数. 使用 RBFN 模型预测一个体适应度值的时间复杂度是$O(C)$. 若采用 GA 作为基础优化器, GA 的时间复杂度主要来自于每轮使用代理模型对新种群中个体的适应度值进行预测, 其时间复杂度是$O(MCNG)$, N表示种群的大小, G表示迭代次数.

预搜索过程的代理模型只使用了一个 RBFN 模型, 并使用 GA 迭代G次, 故预搜索过程的时间复杂度是$\max(O(N_{\text{data}}C^2), O(CNG))$. 精细搜索过程建立了$3k$个 RBFN 模型, 该部分的时间复杂度是$O(kN_{\text{data}}C^2)$, 基于 Kendall's Tau 的 RC 指标对模型进行评估与选择的时间复杂度是$O(kCN)$, 精细搜索使用 GA 迭代$(1-\rho)G$次, 故精细搜索的时间复杂度是$\max(O(kN_{\text{data}}C^2), O(kCN), O((1-\rho)kCNG))$. 相比精细搜索, 预搜索只使用了一个 RBFN 模型, 其计算开销可以省略, 同时由于在 GA 中G通常要远大于k、N、C, DDEA-PMS 的时间复杂度可以简洁地表达为$O((1-\rho)kCNG)$. 预搜索的策略正是通过在迭代过程中减少使用精细代理模型预测种群个体的次数来减少算法的计算开销.

3　实验设计与分析

3.1　实验设置

为了验证 DDEA-PMS 的性能, 首先采用 6 个最新的离线 DDEAs 在 5 个应用广泛的单峰与多峰基准函数上与 DDEA-PMS 进行比较, 5 个基准函数的具体细节如表 1 所示. 6 个离线 DDEAs 分别为 DDEA-SE[21]、SDK-DDEA[37]、TT-DDEA[38]、DDEA-PES[39]、CL-DDEA[25]、DDEA-PSG[40]. 其次对 DDEA-PMS 所采用的预选择策略与基于 Kendall's Tau 的模型评估与选择策略的有效性进行验证分析. 最后分析 DDEA-PMS 中一些参数的敏感性.

表1　5 个基准函数

基准函数	维度	搜索范围	特点
Ellipsoid	10, 30, 50, 100	[−5.12, 5.12]	单峰
Rosenbrock	10, 30, 50, 100	[−2.048, 2.048]	单峰, 狭长谷形
Ackley	10, 30, 50, 100	[−32.768, 32.768]	多峰
Griewank	10, 30, 50, 100	[−600, 600]	多峰
Rastrigin	10, 30, 50, 100	[−5.12, 5.12]	复杂多峰

为了实验的公平性, 对比实验的设置如下:

1) 所有的训练数据在 GA 开始前通过拉丁超立

方采样 (LHS) 得到, 数据量均为$11D$, 其中D是决策变量的维度. 在 GA 迭代过程中不允许采样生成新的数据.

2) 所有的算法均采用 GA 算法作为基础优化器, 初始种群通过 LHS 得到, 迭代次数G与种群规模都统一设置为 100.

3) 为避免实验过程的偶然性, 每个问题都独立运行 20 次, 并以 20 次实验结果的平均值与方差作为最终结果. 通过威尔科克森符号秩 (Wilcoxon) 检验来分析其他 6 个对比算法与 DDEA-PMS 优劣的显著性, 显著性参数设定为 0.05. 使用“+”“–”“≈”分别表示对比算法优于、劣于、近似于 DDEA-PMS. 对于每一个问题, 性能最好算法的结果通过粗体来显示, 并通过弗里德曼 (Friedman) 检验给出所有算法在所有问题上的平均秩.

测试所使用的环境为 Intel Core i513600KF (5.1 GHz). DDEA-PES 与其变体算法以及对比的 6 个算法 DDEA-SE、TT-DDEA、SDK-DDEA、DDEA-PES、CL-DDEA、DDEA-PSG 均通过 Python 实现. 对比的 6 个 DDEAs 均采用与原文相同的最优参数设置.

3.2 算法设置

在 DDEA-PMS 中, 预搜索部分 RBFN 模型隐含层的节点数设置为$\sqrt{D}$, 其余超参数与精细搜索中 RBFN 模型设置的超参数一致. ρ设置为 0.7, 即预搜索迭代 70 次. k_b设置为 0.1, 即随机种群的上下边界为$0.1\,\mathrm{UB}$与$0.1\,\mathrm{LB}$. 交叉验证的折数设置为 30. 3 种 RBFN 模型均使用高斯核函数作为隐含层的激活函数, 核的扩展常数设置为中心点之间的最大距离, 隐含层中心点通过k-means 聚类算法得到, 隐含层节点数通过式 (2) ～ (4) 确定, 隐含层与输出层之间的权重参数通过伪逆矩阵法计算得到. 在模型评估与选择部分, th 设置为 0.6, 基于 RP 指标的模型保留率设置为 0.5.

基础优化器 GA 使用模拟二进制交叉 ($\eta = 15$) 和多项式变异 ($\eta = 15$), 交叉概率设置为 1, 变异概率为$1/D$, D是决策变量的维度.

3.3 对比实验结果与分析

在这部分, DDEA-PMS 与 6 个最新的 DDEAs 进行比较, 这 6 个 DDEAs 概述如下:

DDEA-SE[21]: 使用了 bagging 的集成方法, 在使用 GA 之前通过对原始数据集进行多次重采样来得到大量的数据集, 然后使用这些数据集构建多个 RBFN 模型. 在优化过程中使用全部的模型来评估上一代的最优解, 根据评估结果自适应地选择其中表现较好的模型用于集成, 以达到最佳的局部逼近精度与降低计算复杂度的效果.

SDK-DDEA[37]: 在运行 GA 之前构建多个具有不同核函数的 RBFN 模型. 在 GA 运行中, 使用这些 RBFN 模型预测新种群中的个体适应度值后, 使用随机排序的模型管理方法对种群进行排序, 根据排序结果选择下一代种群.

TT-DDEA[38]: 为了克服 DDEAs 中数据不足的问题, 将半监督学习引入到离线数据驱动的进化优化过程中, 该文提出可以将原始的离线数据视为有标签数据, 将 GA 中产生的新个体视为无标签数据. 通过 tri-training 的方法选择模型预测的具有高置信度的个体, 将该个体赋予伪标签并加入到训练集中. 该方法缓解了 DDEAs 中数据短缺的问题, 丰富代理模型的训练数据, 提高代理模型对当前种群评估的准确性.

DDEA-PES[39]: 为了缓解 DDEAs 中可用数据不足问题, 提出可以对已有的数据进行数据扰动来生成新的数据, 同时提出一种多样化的代理模型生成方法. 该方法重复在模型预测误差大的数据处进行数据扰动生成新的数据, 并利用这些数据来训练新模型. 最后选择一些对原始数据中最优个体预测误差小的模型用于集成最终的预测模型.

CL-DDEA[25]: 提出在离线 DDEAs 的背景下, 通过小数据集训练出高质量的回归模型非常具有挑战性, 回归模型在 DDEAs 中的作用并不是预测个体的精确适应度, 而是判断个体之间的支配关系. 基于上述考虑, 作者探索了另一种实现 DDEAs 的方法, 首先训练一个基于孪生神经网络的对比学习模型, 以预测每个个体之间的支配关系, 然后利用个体之间成对的支配关系构建有向图, 最终通过拓扑排序来得到当前种群个体的排名.

DDEA-PSG[40]: 为了提高 DDEAs 的泛化能力, 提出了一种基于 Stacking 的代理模型构建方法. 该方法首先使用多种异构的模型来建立模型池; 然后对模型池里的模型评估后剪枝, 以便在提高集成效率的同时进一步提升代理模型的准确率; 最后使用回归模型学习异构模型之间的加权权重, 使得性能较好的模型分配较大的权重, 性能较差的模型分配较小的权重.

上面提到的算法从多个方面来提高离线 DDEAs 的性能, 一方面是采用集成学习的方法, 训练多个模型然后选择性集成一部分模型; 另一方面是通过不

同的方式来提高离线数据量, 进而提高代理模型的质量. 值得注意的是, 这些算法都采用了 RBFN 模型, 且 RBFN 模型的超参数大都是固定的经验值, 与它们相比有助于验证 DDEA-PMS 的有效性.

1) 性能比较.

表 2 显示了比较结果. 可以看出: 在 5 个基准函数, 决策变量为 10、30、50、100 维的共 20 个测试问题上, DDEA-PMS 在 17 个问题上取得了最好的性能表现, 平均排名第 1; 同时在表现不是最优的测试问题上, DDEA-PMS 的表现也是仅次于最优的算法, 这验证了本文提出的算法有着强大的泛化能力与性能表现.

表2　DDEA-PMS 与 6 个 DDEAs 在 5 个测试问题上的结果

问题	维度	DDEA-PMS	DDEA-SE	SRK-DDEA	TT-DDEA	DDEA-PES	CL-DDEA	DDEA-PSG
Ellipsoid	10	0.13 ± 0.04	1.36 ± 0.55(−)	2.11 ± 1.21(−)	1.18 ± 0.79(−)	2.87 ± 1.4(−)	1.02 ± 0.53(−)	**0.00 ± 0.00**(+)
	30	**1.33 ± 0.25**	15.44 ± 4.30(−)	9.99 ± 2.11(−)	5.12 ± 2.31(−)	27.96 ± 6.84(−)	5.05 ± 2.41(−)	2.29 ± 0.23(−)
	50	**2.65 ± 0.38**	46.93 ± 7.50(−)	26.30 ± 5.76(−)	21.72 ± 6.27(−)	69.67 ± 21.19(−)	8.54 ± 1.88(−)	4.52 ± 1.08(−)
	100	**28.50 ± 4.70**	354.08 ± 59.75(−)	255.76 ± 39.91(−)	258.26 ± 44.13(−)	393.13 ± 58.579(−)	48.243 ± 10.73(−)	40.70 ± 6.86(−)
Rosenbrock	10	**14.83 ± 0.98**	42.20 ± 9.69(−)	102.63 ± 36.91(−)	34.48 ± 8.02(−)	56.09 ± 10.53(−)	35.82 ± 9.60(−)	32.82 ± 1.53(−)
	30	**35.88 ± 1.04**	131.61 ± 12.08(−)	190.86 ± 43.06(−)	55.40 ± 7.26(−)	169.83 ± 21.62(−)	63.49 ± 9.63(−)	80.49 ± 2.99(−)
	50	**57.65 ± 1.12**	227.24 ± 14.83(−)	266.11 ± 54.86(−)	86.53 ± 10.13(−)	285.06 ± 21.61(−)	90.87 ± 8.54(−)	118.24 ± 3.62(−)
	100	**115.05 ± 2.28**	519.51 ± 27.50(−)	784.53 ± 142.42(−)	211.41 ± 19.31(−)	605.38 ± 33.00(−)	151.33 ± 7.23(−)	140.50 ± 4.60(−)
Ackley	10	**2.95 ± 0.35**	7.18 ± 1.12(−)	8.12 ± 1.25(−)	6.42 ± 1.25(−)	8.62 ± 1.73(−)	6.22 ± 1.28(−)	5.38 ± 0.49(−)
	30	**3.02 ± 0.11**	7.77 ± 0.73(−)	6.44 ± 0.92(−)	4.95 ± 0.55(−)	8.85 ± 0.86(−)	4.81 ± 0.44(−)	3.38 ± 0.21(−)
	50	**3.20 ± 0.17**	7.78 ± 0.57(−)	6.15 ± 0.60(−)	5.26 ± 0.35(−)	8.93 ± 0.46(−)	4.37 ± 0.29(−)	3.20 ± 0.19(≈)
	100	**3.56 ± 0.12**	9.17 ± 0.48(−)	7.26 ± 0.31(−)	6.91 ± 0.33(−)	9.74 ± 0.57(−)	4.49 ± 0.21(−)	3.92 ± 0.25(−)
Griewank	10	1.00 ± 0.00	1.02 ± 0.03(−)	0.85 ± 0.19(+)	1.21 ± 0.26(−)	1.05 ± 0.03(−)	1.16 ± 0.08(−)	**0.02 ± 0.01**(+)
	30	**1.05 ± 0.01**	1.05 ± 0.01(≈)	1.17 ± 0.06(−)	1.35 ± 0.09(−)	1.07 ± 0.02(−)	1.12 ± 0.06(−)	1.22 ± 0.02(−)
	50	**1.24 ± 0.05**	1.37 ± 0.09(−)	2.46 ± 0.38(−)	2.54 ± 0.35(−)	1.33 ± 0.06(−)	1.27 ± 0.05(≈)	1.45 ± 0.06(−)
	100	2.80 ± 0.19	11.97 ± 3.29(−)	16.64 ± 2.47(−)	16.69 ± 3.52(−)	9.08 ± 1.98(−)	**2.66 ± 0.25**(≈)	3.68 ± 0.46(−)
Rastrigin	10	**15.56 ± 2.99**	74.46 ± 21.00(−)	73.99 ± 29.21(−)	79.23 ± 29.35(−)	93.52 ± 32.98(−)	61.03 ± 13.08(−)	42.39 ± 3.43(−)
	30	**25.28 ± 9.66**	267.24 ± 40.78(−)	301.21 ± 49.01(−)	298.39 ± 59.45(−)	287.86 ± 37.68(−)	113.98 ± 43.95(−)	41.68 ± 5.23(−)
	50	**39.73 ± 11.47**	439.18 ± 36.65(−)	521.92 ± 51.14(−)	498.04 ± 90.57(−)	471.89 ± 57.08(−)	135.13 ± 27.86(−)	54.93 ± 7.96(−)
	100	**129.58 ± 18.01**	972.63 ± 60.12(−)	1 039.21 ± 67.93(−)	963.81 ± 149.33(−)	988.77 ± 82.71(−)	251.04 ± 29.01(−)	174.27 ± 28.52(−)
+/−/≈		NA	0/1/19	1/0/19	0/0/20	0/0/20	0/2/18	2/1/17
Average Rank		1.25	4.8	5.7	4.7	5.9	3.15	2.5

对于对比算法, DDEA-PSG 平均排名第 2, 并在 Ellipsoid 与 Griewank 的 10 维问题上取得了最好的性能表现, 这是由于 DDEA-PSG 使用了 PR 模型, 该模型能在低维上很好地拟合只含有一个极值凸函数. 同时 DDEA-PSG 可以根据不同异构模型的性能表现给模型分配权重, PR 模型在这类问题上赋予了很大的权重, 所以 DDEA-PSG 在 10 维的 Ellipsoid 问题上取得了与真实适应度函数相似的效果. CL-DDEA 平均排名第 3, 可以看出 CL-DDEA 在低维问题上表现并不突出, 但随着决策变量维度的增加, CL-DDEA 在不同问题上的表现都有所改善. 这是由于 CL-DDEA 的训练数据是通过对于原始的离线数据两两配对得到的, 当决策变量维度增加时, 成对的训练数据成指数级增长, 训练出来的对比模型也更加准确. TT-DDEA 平均排名第 4, 在低维单峰问题 Ellipsoid 与 Rosenbrock 上有着不错的表现. 这是由于 TT-DDEA 采用的伪标签技术每代只生成一个伪标签, 在高维问题上一个伪标签并不能起到太大的作用. 对于剩余的 3 个算法, DDEA-SE 采用 bagging 集成学习的方法, 模型的训练数据是通过对原始离线数据以 0.5 的概率重采样得到的, 使用这些训练数据训练得到的模型虽然有丰富的多样性, 但缺少了准确性. 集成学习的关键在于平衡模型的多样性与准确性, 所以表现一般. SRK-DDEA 使用了 4 种核函数的 RBFN 模型与随机排序选择模型管理方法, 对种群个体进行每轮排序时, 4 种模型被选中的概率是一致的, 可能会出现坏的模型排序结果取代好的模型排序结果的情况, 导致该算法表现较差. DDEA-PES 试图通过数据扰动的方法生成新的数据, 进而提高模型的性能, 但该方法具有局限性, 生成的数据位于真实样本附近, 这会导致模型拟合的损失函数曲面变得平坦, 与真实的损失函数曲面不同, 进而错误地指导种群的搜索方向, 最终导致算法差的表现.

为了直观地展示 DDEA-PSG 与对比离线 DDEAs 在优化过程中的收敛情况, 图 3 展示了这些算法在 Ellipsoid、Rosenbrock、Rastrigin 三个基准问题上运

行 20 次的平均适应度值随代数变化的收敛轨迹. 从整体上看, 在前期 CL-DDEA 与 DDEA-PSG 的收敛速度明显高于其他算法, 其次是 DDEA-PMS. 在后期, DDEA-SE 和 DDEA-PES 在 3 个基准问题上都出现了每代最优解震荡的情况, 这表明这两个算法的代理模型已经失去了判别个体之间支配关系的能力. 同时可以看出 DDEA-PMS 在 70 代适应度值出现了跳变, 这是预搜索转变为精细搜索的缘故. 在 Rastrigin 这个复杂多峰问题上 DDEA-SE、SDK-DDEA、TT-DDEA、DDEA-PES 都出现了过早收敛的情况, 可以看出这些算法均收敛在含有多个局部最优解的区域附近, 而 CL-DDEA、DDEA-PSG、DDEA-PMS 则几乎没有这个问题. 总体而言, DDEA-PMS 相比于其他算法有着更棒的性能表现.

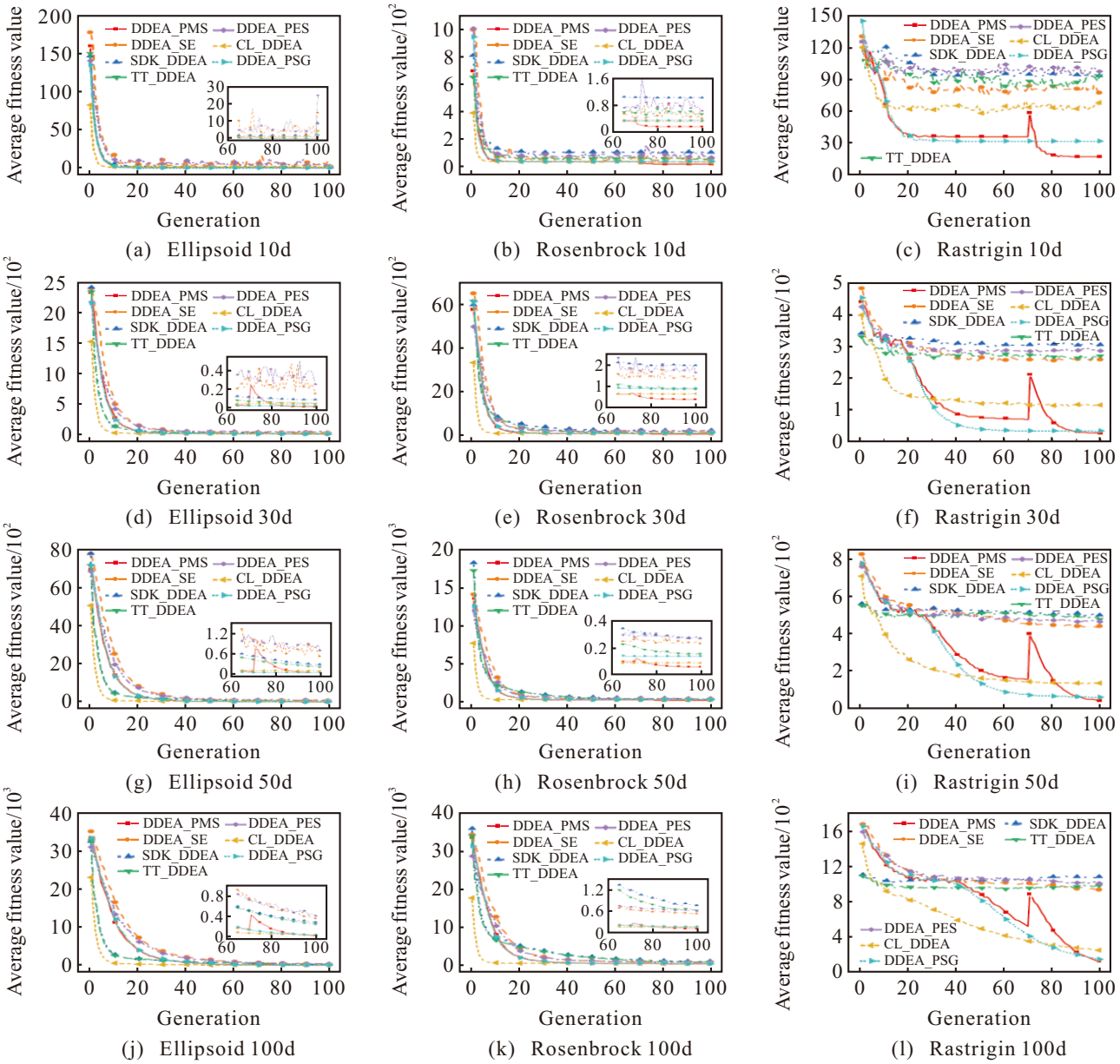


图3 DDEA-PMS 与 6 个 DDEAs 在 3 个测试问题上的收敛曲线

2) 运行时间比较.

算法的运行时间的比较结果如表 3 所示, 图 4 则更加直观地展示了不同算法在不同问题上的平均时间开销. 可以看出, DDEA-PMS 的执行速度大幅领先其他算法, 排名第 1. 这是由于 DDEA-PMS 仅在 GA 开始前训练模型, 并使用预搜索与模型选择策略来减少优化过程中的计算开支. DDEA-SE 排名靠后, 是因为其在迭代中需要重新训练与评估大量的模型. DDEA-PES 也是由于相同的原因导致其平均时间开销大幅较大. CL-DDEA 除了训练对比模型需要消耗较多时间, 每次迭代都需要对种群中两两个体使用对比模型进行评估, 之后还需要进行拓扑排序得到种群的最终排序结果, 所以 CL-DDEA 在所有问题上都有着较大的时间开销. 综合考虑优化精度与运行效率, DDEA-PMS 在这些对比算法中脱颖而出.

表3　DDEA-PMS 与 6 个 DDEAs 在 5 个测试问题上的运行时间

问题	维度	DDEA-PMS	DDEA-SE	SRK-DDEA	TT-DDEA	DDEA-PES	CL-DDEA	DDEA-PSG
Ellipsoid	10	**1.31**	123.50(−)	52.87(−)	9.57(−)	13.50(−)	149.86(−)	11.01(−)
	30	**1.47**	131.66(−)	57.41(−)	15.86(−)	28.62(−)	157.07(−)	26.86(−)
	50	**1.67**	303.21(−)	60.75(−)	21.58(−)	59.62(−)	166.78(−)	40.93(−)
	100	**2.48**	562.20(−)	68.00(−)	37.79(−)	311.24(−)	211.65(−)	87.07(−)
Rosenbrock	10	**1.33**	122.87(−)	51.95(−)	8.61(−)	14.99(−)	150.37(−)	13.70(−)
	30	**1.54**	161.06(−)	56.11(−)	14.65(−)	24.80(−)	156.01(−)	33.04(−)
	50	**1.78**	334.42(−)	59.26(−)	20.36(−)	51.96(−)	166.08(−)	40.98(−)
	100	**2.773**	607.10(−)	66.50(−)	35.33(−)	342.83(−)	211.66(−)	88.36(−)
Ackley	10	**1.29**	123.19(−)	52.05(−)	9.24(−)	12.87(−)	150.17(−)	12.65(−)
	30	**1.50**	161.54(−)	56.07(−)	15.42(−)	27.76(−)	156.13(−)	21.34(−)
	50	**1.72**	313.60(−)	59.41(−)	21.78(−)	58.30(−)	166.50(−)	31.89(−)
	100	**2.48**	548.78(−)	66.76(−)	39.60(−)	292.14(−)	212.56(−)	68.19(−)
Griewank	10	**1.26**	125.20(−)	50.34(−)	9.89(−)	13.04(−)	152.71(−)	12.53(−)
	30	**1.46**	157.80(−)	53.33(−)	16.52(−)	29.29(−)	158.69(−)	28.85(−)
	50	**1.68**	334.99(−)	55.77(−)	22.95(−)	58.62(−)	170.57(−)	48.26(−)
	100	**2.57**	603.53(−)	61.60(−)	38.02(−)	381.37(−)	214.84(−)	96.08(−)
Rastrigin	10	**1.33**	135.30(−)	51.99(−)	8.72(−)	12.87(−)	153.15(−)	12.56(−)
	30	**1.49**	151.48(−)	55.50(−)	15.71(−)	35.71(−)	164.43(−)	25.94(−)
	50	**1.66**	306.11(−)	58.71(−)	21.62(−)	61.35(−)	168.33(−)	43.83(−)
	100	**2.49**	511.91(−)	65.64(−)	310.34(−)	310.34(−)	214.46(−)	88.23(−)
+/−/≈		NA	0/0/20	0/0/20	0/0/20	0/0/20	0/0/20	0/0/20
Average Rank		1	6.6	4.35	2.18	4.53	6.1	3.25

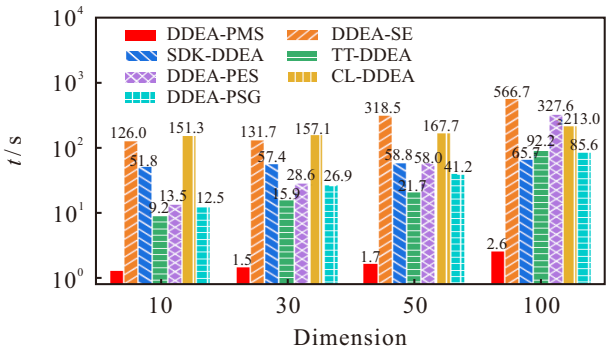


图4　离线 DDEAs 之间的运行时间比较

3.4　策略有效性分析

本节进一步对 DDEA-PMS 所采用预搜索策略与基于 Kendall's Tau 的模型评估与选择策略的有效性进行了分析. 具体而言, 对 DDEA-PMS 与其他 5 个变体算法 DDEA-np/k1/nm、DDEA-np/k2/nm、DDEA-np/k3/nm、DDEA-np/nm、DDEA-p/nm 进行比较. 其中: 前 4 个变体算法都不采用预选择策略以及模型评估与选择策略; 前 3 个变体算法的 3 组 RBFN 模型池使用的 RBFN 隐含层节点数都为同一个值, 分别为K_1、K_2、K_3; 第 4 个变体算法的 3 组 RBFN 模型池的隐含层节点数分别为K_1、K_2、K_3; 第 5 个变体算法使用预选择策略, 但不使用模型评估与选择策略, 其 RBFN 模型的隐含层节点数设置与第 4 个变体算法一致.

表 4 与表 5 分别为 DDEA-PMS 和上述 5 个变体算法在 5 个基准问题的 10、30、50、100 维的比较结果. 可以看出, DDEA-np/k1/nm、DDEA-np/k2/nm、DDEA-np/k3/nm 这 3 个算法在除了 Griewank 的其余 4 个问题上均是 DDEA-np/k1/nm 的效果好, 这表明在 Ellipsoid、Rosenbrock、Ackley、Griewank、Rastrigin 这 4 个问题上, RBFN 模型适合的隐含层节点数为K_1, 在 Griewank 问题上, RBFN 模型适合的隐含层节点数为K_3. DDEA-p/nm 相比于 DDEA-np/nm 使用了预搜索策略, 对比这两个变体的结果可以发现预搜索策略会导致算法在部分问题上找到的最优解变差, 其余的问题几乎没有影响, 并且使用预搜索策略的 DDEA-p/nm 比不使用预搜索策略的 DDEA-np/nm 平均时间消耗降低了 41.4%. DDEA-p/nm 相比于 DDEA-PMS 没有使用模型选择预评估策略, DDEA-p/nm 在 15 个问题上显著弱于 DEA-PMS, 在 5 个问题上与 DEA-PMS 相似, 这表明本文的模型选择预评估策略确实发挥出应有的作用, 可以将差的模型筛选出去. 同时, 本文也从另一个方面验证了提出的模型选择预评估策略的有效性. 统计了 20 次运行中, 基于模型选择与评估策略剔除隐含层节点数分别为K_1、K_2、K_3的 RBFN 模型的数量, 同时根据表 5 的结果, 将去除最差的模型记为一次成功去除,

计算模型选择与评估策略在每个问题上剔除的准确率, 结果如表 6 所示. 模型选择预评估策略剔除最差模型的整体准确率为 75.27%, 这表明设计的基于 Kendall's Tau 的模型评估与选择策略确实是有效的.

表4 DDEA-PMS 与 5 个变体算法的比较结果

维度	问题	DDEA-np/k1/nm	DDEA-np/k2/nm	DDEA-np/k3/nm	DDEA-np/nm	DDEA-p/nm	DDEA-PMS
10	Ellipsoid	**0.170 ± 0.050**(+)	0.382 ± 0.054(−)	0.551 ± 0.036(−)	0.318 ± 0.043(≈)	0.323 ± 0.05(≈)	0.303 ± 0.077
	Rosenbrock	**10.46 ± 0.236**(+)	15.139 ± 0.487(−)	19.183 ± 0.704(−)	13.622 ± 0.433(−)	13.499 ± 0.445(−)	13.089 ± 0.476
	Ackley	**1.910 ± 0.308**(+)	3.129 ± 0.184(−)	4.806 ± 0.187(−)	3.059 ± 0.099(−)	3.021 ± 0.157(−)	2.886 ± 0.238
	Griewank	1.021 ± 0.004(≈)	1.018 ± 0.002(≈)	**1.018 ± 0.003**(≈)	1.015 ± 0.003(≈)	1.016 ± 0.004(≈)	1.019 ± 0.003
	Rastrigin	**3.836 ± 0.934**(+)	15.532 ± 2.179(−)	28.407 ± 2.213(−)	11.479 ± 1.946(−)	11.583 ± 1.438(−)	10.554 ± 1.305
30	Ellipsoid	**0.582 ± 0.077**(+)	0.927 ± 0.129(≈)	1.31 ± 0.166(−)	0.887 ± 0.117(+)	1.083 ± 0.21(−)	0.96 ± 0.27
	Rosenbrock	**34.329 ± 0.784**(+)	36.790 ± 0.737(−)	39.51 ± 0.629(−)	36.392 ± 0.762(≈)	36.425 ± 0.964(≈)	36.273 ± 1.369
	Ackley	**3.531 ± 0.122**(+)	3.939 ± 0.142(≈)	4.218 ± 0.125(−)	3.867 ± 0.169(≈)	3.896 ± 0.175(≈)	3.84 ± 0.158
	Griewank	1.029 ± 0.005(+)	1.029 ± 0.005(+)	**1.027 ± 0.004**(+)	1.03 ± 0.004(+)	1.066 ± 0.021(≈)	1.061 ± 0.021
	Rastrigin	**20.677 ± 3.133**(+)	30.975 ± 2.697(≈)	44.23 ± 3.821(−)	29.291 ± 3.445(≈)	32.096 ± 2.702(−)	29.231 ± 2.571
50	Ellipsoid	**0.778 ± 0.19**(+)	1.184 ± 0.196(+)	1.738 ± 0.253(+)	0.98 ± 0.202(+)	2.361 ± 0.444(−)	1.523 ± 0.469
	Rosenbrock	**52.663 ± 0.479**(+)	56.506 ± 0.697(≈)	60.534 ± 0.988(−)	54.766 ± 0.827(≈)	56.442 ± 1.008(−)	54.136 ± 0.906
	Ackley	**3.001 ± 0.049**(+)	3.529 ± 0.075(−)	3.9 ± 0.08(−)	3.356 ± 0.128(+)	3.547 ± 0.196(−)	3.414 ± 0.223
	Griewank	1.084 ± 0.018(+)	1.096 ± 0.022(≈)	**1.069 ± 0.022**(+)	1.083 ± 0.026(+)	1.262 ± 0.049(−)	1.152 ± 0.051
	Rastrigin	30.365 ± 4.021(+)	44.53 ± 5.003(+)	63.433 ± 3.963(−)	44.082 ± 3.898(≈)	52.253 ± 8.259(−)	45.619 ± 5.117
100	Ellipsoid	29.702 ± 5.464(≈)	31.832 ± 6.738(−)	32.058 ± 4.587(−)	29.598 ± 5.912(−)	29.534 ± 4.331(−)	**28.036 ± 3.315**
	Rosenbrock	111.077 ± 3.226(+)	117.641 ± 3.179(−)	123.115 ± 3.353(−)	114.344 ± 2.747(−)	115.866 ± 3.039(−)	113.738 ± 2.813
	Ackley	3.575 ± 0.225(≈)	3.712 ± 0.206(−)	3.841 ± 0.178(−)	3.761 ± 0.197(−)	3.642 ± 0.15(−)	**3.571 ± 0.112**
	Griewank	2.753 ± 0.333(≈)	2.702 ± 0.343(+)	**2.695 ± 0.235**	2.724 ± 0.364(−)	2.79 ± 0.259(−)	2.750 ± 0.192
	Rastrigin	133.967 ± 19.03(−)	138.674 ± 20.657(−)	152.655 ± 26.697(−)	130.153 ± 23.537(−)	132.106 ± 13.817(−)	**126.884 ± 16.235**
+/−/≈		15/4/1	4/6/10	3/1/16	5/7/8	0/5/15	NA
Average Rank		1.25	4.23	5.35	3.3	4	2.58

表5 DDEA-PMS 与 5 个变体算法的运行时间

维度	问题	DDEA-np/k1/nm	DDEA-np/k2/nm	DDEA-np/k3/nm	DDEA-np/nm	DDEA-p/nm	DDEA-PMS
10	Ellipsoid	2.074(−)	2.174(−)	2.491(−)	2.165(−)	**1.266**(+)	1.296
	Rosenbrock	2.072(−)	2.471(−)	2.528(−)	2.168(−)	**1.257**(+)	1.36
	Ackley	2.081(−)	2.456(−)	2.254(−)	2.139(−)	**1.253**(+)	1.319
	Griewank	2.073(−)	2.472(−)	2.223(−)	2.141(−)	**1.258**(+)	1.291
	Rastrigin	2.074(−)	2.437(−)	2.208(−)	2.137(−)	**1.273**(+)	1.323
30	Ellipsoid	2.426(−)	2.913(−)	2.965(−)	2.524(−)	**1.483**(+)	1.52
	Rosenbrock	2.428(−)	2.909(−)	3.080(−)	2.542(−)	**1.500**(+)	1.609
	Ackley	2.424(−)	2.935(−)	2.613(−)	2.528(−)	**1.489**(+)	1.579
	Griewank	2.405(−)	2.898(−)	2.607(−)	2.516(−)	1.522(−)	**1.496**
	Rastrigin	2.420(−)	2.832(−)	2.609(−)	2.505(−)	**1.500**(+)	1.560
50	Ellipsoid	2.707(−)	3.385(−)	3.578(−)	2.918(−)	**1.718**(+)	1.72
	Rosenbrock	2.718(−)	3.310(−)	3.628(−)	2.886(−)	**1.757**(+)	1.817
	Ackley	2.712(−)	3.361(−)	3.102(−)	2.908(−)	**1.719**(+)	1.787
	Griewank	2.698(−)	3.279(−)	3.092(−)	2.921(−)	1.781(−)	**1.691**
	Rastrigin	2.715(−)	3.293(−)	3.105(−)	2.909(−)	**1.729**(+)	1.823
100	Ellipsoid	4.072(−)	5.417(−)	5.715(−)	4.527(−)	2.581(−)	**2.529**
	Rosenbrock	4.069(−)	5.336(−)	4.882(−)	4.455(−)	2.721(−)	**2.666**
	Ackley	4.044(−)	5.236(−)	4.852(−)	4.48(−)	2.564(−)	**2.524**
	Griewank	4.053(−)	5.234(−)	4.857(−)	4.568(−)	2.741(−)	**2.451**
	Rastrigin	4.078(−)	5.354(−)	4.861(−)	4.464(−)	2.574(−)	**2.413**
+/−/≈		0/0/20	0/0/20	0/0/20	0/0/20	13/0/7	NA
Average Rank		3	5.65	5.35	4	1.3	1.7

表6　模型评估与选择策略剔除模型统计

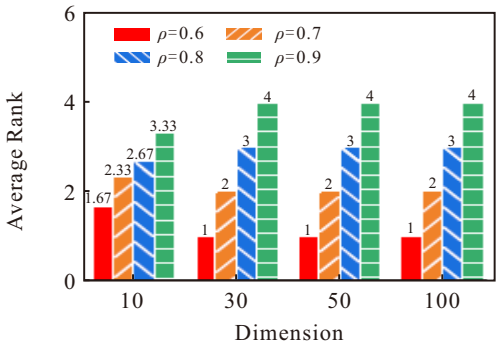
维度	问题	RBF-K_1	RBF-K_2	RBF-K_3	准确率/%
10	Ellipsoid	1	3	6	60.00
	Rosenbrock	0	1	5	83.33
	Ackley	0	5	9	64.28
	Griewank	15	4	0	78.94
	Rastrigin	0	3	5	62.50
30	Ellipsoid	3	2	11	68.75
	Rosenbrock	0	1	4	80.00
	Ackley	0	2	8	80.00
	Griewank	14	4	2	70.00
	Rastrigin	0	5	8	61.54
50	Ellipsoid	2	3	11	68.75
	Rosenbrock	0	2	7	77.78
	Ackley	2	2	9	69.23
	Griewank	20	0	0	100.00
	Rastrigin	1	1	8	80.00
100	Ellipsoid	2	4	13	68.42
	Rosenbrock	0	1	5	83.33
	Ackley	0	5	14	73.68
	Griewank	19	0	1	95.00
	Rastrigin	0	3	12	80.00

3.5　参数敏感度分析

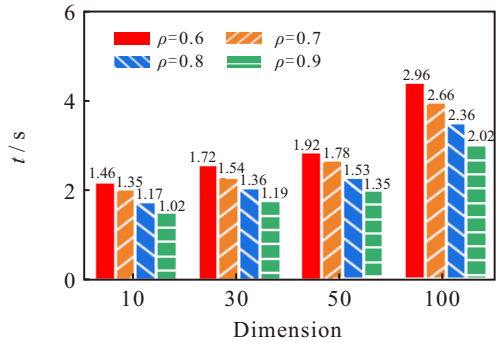
本节对 DDEA-PMS 中的两个关键参数ρ与th进行敏感度分析. 其中: ρ用于控制预搜索的迭代次数, th用于模型选择.

1) 参数ρ敏感度分析. 预搜索迭代次数为ρG, 精细搜索的迭代次数为$(1-\rho)G$. 若ρ设置得过大, 则会导致精细搜索的迭代次数减少, 进而导致优化得到的最优解变差; 反之若ρ设置得过小, 则必然会增加算法的运行时间. 为了探究ρ对所提算法性能的影响, 将ρ设置为 0.6、0.7、0.8、0.9. 图 5 展示了这些参数对 DDEA-PMS优化性能和运行时间的影响, 测试实例为 10、30、50、100 维的 Ellipsoid、Rosenbrock、Rastrigin 问题, 并以 20 次独立运行的平均值为最终结果. 可以看出, ρ越小算法性能越强, 运行时间越长, 为平衡算法的性能与运行时间, 将ρ设置为一个折中的值, 即 0.7.

2) 参数th敏感度分析. 只有当 3 种 RBFN 模型存在任意一个模型的排序置信度指标 RC 大于参数th时, 才除去 RC 指标最小的模型. 若将th设置得较小, 则当出现 3 种模型 RC 指标都较小时的情况时, 无法准确判断出哪个模型优异, 可能将本来优异的模型误删除; 若将th设置得较大, 则可能出现模型评估与选择策略增加时间开销, 却没有提高算法性能的情况. 将th设置为 0.5、0.6、0.7、0.8、0.9 来研究th对所提出算法的影响. 图 6 展示了这些参数对



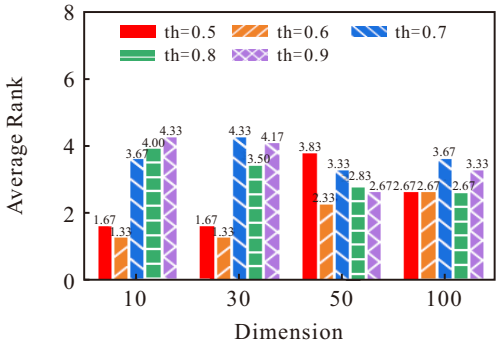
(a)　对优化性能的影响



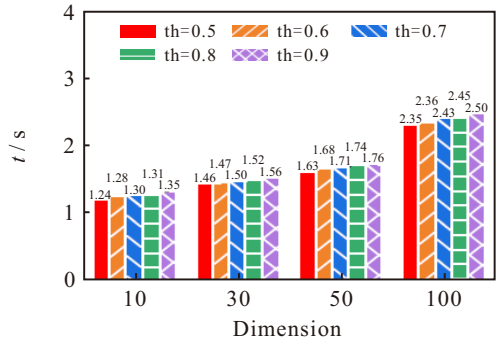
(b)　对优化时间的影响

图5　ρ的参数敏感度分析

DDEA-PMS 优化性能和运行时间的影响, 测试实例为 10、30、50、100 维的 Ellipsoid、Rosenbrock、Rastrigin 问题, 并以 20 次独立运行的平均值为最终结果. 可以看出, th设为 0.6 的算法在 10、30、50 维上排名均为第 1, 在 100 维上效果与th设为 0.5、0.6 的算法相当. 同时也可以发现th设置得越大, 算法的运



(a)　对优化性能的影响



(b)　对优化时间的影响

图6　th的参数敏感度分析

行时间也越长, 这是由于th设置得过大导致在优化前去除相对较差的模型数量较少, 进而增加了在优化中的时间开销, 从而导致算法运行时间增加. 基于上述分析, 最终将th设置为 0.6.

4 结 论

在离线数据驱动优化中为了提高代理模型的性能, 通常使用集成学习以及复杂的模型管理策略, 这些方法虽然提高了算法的性能, 但同样提高了算法的运行时间. 为了缓解这一问题, 本文首先提出了一种预搜索的策略, 该策略先使用一个复杂度较低的CM 快速地将种群迭代到最优解附近, 再将该种群迁移给复杂度高的 FM, FM 在该种群的基础上继续指导种群迭代到满足终止条件为止. RBFN 模型由于其优异的性能被广泛应用在离线数据驱动中, 不同的问题适合的 RBFN 超参数不同, 但目前离线 DDEAs 中均将其设置为统一的经验值. 针对这个问题, 本文提出了一种基于 Kendall's Tau 的 RC 指标, 并基于该指标设计了一种模型评估与选择策略, 该策略可以从给出的 RBFN 超参数中选出适合当前问题的超参数. 基于上述两个策略并结合 Stacking 集成方法, 本文提出了 DDEA-PMS, 并通过多项实验来验证了所提出 DDEA-PMS 的性能, 包括与最新 DDEAs 的比较、消融实验、参数分析. 实验结果表明, DDEA-PMS 以较少的时间开销达到了较高的性能表现.

与此同时, 尽管 DDEA-PMS 有着较高的性能表现, 但也存在这一些问题. 首先预搜索策略并不能根据问题维度的不同自适应地调整迭代次数; 其次, 如何利用 DDEA-PMS 来解决实际生产中各种复杂的问题也是值得关注的重点; 探索更多解决昂贵优化问题的方法也值得深思.

参考文献 (References)

[1] Eiben A E, Smith J. From evolutionary computation to the evolution of things[J]. Nature, 2015, 521(7553): 476-482.

[2] Zhan Z H, Shi L, Tan K C, et al. A survey on evolutionary computation for complex continuous optimization[J]. Artificial Intelligence Review, 2022, 55(1): 59-110.

[3] 肖人彬, 李贵, 陈峙臻. 进化超多目标优化研究进展及展望[J]. 控制与决策, 2023, 38(7): 1761-1788.
(Xiao R B, Li G, Chen Z Z. Research progress and prospect of evolutionary many-objective optimization[J]. Control and Decision, 2023, 38(7): 1761-1788.)

[4] Ding J L, Chai T Y, Wang H, et al. Knowledge-based global operation of mineral processing under uncertainty[J]. IEEE Transactions on Industrial Informatics, 2012, 8(4): 849-859.

[5] Chugh T, Chakraborti N, Sindhya K, et al. A data-driven surrogate-assisted evolutionary algorithm applied to a many-objective blast furnace optimization problem[J]. Materials and Manufacturing Processes, 2017, 32(10): 1172-1178.

[6] Dan G, Chai T Y, Ding J L, et al. Small data driven evolutionary multi-objective optimization of fused magnesium furnaces[C]. 2016 IEEE Symposium Series on Computational Intelligence. Athens, 2016: 1-8.

[7] Wang H D, Jin Y C, Jansen J O. Data-driven surrogate-assisted multiobjective evolutionary optimization of a trauma system[J]. IEEE Transactions on Evolutionary Computation, 2016, 20(6): 939-952.

[8] Zhan D W, Xing H L. Expected improvement for expensive optimization: A review[J]. Journal of Global Optimization, 2020, 78(3): 507-544.

[9] Huang K H, Zhen H X, Gong W Y, et al. Surrogate-assisted evolutionary sampling particle swarm optimization for high-dimensional expensive optimization[J]. Neural Computing and Applications, https://doi.org/10.1007/s00521-023-08661-3.

[10] He C L, Zhang Y, Gong D W, et al. A review of surrogate-assisted evolutionary algorithms for expensive optimization problems[J]. Expert Systems with Applications, 2023, 217: 119495.

[11] Pan J S, Liu N X, Chu S C, et al. An efficient surrogate-assisted hybrid optimization algorithm for expensive optimization problems[J]. Information Sciences, 2021, 561: 304-325.

[12] Jin Y C, Wang H D, Chugh T, et al. Data-driven evolutionary optimization: An overview and case studies[J]. IEEE Transactions on Evolutionary Computation, 2019, 23(3): 442-458.

[13] Zhen H X, Gong W Y, Wang L, et al. Two-stage data-driven evolutionary optimization for high-dimensional expensive problems[J]. IEEE Transactions on Cybernetics, 2023, 53(4): 2368-2379.

[14] Zhou Z Z, Ong Y S, Nguyen M H, et al. A study on polynomial regression and Gaussian process global surrogate model in hierarchical surrogate-assisted evolutionary algorithm[C]. 2005 IEEE Congress on Evolutionary Computation. Edinburgh, 2005: 2832-2839.

[15] Abiodun O I, Jantan A, Omolara A E, et al. State-of-the-art in artificial neural network applications: A survey[J]. Heliyon, 2018, 4(11): e00938.

[16] Regis R G. Evolutionary programming for high-dimensional constrained expensive black-box optimization using radial basis functions[J]. IEEE Transactions on Evolutionary Computation, 2014, 18(3): 326-347.

[17] Liu B, Zhang Q F, Gielen G G E. A Gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems[J]. IEEE Transactions on Evolutionary Computation, 2014, 18(2):

180-192.

[18] Brereton R G, Lloyd G R. Support vector machines for classification and regression[J]. Analyst, 2010, 135(2): 230-267.

[19] 林成龙, 马义中, 肖甜丽, 等. 数据驱动的贝叶斯SVR自适应建模及昂贵约束多目标代理优化[J]. 控制与决策, 2023, 38(10): 2977-2986.
(Lin C L, Ma Y Z, Xiao T L, et al. Data-driven Bayesian SVR adaptive modeling and expensive constrained multi-objective surrogate-based optimization[J]. Control and Decision, 2023, 38(10): 2977-2986.)

[20] Chugh T, Jin Y C, Miettinen K, et al. A surrogate-assisted reference vector guided evolutionary algorithm for computationally expensive many-objective optimization[J]. IEEE Transactions on Evolutionary Computation, 2018, 22(1): 129-142.

[21] Wang H D, Jin Y C, Sun C L, et al. Offline data-driven evolutionary optimization using selective surrogate ensembles[J]. IEEE Transactions on Evolutionary Computation, 2019, 23(2): 203-216.

[22] Yu F Z, Gong W Y, Zhen H X. A data-driven evolutionary algorithm with multi-evolutionary sampling strategy for expensive optimization[J]. Knowledge-Based Systems, 2022, 242: 108436.

[23] Zhang Y, Hu W, Yao W, et al. Offline data-driven multiobjective optimization evolutionary algorithm based on generative adversarial network[J]. IEEE Transactions on Evolutionary Computation, 2024, 28(2): 293-306.

[24] Li J Y, Zhan Z H, Wang C, et al. Boosting data-driven evolutionary algorithm with localized data generation[J]. IEEE Transactions on Evolutionary Computation, 2020, 24(5): 923-937.

[25] Huang H G, Gong Y J. Contrastive learning: An alternative surrogate for offline data-driven evolutionary computation[J]. IEEE Transactions on Evolutionary Computation, 2023, 27(2): 370-384.

[26] Abdi H. The Kendall rank correlation coefficient[J]. Encyclopedia of measurement and statistics, 2007, 2: 508-510.

[27] Garcia-Pedrajas N, Hervas-Martinez C, Ortiz-Boyer D. Cooperative coevolution of artificial neural network ensembles for pattern classification[J]. IEEE Transactions on Evolutionary Computation, 2005, 9(3): 271-302.

[28] Breiman L. Bagging predictors[J]. Machine Learning, 1996, 24(2): 123-140.

[29] Schapire R E. Boosting: Foundations and algorithms[J]. Kybernetes, 2013, 42(1): 164-166.

[30] Naimi A I, Balzer L B. Stacked generalization: An introduction to super learning[J]. European Journal of Epidemiology, 2018, 33(5): 459-464.

[31] Puth M T, Neuhäuser M, Ruxton G D. Effective use of Spearman's and Kendall's correlation coefficients for association between two measured traits[J]. Animal Behaviour, 2015, 102: 77-84.

[32] McDonald G C. Ridge regression[J]. Wiley Interdisciplinary Reviews: Computational Statistics, 2009, 1(1): 93-100.

[33] Yang C E, Ding J L, Jin Y C, et al. Offline data-driven multiobjective optimization: Knowledge transfer between surrogates and generation of final solutions[J]. IEEE Transactions on Evolutionary Computation, 2020, 24(3): 409-423.

[34] Liu Z N, Wang H D, Jin Y C. Performance indicator-based adaptive model selection for offline data-driven multiobjective evolutionary optimization[J]. IEEE Transactions on Cybernetics, 2023, 53(10): 6263-6276.

[35] Díaz-Manríquez A, Toscano G, Coello C C A. Comparison of metamodeling techniques in evolutionary algorithms[J]. Soft Computing, 2017, 21(19): 5647-5663.

[36] Gong Y J, Zhong Y T, Huang H G. Offline data-driven optimization at scale: A cooperative coevolutionary approach[J]. IEEE Transactions on Evolutionary Computation, 2024, 28(6): 1809-1823.

[37] Huang P F, Wang H D, Ma W P. Stochastic ranking for offline data-driven evolutionary optimization using radial basis function networks with multiple kernels[C]. 2019 IEEE Symposium Series on Computational Intelligence. Xiamen, 2019: 2050-2057.

[38] Huang P F, Wang H D, Jin Y C. Offline data-driven evolutionary optimization based on tri-training[J]. Swarm and Evolutionary Computation, 2021, 60: 100800.

[39] Li J Y, Zhan Z H, Wang H, et al. Data-driven evolutionary algorithm with perturbation-based ensemble surrogates[J]. IEEE Transactions on Cybernetics, 2021, 51(8): 3925-3937.

[40] 梁正平, 黄锡均, 李燊钿, 等. 基于剪枝堆栈泛化的离线数据驱动进化优化[J]. 自动化学报, 2023, 49(6): 1306-1325.
(Liang Z P, Huang X J, Li S T, et al. Offline data driven evolutionary optimization based on pruning stacked generalization[J]. Acta Automatica Sinica, 2023, 49(6): 1306-1325.)

作者简介

李二超 (1979−), 男, 教授, 博士, 博士生导师, 主要研究方向为人工智能、进化计算、机器人控制, E-mail: lecstarr@163.com;

原万吉 (2000−), 男, 硕士, 主要研究方向为数据驱动的进化优化, E-mail: yuanwanji2022@163.com.