计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.241100005

基于混合策略的自适应红嘴蓝鹊优化算法

段博文　殷继彬　张　航
昆明理工大学信息工程与自动化学院　昆明 650500
(1968192531@qq.com)

摘　要　针对红嘴蓝鹊优化算法(Red-billed Blue Magpie Optimization Algorithm,RBMO)存在多样性迅速退化、寻优精度差、易陷入局部最优的问题,提出了一种基于混合策略的自适应红嘴蓝鹊优化算法(Adaptive Red-Billed Blue Magpie Optimization Algorithm Based on Mixed Strategy,JRBMO)。首先,引入 Hammersley 序列初始化种群,使初始解分布更均匀,为寻优提供基础;其次,在勘探阶段,提出自适应螺旋围捕策略,通过动态控制个体的勘探范围与方向,提高 RBMO 的搜索能力。在开发阶段,引入莱维飞行策略,对当前最优解进行局部扰动,增强算法局部开发能力;最后,提出自适应维度变异策略,根据种群适应度分布的变化,对个体进行维度变异,避免算法陷入局部最优。在 CEC2017 与 CEC2019 测试集上对算法性能进行评估,结果显示 JRBMO 均值胜率分别达到 88.9%和 70%,验证了 JRBMO 的有效性。此外,将 JRBMO 应用于拉(压)弹簧设计问题和三维无线传感器网络(WSN)节点覆盖问题上,JRBMO 均取得了最优的结果,其中 WSN 节点均值覆盖率高出原算法 6.3%,体现了 JRBMO 在实际应用中的普适性。

关键词:红嘴蓝鹊优化算法;自适应;Hammersley 序列;螺旋围捕;莱维飞行;维度变异
中图分类号　TP301

Adaptive Red-billed Blue Magpie Optimization Algorithm Based on Mixed Strategy

DUAN Bowen,YIN Jibin and ZHANG Hang
Faculty of Information Engineering and Automation,Kunming University of Science and Technology,Kunming 650500,China

Abstract　Aiming at the problems of rapid degradation of diversity,poor optimization accuracy,and susceptibility to local optima in the Red billed Blue Magpie Optimization Algorithm(RBMO),a hybrid strategy based adaptive Red billed Blue Magpie Optimization Algorithm(JRBMO) is proposed. Firstly,the Hammersley sequence is introduced to initialize the population,making the initial solution distribution more uniform and providing a foundation for optimization. Secondly,during the exploration phase,an adaptive spiral capture strategy is proposed to improve the search capability of RBMO by dynamically controlling the exploration range and direction of individuals. In the exploitation phase,the Levy flight strategy is introduced to locally perturb the current optimal solution and enhance the algorithm's local development capability. Finally,an adaptive dimension mutation strategy is proposed to perform dimension mutation on individuals based on changes in population fitness distribution,avoiding the algorithm from getting stuck in local optima. The algorithm performance was evaluated on the CEC2017 and CEC2019 test sets,and the results showed that JRBMO had average win rates of 88.9% and 70%,respectively,verifying the effectiveness of JRBMO. In addition,applying JRBMO to the tension(compression) spring design problem and the three-dimensional wireless sensor network (WSN) node coverage problem,JRBMO achieves the optimal results,in which the WSN node mean coverage is 6.3% higher than that of the original algorithm,which demonstrates the universality of JRBMO in practical applications.

Keywords　Red billed blue magpie optimization algorithm,Adaptive,Hammersley sequence,Spiral capture,Lévy flight,Dimension mutation

1　引言

近年来,元启发式算法(MAs)因其在解决各种工程领域复杂优化问题中的卓越表现而备受关注。相较于传统的数值方法,MAs 不仅经济成本较低且寻优效率较高,并具备较强的灵活性和跳出局部最优解的能力[1]。目前,MAs 已被广泛应用于众多领域,如特征选择、泊位分配、多目标优化、图像分割、路径规划、无线传感网络等[2-4]。

大多数元启发式算法从自然界汲取灵感,根据灵感来源不同,MAs 算法可以分为 4 类:基于进化的、基于物理的、基于人类行为的和基于群体的[5]。基于进化的算法通过模拟自然界优胜劣汰的进化法则进行建模,这类算法在处理离散及多目标优化问题上,具有较强的鲁棒性,如遗传算法(Genetic Algorithm,GA)[6]、差分进化算法(Differential Evolution,DE)[7];基于物理的优化算法从物理现象或化学反应中汲取灵感,该类算法能在多个局部最优解之间跳跃,并通过模拟真实物理现象,优化算法搜索策略以提升收敛精度,如霜冰优化算法(Rime Optimization Algorithm,RIME)[8];基于人类的优

基金项目:国家自然科学基金(61741206)
This work was supported by the National Natural Science Foundation of China(61741206).
通信作者:殷继彬(41868028@qq.com)

化算法受到人类对不同活动行为的启发，具有很强的学习能力和适应性能力，例如逻辑优化算法（Incomprehensible but Intelligible-in-time logics，IBL）[9]；基于群体的算法受到集体群原理的启发，通过模拟群体的智慧，充分利用群体中个体的协同合作来求解复杂优化问题，具有灵活性、鲁棒性、自组织性等优点，如灰狼优化算法（Grey Wolf Optimizer，GWO）[10]、鲸鱼优化算法（Whale Optimization Algorithm，WOA）[11]、金豺优化算法（Golden Jackal Optimization，GJO）[12]、冠豪猪优化器（Crested Porcupine Optimizer，CPO）[13]。

元启发式算法的核心在于探索和开发阶段，探索的目标是在搜索空间中寻找新的区域，以发现潜在的解决方案；开发则侧重于利用局部信息进一步优化现有的解决方案。然而，过度的探索可能导致算法收敛缓慢，而过度的开发则可能使算法陷入局部最优。因此，平衡探索和开发是元启发式算法的关键目标，以保证收敛速度的同时提升收敛精度。此外，No-Free Lunch（NFL）指出，没有一种算法可以在所有优化问题上都表现出有效性[14]。因此，提出的新算法或增强现有元启发式算法的性能和效果，成为了近年来元启发式算法研究的热门方向。

例如，Wu 等[15]提出的 WOA 的变体，利用一种新型的平滑勘探系统（Smooth Exploration System，SES）改善 WOA 的勘探过程，称为平滑 WOA（Smooth WOA，SWOA），具体而言，在通过无序维数抽样、随机交叉和顺序变异策略增强算法全局探索能力的同时提升算法寻找最优解的能力。Zamani 等[16]提出了一种进化乌鸦搜索算法（Evolutionary Crow Search Algorithm，ECSA），用于优化诊断慢性疾病的神经网络超参数，具体来说，通过引入进化搜索策略、自适应飞行步长来有效地探索和开发问题空间，同时利用交互式记忆机制记录和更新乌鸦的最佳解以保持种群多样性。Wang 等[17]提出了一种基于助手机制的金豺优化算法（Helper Mechanism Based Golden Jackal Optimization，HGJO），引入助手机制缓解其他个体对最优个体的强依赖，使用对立学习策略增加种群的多样性，采用非线性自适应因子平衡开发和探索，并将柯西分布融入到现有的更新策略，避免算法陷入局部最优。

RBMO 是由 Fu 等[18]提出的新元启发式算法，通过对红嘴蓝鹊的搜索、追逐、攻击和食物存储行为的模拟，建立了 RBMO 的数学模型，具备参数量少、操作简单、局部开发强的优点。然而，RBMO 在解决复杂问题时仍然存在一定局限性：多样性迅速退化、寻优精度差以及易陷入局部最优。其主要原因在于：RBMO 以探索—存储—开发—存储的寻优结构进行问题求解；探索和开发阶段均以小组或集群的位置为参照；开发阶段过度依赖最优个体指导。具体来说，当算法首次进行开发、存储阶段后，种群个体向最优解靠近，导致种群分布集中，参照团体均值位置更新方式逐渐失效，之后算法进入下一次迭代时，个体又重复上述寻优结构进行问题寻优，随着迭代推进，这种寻优结构不断加剧个体向局部最优解趋近，最终导致以上问题出现。

为解决以上问题，增强 RBMO 的性能，本文提出一种基于混合策略的自适应红嘴蓝鹊优化算法（JRBMO）。对 RBMO 整体寻优结构和每阶段进行改进，首先，利用 Hammersley 序列初始化种群，使初解更均匀地分布在问题空间，以丰富种群的多样性，为后续寻优提供基础。其次，在探索阶段提出自适应螺旋围捕策略，增添非线性因子，动态增加个体探索范围与方向，使算法的在迭代后期仍具全局搜索能力。在开发阶段引入莱维飞行干扰策略，当莱维飞行步长较小时，帮助个体在最优解附近形成邻域包围圈，提高算法的局部开发能力，反之，步长较大时，使个体远离当前最优解，提升算法探索能力。最后，提出自适应维度变异策略，根据检测每次迭代中种群适应度值分布的变化，得到适应度阈值 m，根据 m 对种群个体进行维度替换，避免算法陷入局部最优陷阱，反之，利用个体差异帮助个体在当前搜索区域进行精细开发，提升寻优精度。为验证 JRBMO 的性能，在 CEC2017 和 CEC2019 测试函数上进行实验，与高被引用算法和变体类算法相对比，验证了 JRBMO 在收敛精度上的有效性，此外将 JRBMO 运用于拉（压）弹簧设计问题和三维无线传感器网络（WSN）节点覆盖问题中，证明了该算法在实际问题中的适用性。

2 红嘴蓝鹊优化算法（RBMO）

RBMO 通过模拟红嘴蓝鹊合作、高效的捕食行为，根据其特点，将捕食过程分为 3 个阶段：寻找食物、攻击猎物和存储食物。为了方便理解 RBMO，本节及之后章节 r 和 r_i 都代表 0-1 的随机数，其中 i 为从 1 开始的正整数。

2.1 寻找食物（探索阶段）

在寻找食物时，红嘴蓝鹊根据环境条件和可用资源采用多样化的狩猎策略，以确保充足的食物供应。通常以小组（2 至 5 只）或集群（超过 10 只）的形式外出寻找食物，并使用各种方式，如跳跃在地面上、行走或搜查树木寻找食物资源。寻找食物的方式受种群平衡系数 ε 控制，具体值为 0，当 $r<\varepsilon$ 时，代表团体食物较为充足，此时红嘴蓝鹊会以小组的方式寻找食物。具体表达式如式（1）所示：

$$X^i(t+1)=X^i(t)+\left(\frac{1}{p}\times\sum_{m=1}^{p}X^m(t)-X^{rs}(t)\right)\times r_1 \quad (1)$$

其中，t 代表迭代次数，$X^i(t)$ 表示当前个体的位置，$X^{rs}(t)$ 表示当前迭代中随机个体的位置，$X^i(t+1)$ 代表下一次迭代时当前个体的位置，p 代表小组的红嘴蓝鹊数量，其取值为[2，5]的随机整数，$X^m(t)$ 代表小组中的个体。当 $r>\varepsilon$ 时，表示群体食物匮乏，为了快速地寻找食物，它们会以集群的方式外出，如式（2）所示：

$$X^i(t+1)=X^i(t)+\left(\frac{1}{q}\times\sum_{m=1}^{q}X^m(t)-X^{rs}(t)\right)\times r_1 \quad (2)$$

其中，q 代表集群的红嘴蓝鹊的数量，其取值为[10，N]的整数，N 代表整个种群中个体的数量。

2.2 攻击猎物（开发阶段）

红嘴蓝鹊在追捕猎物时，根据猎物大小不同，采取不同的出行方式：当猎物为小型猎物时，红嘴蓝鹊会参照小组个体的均值，并由最优个体引导进行攻击猎物。具体表达式如下：

$$X^i(t+1)=X^{\text{food}}(t)+CF\times\left(\frac{1}{p}\times\sum_{m=1}^{p}X^m(t)-X^i(t)\right)\times r_2 \quad (3)$$

其中，$X^{\text{food}}(t)$ 代表当前迭代最优个体。CF 是从 1 非线性递减到 0 因子，在迭代过程中，控制红嘴蓝鹊开发的范围，计算式如式（4）所示：

$$CF=\left(1-\frac{t}{T}\right)^{2\times\frac{t}{T}} \quad (4)$$

其中，T 为最大迭代次数。

当猎物为大型猎物时，红嘴蓝鹊以集群的方式攻击猎物，如式（5）所示：

$$X^i(t+1)=X^{\text{food}}(t)+$$
$$CF\times\left(\frac{1}{q}\times\sum_{m=1}^{q}X^m(t)-X^i(t)\right)\times r_2 \tag{5}$$

2.3 存储食物(历史资源储备)

除了外出觅食和攻击猎物外，红嘴蓝鹊还会将多余的食物储存在树洞或其他隐蔽地点，以备未来使用。这一行为有效地保留了历史资源信息，使个体在食物短缺的情况下能够依赖这些食物储备，进而增加算法找到全局最优解的可能性，此过程如式(6)所示：

$$X^i(t+1)=\begin{cases}X^i(t), & fitness^i_{\text{old}}<fitness^i_{\text{new}}\\ X^i(t+1), & \text{else}\end{cases} \tag{6}$$

其中，$fitness^i_{\text{old}}$ 和 $fitness^i_{\text{new}}$ 分别表示第 i 只红嘴蓝鹊位置更新前后的适应度。

3 混合策略自适应的红嘴蓝鹊优化算法

3.1 Hammersley 序列的初始化

在对复杂高维优化问题进行求解时，若问题空间分布未知，则个体初始位置应在问题空间均匀分布[19]。RBMO 初始解通过随机方法在问题空间生成个体，具有较大的随机性以及明显的不稳定性，尤其是在面对高维空间时更为明显，这种初始化方法易造成初始种群分布均匀、多样性低的问题，进而影响算法的收敛精度与速度，不利于算法寻优。为了获得更均匀的初始种群分布，确保初始解的多样性，本文在初始化阶段使用低差异序列——Hammersley 序列初始化种群。Hammersley 序列是一种确定性的初始化样本方法，能在无穷维度上生成均匀分布的点集，其序列生成步骤如下：

step1 对于任意自然数 n 使用素数 p 进行表示：

$$n=\sum_{k=0}^{m}c_kp^k=c_mp^m+\cdots+c_1p^1+c_0p^0 \tag{7}$$

其中，$c_k\in[0,p-1]$

step2 将式(7)中的系数$\{c_m,\cdots,c_1,c_0\}$反序排列，求其值可表示为：

$$\phi_p(n)=c_0p^{-1}+\cdots+c_{m-1}p^{-m}+c_mp^{-m-1} \tag{8}$$

step3：自然数 n 在 d 维空间中的 Hammersley 序列为：

$$H(n)=\left(\frac{n}{N},\phi_{p^1}(n),\phi_{p^2}(n),\cdots,\phi_{p^{d-1}}(n)\right) \tag{9}$$

其中，N 为样本数，n 取值为$\{0,1,\cdots,N\}$，使用 Hammersley 序列初始化种群时，在其问题空间映射生成 N 个个体，映射规则为：

$$X_{(i,j)}=X^{lb}_{(i,j)}+H(i,j)\times(X^{ub}_{(i,j)}-X^{lb}_{(i,j)}) \tag{10}$$

其中，ub 和 lb 分别代表问题空间上下界，$H(i,j)$代表 Hammersley 序列生成的样本点。

图 1、图 2 分别给出了 Random 序列和 Hammersley 序列初始化方法在二维空间中生成的种群分布图。

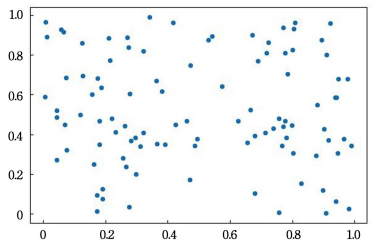


图 1 随机生成的种群分布图

Fig. 1 Population distribution map generated by random

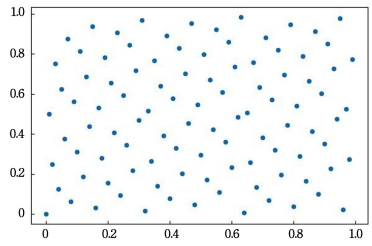


图 2 Hammersley 序列生成的种群分布图

Fig. 2 Population map generated by the Hammersley sequence

对比图 1、图 2 可知，Random 序列初始化种群方法生成的点存在区域集中和稀疏现象，而 Hammersley 序列生成的点分布均匀，且未出现重叠现象，为算法提供了良好的寻优基础。

3.2 自适应螺旋围捕策略

由式(1)、式(2)可知，RBMO 个体探索以部分个体的均值为参照，且探索步长和方向由均值与随机个体的差异决定。这种考虑团体均值和随机个体的搜索方式，虽然能加快算法的收敛速度，但也存在随迭代的推进种群多样性和全局勘探能力迅速下降的问题。为改善此问题，在探索阶段，引入自适应螺旋围捕策略替换 RBMO 的分群体的搜索方式，利用自适应因子动态控制个体的勘探方向与范围，提高算法的全局勘探能力。具体表达式如式(11)所示：

$$X^i(t+1)=X^i(t)+(X^{rz}(t)-X^i(t))\times\frac{\sin 2\theta}{2e^{2\alpha}} \tag{11}$$

其中，$X^{rz}(t)$代表从解空间中随机选择的位置，α 和 θ 分别是非线性的自适应因子和螺旋控制因子，计算式分别如式(12)、式(13)所示：

$$\alpha=r\times\left(\frac{t^2}{T^2}-\frac{2t}{T}+1\right) \tag{12}$$

$$\theta=(2r-1)\times\pi \tag{13}$$

3.3 莱维飞行扰动策略

在开发阶段，由式(3)、式(5)可知，个体以最优个体为中心，在其邻域进行局部开发，这种过度依赖最优个体指导进行局部开发的方式，降低了算法的开发能力。为避免个体对最优个体的过度依赖，本文在开发阶段引入莱维飞行步长增强 RBMO 中个体的开发能力。

莱维飞行是一种随机游走，其步长概率分布具备重尾的特点——大概率小步长，小概率大步长，其步长计算式如式(14)所示：

$$Levy(\beta)=\frac{\mu}{|v|^{(1/\beta)}} \tag{14}$$

其中，β 为莱维参数，μ，v 服从正态分布，$\mu\sim N(0,\sigma_x^2)$，$\mu\sim N(0,\sigma_y^2)$。其中，σ_x，σ_y 的计算式如式(15)所示：

$$\sigma_x=\left(\frac{\Gamma(1+\beta)\times\sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right)\times\beta\times2^{\frac{\beta-1}{2}}}\right)^{\frac{1}{\beta}},\sigma_y=1 \tag{15}$$

将莱维飞行策略融合到式(3)和式(5)中，以在最优解个体附近进行随机扰动。具体而言，当飞行步长为小步长时，这种扰动在最优个体区域附近进行小范围干预，增强个体局部开发能力，反之，当为大步长时，对最优个体的位置进行大步干预，帮助个体跳出当前搜索区域，降低算法陷入局部最优的可能性。莱维飞行干扰的小组更新式如式(16)所示：

$$X^i(t+1)=X^{\text{food}}(t)+CF\times r_3\times \left(\frac{1}{p}\times\sum_{m=1}^{p}X^m(t)-X^i(t)\right)+r*L_{\text{evy}}(\beta) \quad (16)$$

莱维飞行干扰的集群更新式如式(17)所示：

$$X^i(t+1)=X^{\text{food}}(t)+CF\times r_3\times \left(\frac{1}{q}\times\sum_{m=1}^{q}X^m(t)-X^i(t)\right)+r*L_{\text{evy}}(\beta) \quad (17)$$

其中，$\beta=1.5$。

3.4 自适应维度变异策略

RBMO 以探索-存储-开发-存储的寻优结构寻优，这易导致 RBMO 应用于多峰优化问题时易过早收敛到局部最优解，影响算法收敛精度。为进一步提高算法寻优能力，本文提出自适应维度变异策略更改 RBMO 的寻优过程。其数学模型如式(18)所示：

$$X^i(t+1)=\begin{cases}X^i(t)+CF*(lb+r_4*(ub-lb)*E), & r<m\\ X^i(t)+(m*(1-r_4)+r_4)*(X^a(t)-X^b(t)), & \text{else}\end{cases} \quad (18)$$

其中，$X^a(t)$，$X^b(t)$分别代表种群中两个随机的不同个体，E是包含 0 和 1 的二进制向量，计算式如式(19)所示：

$$E=rand(1,D)<0.3 \quad (19)$$

其中，D 代表解空间的维度。m 是根据种群适应度分布得到的阈值，其计算式如下：

$$m=\frac{\sum_{i=1}^{N}sum(f_{\text{avg}}>f_i)}{N} \quad (20)$$

其中，f_{avg}是此次迭代过程中适应度的平均值，f_i 是个体的适应度值，N 是种群数量。当种群个体集中时，此时多数个体适应度值小于 f_{avg}，阈值 m 较大，多数个体进行维度替换，以跳出当前区域，避免算法陷入局部最优陷阱。反之，当种群个体分布较广时，阈值 m 较小，多数个体利用个体差异性，在当前邻域搜索区域进行精细的局部探索，提升算法的局部寻优能力。

3.5 JRBMO 的基本流程

综合以上针对 RBMO 算法的改进策略，JRBMO 实现的基本流程如图 3 所示。

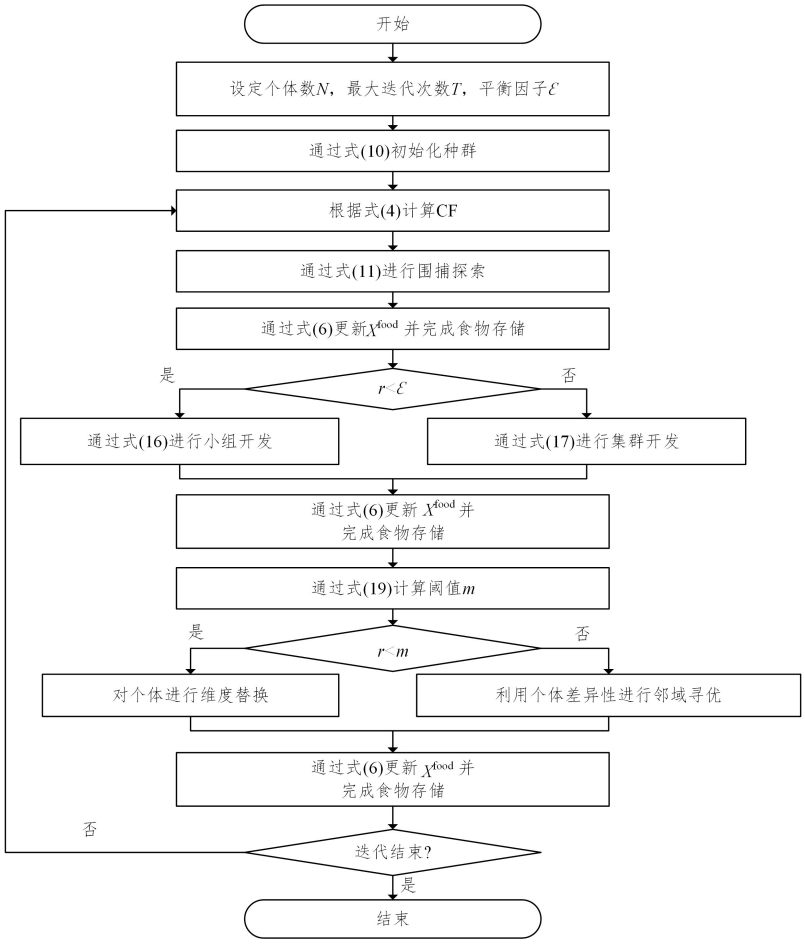


图 3 JRBMO 算法流程

Fig. 3 JRBMO algorithm flows

3.6 JRBMO 时间复杂度分析

由文献[18]可知，RBMO 的总时间复杂度为 $O(T\times N\times D)$。JRBMO 通过 Hammersley 序列初始化种群的时间复杂度为 $O(N\times D)$。使用自适应螺旋围捕策略和莱维飞行干扰策略进行位置更新的总时间复杂度为 $O(T\times N\times D)$，自适应维度变异策略的时间复杂度为 $O(T\times N\times D)$，JRBMO 的总时间复杂度为 $O((N\times D)\times(3T+1))$。综上所述，略去低价项，JRBMO 的总时间复杂度为 $O(T\times N\times D)$，与 RBMO 时间复杂度一致。

4 仿真实验和分析

4.1 对比算法设置

为了证明改进算法 JRBMO 的性能，本文选取了 7 个先进算法与 JRBMO 比较，包括被高频次引用的灰狼优化算法

GWO[10]、哈里斯鹰优化算法（HHO）[20]、金豺优化算法（GJO）[12]、蜣螂优化算法（DBO）[21]，以及最新提出的群智能算法（霜冰优化算法（RIME）[8]、冠豪猪优化算法（CPO）[13]和红嘴蓝鹊优化算法（RBMO）[18]）。这些算法在过去的研究中被广泛验证和应用，具有较强的优化性能，与这些算法进行对比更能体现本文改进算法的有效性。为保证实验的客观性和公平性，实验统一设置种群规模 N 为 30，解空间维度 D 为 100，最大迭代次数为 1 000。每种算法独立运行 30 次，并记录每种测试函数的平均值、标准差。仿真实验的操作系统是 Windows 11，硬件环境为 12th Gen Intel（R）Core（TM）i7-12700H@2.30 GHz，仿真软件为 matlab 2021b。

4.2 测试函数说明

本文采用 CEC2017 测试集来评估算法的性能，该测试集是根据实际问题产生的非线性和噪声复杂度挑战问题而设计的。相比标准测试函数 CEC2017 更加复杂，而与近年的 CEC2022 测试函数集相比，其测试的维度上限更高（CEC2022 最高为 20 维，CEC2017 为 100 维），更能全方位测试算法的性能。CEC2017 共包含 29 个测试函数，其中 F1 和 F3 是单峰函数，全局仅存在一个最优解，用于测试算法的局部开发能力；F4－F10 是多峰函数，由于解空间存在多个局部最优解，这类函数侧重于检验算法的全局探索能力；F11－F20 是混合函数，解空间由不同类型的函数组成，并分布在不同的搜索区域内，用于评估算法的适应能力；F21－F30 是组合函数，求解目标由多个函数以权重形式组合而成，类似于现实中的复杂优化问题，这类函数能够全方位测试算法的综合性能。每种测试函数的理论最优值为 $i*100$，i 为测试函数编号。本实验为了更加准确地测试算法的性能，统一将问题维度设置为 100。

4.3 测试函数结果分析

表 1 列出了各对比算法及改进算法在每种测试函数上的统计结果。从表中可以看出，在单峰函数 F1 和 F3 上，RBMO 取得的平均收敛精度和标准差比其他算法高出 5～8 个数量级，体现了 JRBMO 良好的局部开发能力；在多峰函数 F4 至 F10 上，JRBMO 的平均收敛精度均高于对比算法，仅在 F5 和 F10 上标准差略低于 CPO，这体现了 JRBMO 具有良好的全局探索能力；在混合函数 F11 至 F20 上，JRBMO 表现出色，平均收敛精度和标准差均达到了 60％的优胜率，表明在应对复杂问题时 JRBMO 具备较强的适应能力；在组合函数 F21 至 F30 上，JRBMO 的平均收敛精度高于多数对比算法，表明 JRBMO 在解决实际问题时具备强大的潜力。

表 1　在 CEC2017 测试集上的实验结果（100 维）

Table 1　Experimental results on CEC2017 test suite(100 Dim)

函数		GWO	HHO	GJO	DBO	RIME	CPO	RBMO	JRBMO
F1	AVG	5.53×10^{10}	8.02×10^{9}	1.37×10^{11}	1.46×10^{10}	1.74×10^{8}	7.21×10^{8}	8.24×10^{8}	**7.01×10^{3}**
	STD	9.99×10^{9}	1.93×10^{9}	1.09×10^{10}	5.08×10^{9}	4.29×10^{7}	5.75×10^{8}	8.73×10^{8}	**6.39×10^{3}**
F3	AVG	5.02×10^{5}	3.19×10^{5}	3.43×10^{5}	4.94×10^{5}	6.64×10^{5}	4.08×10^{5}	1.28×10^{5}	**7.73×10^{4}**
	STD	7.17×10^{4}	1.43×10^{4}	4.09×10^{4}	2.00×10^{5}	8.46×10^{4}	3.97×10^{4}	1.87×10^{4}	**1.33×10^{4}**
F4	AVG	5.61×10^{3}	2.26×10^{3}	1.97×10^{4}	1.20×10^{4}	9.85×10^{2}	9.21×10^{2}	1.02×10^{3}	**6.76×10^{2}**
	STD	1.72×10^{3}	3.82×10^{2}	4.55×10^{3}	1.24×10^{4}	8.86×10	1.29×10^{2}	9.09×10	**5.14×10**
F5	AVG	1.20×10^{3}	1.58×10^{3}	1.57×10^{3}	1.52×10^{3}	1.15×10^{3}	1.57×10^{3}	1.06×10^{3}	**8.64×10^{2}**
	STD	5.22×10	6.43×10	9.16×10	2.16×10^{2}	9.56×10	**5.12×10**	5.47×10	5.29×10
F6	AVG	6.45×10^{2}	6.88×10^{2}	6.75×10^{2}	6.80×10^{2}	6.49×10^{2}	6.22×10^{2}	6.22×10^{2}	**6.09×10^{2}**
	STD	4.47	2.96	4.96	1.25×10	6.35	5.42	5.75	**2.90**
F7	AVG	2.18×10^{3}	3.78×10^{3}	2.98×10^{3}	2.86×10^{3}	1.84×10^{3}	2.16×10^{3}	1.73×10^{3}	**1.22×10^{3}**
	STD	1.89×10^{2}	1.26×10^{2}	1.29×10^{2}	2.78×10^{2}	1.75×10^{2}	1.06×10^{2}	1.53×10^{2}	**6.07×10**
F8	AVG	1.53×10^{3}	1.98×10^{3}	1.92×10^{3}	1.93×10^{3}	1.45×10^{3}	1.84×10^{3}	1.36×10^{3}	**1.13×10^{3}**
	STD	1.28×10^{2}	6.09×10	1.07×10^{2}	2.29×10^{2}	9.10×10	5.46×10	6.54×10	**3.70×10**
F9	AVG	4.22×10^{4}	6.33×10^{4}	5.65×10^{4}	6.52×10^{4}	4.01×10^{4}	3.90×10^{4}	1.33×10^{4}	**6.88×10^{3}**
	STD	1.16×10^{4}	5.41×10^{3}	1.24×10^{4}	1.68×10^{4}	1.29×10^{4}	4.87×10^{3}	4.37×10^{3}	**2.58×10^{3}**
F10	AVG	1.88×10^{4}	2.33×10^{4}	2.47×10^{4}	2.69×10^{4}	1.82×10^{4}	2.95×10^{4}	2.12×10^{4}	**1.52×10^{4}**
	STD	4.80×10^{3}	1.54×10^{3}	4.59×10^{3}	4.44×10^{3}	1.42×10^{3}	**6.20×10^{2}**	2.97×10^{3}	1.59×10^{3}
F11	AVG	8.51×10^{4}	9.27×10^{4}	9.52×10^{4}	1.50×10^{5}	1.22×10^{4}	4.70×10^{4}	5.36×10^{3}	**2.28×10^{3}**
	STD	1.52×10^{4}	1.60×10^{4}	2.03×10^{4}	4.88×10^{4}	3.67×10^{3}	9.36×10^{3}	1.00×10^{3}	**2.34×10^{2}**
F12	AVG	1.37×10^{10}	1.53×10^{9}	5.14×10^{10}	3.66×10^{9}	9.37×10^{8}	9.78×10^{7}	1.01×10^{8}	**2.76×10^{7}**
	STD	7.79×10^{9}	5.31×10^{8}	1.21×10^{10}	1.53×10^{9}	4.13×10^{8}	3.96×10^{7}	1.56×10^{8}	**1.18×10^{7}**
F13	AVG	1.35×10^{9}	1.81×10^{7}	8.66×10^{9}	1.46×10^{8}	7.63×10^{5}	**8.97×10^{3}**	1.33×10^{4}	9.08×10^{3}
	STD	1.16×10^{9}	3.96×10^{6}	3.09×10^{9}	1.64×10^{8}	1.88×10^{6}	**2.72×10^{3}**	4.90×10^{3}	4.65×10^{3}
F14	AVG	9.76×10^{6}	7.81×10^{6}	1.90×10^{7}	1.30×10^{7}	5.48×10^{6}	1.74×10^{6}	2.42×10^{5}	**2.79×10^{4}**
	STD	3.76×10^{6}	1.94×10^{6}	7.99×10^{6}	7.58×10^{6}	2.60×10^{6}	7.91×10^{5}	1.72×10^{5}	**2.05×10^{4}**
F15	AVG	2.54×10^{8}	4.03×10^{6}	2.33×10^{9}	1.30×10^{7}	1.47×10^{5}	**2.94×10^{3}**	7.71×10^{3}	4.19×10^{3}
	STD	2.42×10^{8}	1.14×10^{6}	1.59×10^{9}	3.22×10^{7}	5.21×10^{4}	**7.39×10^{2}**	6.97×10^{3}	3.47×10^{3}
F16	AVG	6.71×10^{3}	8.46×10^{3}	9.05×10^{3}	8.90×10^{3}	6.90×10^{3}	8.89×10^{3}	6.42×10^{3}	**5.11×10^{3}**
	STD	7.17×10^{2}	6.88×10^{2}	1.15×10^{3}	1.17×10^{3}	8.55×10^{2}	6.01×10^{2}	7.03×10^{2}	**5.28×10^{2}**
F17	AVG	5.14×10^{3}	6.97×10^{3}	1.60×10^{4}	8.63×10^{3}	6.00×10^{3}	6.21×10^{3}	5.54×10^{3}	**4.66×10^{3}**
	STD	6.51×10^{2}	7.08×10^{2}	2.20×10^{4}	1.12×10^{3}	6.24×10^{2}	4.62×10^{2}	5.07×10^{2}	**4.03×10^{2}**
F18	AVG	7.96×10^{6}	7.50×10^{6}	1.65×10^{7}	2.12×10^{7}	8.96×10^{6}	2.79×10^{6}	3.97×10^{5}	**1.95×10^{5}**
	STD	6.74×10^{6}	2.01×10^{6}	1.43×10^{7}	1.13×10^{7}	3.59×10^{6}	1.39×10^{6}	1.85×10^{5}	**7.96×10^{4}**
F19	AVG	2.14×10^{8}	1.82×10^{7}	2.17×10^{9}	6.98×10^{7}	9.60×10^{6}	**4.32×10^{3}**	1.28×10^{4}	7.17×10^{3}
	STD	2.63×10^{8}	7.55×10^{6}	1.44×10^{9}	4.52×10^{7}	3.67×10^{6}	**2.17×10^{3}**	1.04×10^{4}	6.56×10^{3}
F20	AVG	5.13×10^{3}	6.35×10^{3}	6.30×10^{3}	6.92×10^{3}	5.56×10^{3}	6.73×10^{3}	4.98×10^{3}	**4.82×10^{3}**
	STD	7.58×10^{2}	5.36×10^{2}	9.64×10^{2}	7.04×10^{2}	5.28×10^{2}	3.04×10^{2}	4.67×10^{2}	**3.00×10^{2}**

（续表）

函数		GWO	HHO	GJO	DBO	RIME	CPO	RBMO	JRBMO
F21	AVG	3.06×10^{3}	4.26×10^{3}	3.51×10^{3}	3.98×10^{3}	3.04×10^{3}	3.29×10^{3}	3.00×10^{3}	$\mathbf{2.69\times10^{3}}$
	STD	9.02×10	1.69×10^{2}	1.16×10^{2}	1.41×10^{2}	1.11×10^{2}	$\mathbf{3.94\times10}$	1.04×10^{2}	4.46×10
F22	AVG	2.03×10^{4}	2.59×10^{4}	2.88×10^{4}	2.85×10^{4}	2.01×10^{4}	3.20×10^{4}	2.37×10^{4}	$\mathbf{1.55\times10^{4}}$
	STD	1.57×10^{3}	1.26×10^{3}	4.83×10^{3}	4.58×10^{3}	1.61×10^{3}	$\mathbf{4.99\times10^{2}}$	1.84×10^{3}	1.16×10^{3}
F23	AVG	3.64×10^{3}	5.69×10^{3}	4.42×10^{3}	4.72×10^{3}	3.51×10^{3}	3.83×10^{3}	3.76×10^{3}	$\mathbf{3.18\times10^{3}}$
	STD	$\mathbf{3.88\times10}$	4.02×10^{2}	1.96×10^{2}	2.12×10^{2}	8.59×10	5.18×10	1.60×10^{2}	7.17×10
F24	AVG	4.34×10^{3}	7.75×10^{3}	5.90×10^{3}	6.02×10^{3}	4.25×10^{3}	4.36×10^{3}	4.57×10^{3}	$\mathbf{3.76\times10^{3}}$
	STD	8.91×10	5.77×10^{2}	3.18×10^{2}	5.37×10^{2}	1.59×10^{2}	$\mathbf{8.33\times10}$	2.44×10^{2}	9.09×10
F25	AVG	6.60×10^{3}	4.45×10^{3}	1.12×10^{4}	1.11×10^{4}	3.68×10^{3}	3.87×10^{3}	3.72×10^{3}	$\mathbf{3.34\times10^{3}}$
	STD	7.64×10^{2}	2.19×10^{2}	1.41×10^{3}	5.78×10^{3}	8.30×10	8.41×10	9.34×10	$\mathbf{5.07\times10}$
F26	AVG	1.65×10^{4}	2.68×10^{4}	2.74×10^{4}	2.27×10^{4}	1.55×10^{4}	1.88×10^{4}	1.53×10^{4}	$\mathbf{9.98\times10^{3}}$
	STD	$\mathbf{1.33\times10^{3}}$	4.61×10^{3}	1.95×10^{3}	3.69×10^{3}	1.40×10^{3}	2.60×10^{3}	2.83×10^{3}	2.18×10^{3}
F27	AVG	4.13×10^{3}	4.94×10^{3}	5.72×10^{3}	4.35×10^{3}	3.89×10^{3}	3.81×10^{3}	3.62×10^{3}	$\mathbf{3.45\times10^{3}}$
	STD	1.91×10^{2}	5.43×10^{2}	4.27×10^{2}	3.58×10^{2}	1.44×10^{2}	9.79×10	1.16×10^{2}	$\mathbf{4.64\times10}$
F28	AVG	8.79×10^{3}	5.45×10^{3}	1.55×10^{4}	1.68×10^{4}	3.76×10^{3}	4.17×10^{3}	4.44×10^{3}	$\mathbf{3.46\times10^{3}}$
	STD	1.22×10^{3}	4.49×10^{2}	1.89×10^{3}	4.88×10^{3}	9.11×10	1.69×10^{2}	1.14×10^{3}	$\mathbf{3.92\times10}$
F29	AVG	9.20×10^{3}	1.13×10^{4}	1.56×10^{4}	1.17×10^{4}	9.17×10^{3}	9.19×10^{3}	7.58×10^{3}	$\mathbf{6.37\times10^{3}}$
	STD	1.36×10^{3}	8.29×10^{2}	3.97×10^{3}	2.42×10^{3}	7.60×10^{2}	4.52×10^{2}	7.64×10^{2}	$\mathbf{4.23\times10^{2}}$
F30	AVG	1.24×10^{9}	1.89×10^{8}	8.64×10^{9}	1.22×10^{8}	1.12×10^{8}	5.23×10^{5}	1.90×10^{5}	$\mathbf{3.04\times10^{4}}$
	STD	1.13×10^{9}	6.60×10^{7}	4.44×10^{9}	6.44×10^{7}	3.94×10^{7}	2.66×10^{5}	2.40×10^{5}	$\mathbf{1.23\times10^{4}}$

4.4 收敛曲线分析

为了更直观地展示 JRBMO 在 CEC2017 测试集上的收敛性能，图 4 给出了部分测试函数的平均适应度收敛曲线。图 4 中，横轴表示迭代次数，纵轴表示 30 次独立运行后的平均适应度值。由图 4 可得出，JRBMO 在所有测试函数类型上均表现出优于其他对比算法的优势。具体而言，在单峰函数 F1 的收敛曲线上，JRBMO 在精度上明显优于其他算法，精度差距达到数量级。此外，算法在迭代至 1000 次时依然保持稳健，并未出现陷入局部最优的趋势，显示了 JRBMO 较优的寻优性能。在多峰函数 F7 和 F9 上，对比算法约在迭代 400 次后陷入局部最优，寻优速度显著减慢，而 JRBMO 在 800 次左右才逐渐减缓，这得益于探索阶段使用基于自适应螺旋围捕策略，能够随着迭代的推进动态增强算法的勘探范围。在混合函数 F12，F14 和 F18 上，JRBMO 展现出强大的寻优能力，收敛精度上至少领先其他算法一个数量级，且没有陷入局部最优。在组合函数 F24，F26 和 F30 上，JRBMO 也没有过早进入停滞，表明自适应维度变异策略能够有效根据适应度变化，帮助算法逃离当前区域，从而提升算法寻找潜在最优解的能力。

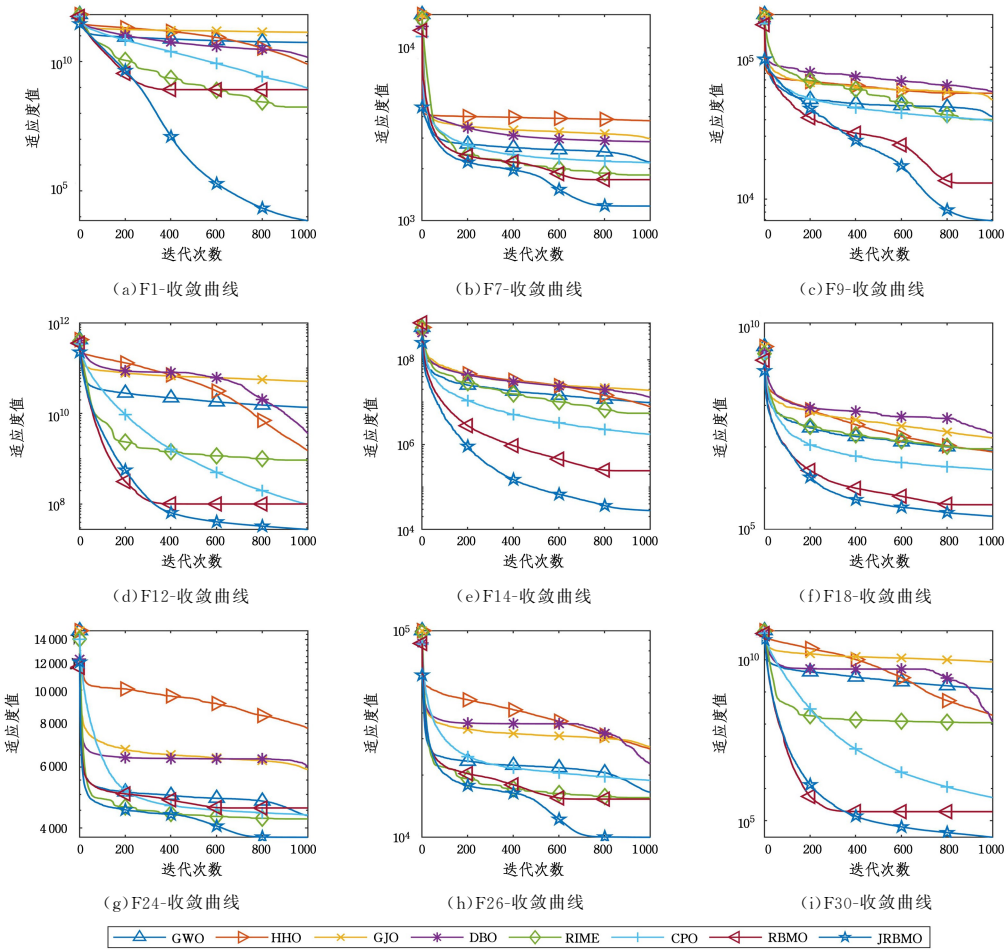


图 4 收敛曲线图

Fig. 4 Convergence curve graph

4.5 Wilcoxon 秩和检验与 Friedman 检验

为了更全面地评估算法的性能，本文采用了 Wilcoxon 秩和检验与 Friedman 检验进行非参数分析。其中，Wilcoxon 秩和检验用于统计分析两两算法之间的差异性，其结果以 p 值表示，当 p 值小于 0.05 时，表明两种算法的结果存在显著差异；而当 p 值大于等于 0.05 时，说明两种算法在统计意义上没有明显差异。Friedman 检验则用于评估多种算法的相对优劣，通过计算平均秩排名来比较各算法的优劣程度。

表 2 列出了 Wilcoxon 秩和检验与 Friedman 检验的统计分析结果，其中，“+/=/−”分别表示在 Wilcoxon 秩和检验中，JRBMO 算法相对于对比算法“优于/相当/劣于”的测试函数数量。

Wilcoxon 秩和检验结果表明，JRBMO 在多数测试函数中显著优于 GWO，CPO 和 RBMO，并且在所有测试函数中均显著超越 HHO，GJO，DBO 和 RIME。此外，JRBMO 在 Friedman 检验中获得了最高排名。结合之前的实验结果，可以得出，JRBMO 在解决复杂问题时相较于其他对比算法表现出更优越的性能。

表 2 Wilcoxon 秩和检验与 Friedman 检验结果

Table 2 Results of Wilcoxon Rank-Sum test and friedman test

算法	Wilcoxon(+/=/−)	Friedman
GWO	28/1/0	4.8966
HHO	29/0/0	6.1034
GJO	29/0/0	7.4828
DBO	29/0/0	6.8621
RIME	29/0/0	3.5828
CPO	26/2/1	3.4103
RBMO	28/1/0	3.9862
JRBMO	0/0/0	**1.1379**

4.6 改进策略有效性分析

为了验证每种策略的有效性，本节按比例选取 CEC2017 中不同类型的测试函数进行测试。具体来说，选取 F1，F4，F6，F9，F11，F14，F19，F22，F25，F30 共 10 个测试函数进行验证。测试的对比算法分别为：RBMO1——Hammersley 序列改进的 RBMO；RBMO2——自适应螺旋围捕策略改进的 RBMO；RBMO3——莱维飞行干扰的 RBMO；RBMO4——自适应维度变异策略改进的 RBMO。实验结果如表 3 所列。

表 3 策略有效性实验结果

Table 3 Experimental results of strategy effectiveness

		RBMO	RBMO1	RBMO2	RBMO3	RBMO4	JRBMO
F1	AVG	8.44×10^{8}	8.23×10^{8}	7.11×10^{4}	2.01×10^{4}	7.46×10^{7}	$\mathbf{7.11\times10^{3}}$
	STD	7.24×10^{8}	6.95×10^{8}	6.90×10^{3}	6.70×10^{3}	4.28×10^{4}	$\mathbf{6.27\times10^{3}}$
F4	AVG	1.02×10^{3}	9.07×10^{2}	7.28×10^{2}	7.31×10^{2}	8.85×10^{2}	$\mathbf{6.68\times10^{2}}$
	STD	1.02×10^{2}	8.87×10	8.43×10	$\mathbf{4.75\times10}$	5.82×10	5.17×10
F6	AVG	6.22×10^{2}	6.24×10^{2}	6.18×10^{2}	6.17×10^{2}	6.18×10^{2}	$\mathbf{6.08\times10^{2}}$
	STD	5.98	4.75	4.28	4.98	5.08	**2.93**
F9	AVG	1.30×10^{4}	1.32×10^{4}	1.07×10^{4}	1.03×10^{4}	1.24×10^{4}	$\mathbf{6.91\times10^{3}}$
	STD	3.97×10^{3}	3.81×10^{3}	4.12×10^{3}	2.88×10^{3}	3.43×10^{3}	$\mathbf{2.52\times10^{3}}$
F11	AVG	5.28×10^{3}	5.41×10^{3}	3.92×10^{3}	2.68×10^{3}	3.22×10^{3}	$\mathbf{2.30\times10^{3}}$
	STD	1.06×10^{3}	1.24×10^{3}	$\mathbf{1.01\times10^{2}}$	3.18×10^{2}	3.27×10^{3}	2.35×10^{2}
F14	AVG	2.62×10^{5}	2.59×10^{5}	1.24×10^{5}	1.03×10^{5}	4.79×10^{4}	$\mathbf{2.73\times10^{4}}$
	STD	2.96×10^{5}	2.89×10^{5}	1.52×10^{5}	9.62×10^{4}	2.88×10^{4}	$\mathbf{2.01\times10^{4}}$
F19	AVG	1.07×10^{4}	1.04×10^{4}	8.81×10^{3}	8.66×10^{3}	$\mathbf{7.17\times10^{3}}$	7.18×10^{3}
	STD	9.15×10^{3}	8.99×10^{3}	7.75×10^{3}	$\mathbf{5.26\times10^{3}}$	5.87×10^{3}	6.62×10^{3}
F22	AVG	2.34×10^{4}	2.30×10^{4}	2.10×10^{4}	1.77×10^{4}	1.68×10^{4}	$\mathbf{1.52\times10^{4}}$
	STD	2.06×10^{3}	1.99×10^{3}	1.86×10^{3}	1.26×10^{3}	1.16×10^{3}	$\mathbf{1.13\times10^{3}}$
F25	AVG	3.71×10^{3}	3.70×10^{3}	3.51×10^{3}	3.37×10^{3}	3.55×10^{3}	$\mathbf{3.35\times10^{3}}$
	STD	1.09×10^{2}	1.05×10^{2}	8.92×10	4.97×10	$\mathbf{4.86\times10}$	5.03×10
F30	AVG	1.21×10^{5}	1.23×10^{5}	1.04×10^{5}	1.12×10^{5}	4.66×10^{4}	$\mathbf{3.14\times10^{4}}$
	STD	1.06×10^{5}	1.04×10^{5}	9.13×10^{4}	9.38×10^{4}	3.34×10^{4}	$\mathbf{1.24\times10^{4}}$

从表中可以得出，使用单个策略的 RBMO1，RBMO2，RBMO3，RBMO4 在测试集上的性能均好于原 RBMO，且 JRBMO 在大多数测试函数上取得了最优结果。这表明每种策略均能够提升 RBMO 的性能且 JRBMO 能够有效地将提出的种策略融合。

4.7 补充实验

为充分凸显 JRBMO 的性能，本节选取了 5 种经典算法的变体以及 RBMO 在 CEC2019 测试集上与 JRBMO 进行比较实验。对比算法具体为：自适应人工电厂算法（IAEFA）[22]、助手辅助的金豺优化算法（HGJO）[17]、增量灰狼优化算法（IGWO）[23]、融合天牛须搜索的遗传算法（BASGA）[24]、自适应差分进化算法（L-SHADE）[25]。CEC2019 测试函数详情如表 4 所列。

为保证实验的公平性，本次实验以算法运行时间为终止条件，其值统一设置为 1 000 ms。表 5 列出了 JRBMO 与对比算法在 CEC2019 测试集上、最大运行时间为 1 000 ms 下的实验结果。

表 4 CEC2019 测试集

Table 4 CEC2019 test suite

函数名	编号	搜索范围	$F_{\min}$
Storn's6 Polynomial Fitting Problem	F31	[−8192,8192]	1
Inverse Hilbert Matrix Problem	F32	[16384,16384]	1
Lennard-Jones Minimum Energy Cluster	F33	[−4,4]	1
Rastrigin's	F34	[−100,100]	1
Rastrigin's	F35	[−100,100]	1
Weierstrass	F36	[−100,100]	1
ModifiedSchwefel's	F37	[−100,100]	1
Expanded Schaffer's F6	F38	[−100,100]	1
Happy Cat	F39	[−100,100]	1
Ackley	F40	[−100,100]	1

表 5 CEC2019 测试集上的实验结果(1 000 ms)

Table 5 Experimental results on CEC2019 test suite(1 000 ms)

	F31		F32	
	AVG	STD	AVG	STD
IAEFA	1.96×10^{2}	7.09×10	8.09	6.43×10^{-1}
HGJO	**1.00**	7.04×10^{-3}	4.98	7.13×10^{-1}
IGWO	**1.00**	7.10×10^{-8}	4.00	1.14
BASGA	1.46×10^{7}	1.51×10^{7}	3.37×10^{3}	1.43×10^{3}
LSHADE	**1.00**	4.32×10^{-7}	1.87	1.44
RBMO	1.03	1.98×10^{-6}	4.62	1.73
JRBMO	**1.00**	$\mathbf{6.19\times10^{-15}}$	**1.43**	$\mathbf{6.31\times10^{-1}}$
	F33		F34	
	AVG	STD	AVG	STD
IAEFA	1.24×10	3.16×10^{-1}	**3.11**	1.91
HGJO	8.06	1.01	1.03×10^{2}	1.17×10
IGWO	1.39×10	1.14	1.81×10^{2}	1.42×10
BASGA	2.24	1.98	2.71×10	1.08×10
LSHADE	2.41	1.33	9.24	2.99
RBMO	2.89	1.38	7.83	3.77
JRBMO	**2.13**	$\mathbf{8.01\times10^{-1}}$	4.03	**1.46**
	F35		F36	
	AVG	STD	AVG	STD
IAEFA	1.11×10	1.57	4.47	2.92
HGJO	1.27×10^{2}	2.19×10	8.30	9.66×10^{-1}
IGWO	2.81×10^{2}	6.51×10	1.53×10	1.43
BASGA	1.54	4.28×10^{-1}	4.33	1.31
LSHADE	2.00	2.92×10^{-2}	4.01	1.19
RBMO	1.99	3.11×10^{-2}	3.08	2.21
JRBMO	**1.02**	$\mathbf{1.56\times10^{-2}}$	**1.27**	$\mathbf{5.03\times10^{-1}}$
	F37		F38	
	AVG	STD	AVG	STD
IAEFA	8.07×10^{2}	4.38×10^{2}	6.88	9.21×10^{-1}
HGJO	2.21×10^{3}	$\mathbf{1.95\times10^{2}}$	2.36	2.49×10^{-1}
IGWO	2.71×10^{3}	2.53×10^{2}	5.73	1.98×10^{-1}
BASGA	9.02×10^{2}	2.93×10^{2}	4.57	5.33×10^{-1}
LSHADE	$\mathbf{1.78\times10^{2}}$	1.97×10^{2}	2.77	6.01×10^{-1}
RBMO	3.84×10^{2}	2.16×10^{2}	2.86	5.44×10^{-1}
JRBMO	2.52×10^{2}	2.00×10^{2}	**2.01**	$\mathbf{1.32\times10^{-1}}$
	F39		F40	
	AVG	STD	AVG	STD
IAEFA	8.11	9.53×10^{-2}	4.13×10	9.24×10^{-1}
HGJO	4.34	2.13×10^{-1}	2.14×10	1.31×10^{-1}
IGWO	4.92	3.07×10^{-1}	3.33×10	8.22×10^{-1}
BASGA	1.94	3.74×10^{-2}	2.24×10	8.71×10^{-2}
LSHADE	1.51	3.59×10^{-2}	$\mathbf{1.53\times10}$	1.09×10^{-1}
RBMO	1.88	3.94×10^{-2}	2.12×10	1.59×10^{-1}
JRBMO	**1.11**	$\mathbf{2.42\times10^{-2}}$	1.71×10	$\mathbf{6.42\times10^{-2}}$
	Fridman 平均排名			
IAEFA	5.5			
HGJO	4.1			
IGWO	5.8			
BASGA	3.7			
LSHADE	2.4			
RBMO	4.6			
JRBMO	**1.3**			

由表 5 可知,JRBMO 相比于其他算法整体上更优。从平均收敛精度看,JRBMO 在 F31－F33,F35－F36,F38,F39 上取得了最佳收敛精度。从 Fridman 检验排名来看,JRBMO 的排名为 1.3,在所有对比算法中排名第一,综合 4.3 节的结果可知,JRBMO 的整体性能较强。

5 JRBMO 的应用

5.1 拉(压)弹簧设计问题

拉(压)弹簧设计问题是工程应用中较为经典的多参数约束问题,其目的是在满足最小扰度、振动频率和剪应力的约束下,最小化拉压弹簧的重量[26]。图 5 给出了弹簧的 3D 图和 2D 截面侧视图,其中 D,d,N 是该问题的主要约束参数,分别代表弹簧外圈直径、弹簧线圈直径、绕线圈数。设优化目标函数为 $f(\boldsymbol{x})$,代表弹簧重量,即该工程问题建模为:

$$\boldsymbol{x}=(x_1,x_2,x_3)=(d,D,N)$$

$$\min f(\boldsymbol{x})=(x_3+2)x_2x_1^2$$

约束条件为:

$$g_1(\boldsymbol{x})=1-\frac{x_3x_2^3}{71\,785x_1^4}\leqslant 0$$

$$g_2(x)=\frac{4x_2^2-x_1x_2}{12\,566(x_2x_1^3-x_1^4)}+\frac{1}{5\,108x_1^2}$$

$$g_3(\boldsymbol{x})=1-\frac{140.45x_1}{x_3x_2^2}\leqslant 0$$

$$0.05\leqslant x_1,0.25\leqslant x_2\leqslant 1.3,2\leqslant x_3\leqslant 15$$

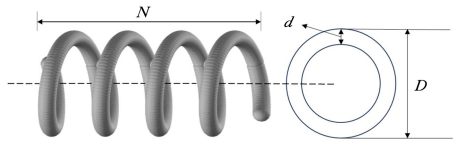


图 5 拉(压)弹簧设计

Fig. 5 Tension(compression) spring design

表 6 列出了 JRBMO 以及对比算法在该问题上的求解结果,从表中可以得出,JRBMO 在求解拉(压)弹簧设计问题中,表现优于其他 7 种算法。并且,相较于原算法,其寻优精度得到了一定程度的提升。

表 6 拉(压)弹簧设计优化结果

Table 6 Optimization results for tension(compression) spring design

算法	d	D	N	$f(x)$
GWO	0.054 899	0.438 850	7.716 5	0.012 852
HHO	0.058 054	0.530 050	5.475 2	0.013 354
GJO	0.05	0.317 294	14.061 5	0.012 741
DBO	0.05	0.317 266	14.048 9	0.012 729
RIME	0.065 384	0.781 230	2.754	0.015 877
CPO	0.051 710 9	0.357 233	11.262	0.012 672
RBMO	0.051 628 1	0.355 253	11.375 3	0.012 698
JRBMO	0.051 901 7	0.361 853	10.995	**0.012 665**

5.2 WSN 节点覆盖问题

无线传感器网络(Wireless Sensor Network,WSN)是由部署在监控区域内的大量传感器节点组成的多跳自组织网络系统[27]。其应用主要集中在军事、环境监测、安防监测、智能家居、灾害现场搜救等领域,尤其在恶劣环境下,WSN 能够替代人类进行所需信息的采集、传输和处理。而 WSN 的覆盖优化问题可以描述为在规定的监测区域内,保证传感器网络连通情况下的节点部署覆盖率最大化约束问题。然而,在实际的应用场景中,节点的部署往往是随机的,这易造成节点覆盖率低的问题,从而影响通信质量。为了解决以上问题,本节中使用 JRBMO 优化节点部署,提高其覆盖率。

设在给定三维区域空间中存在 M 个传感器,其点集合为 $(W_1,W_2,\cdots,W_m)$,空间坐标表示为 $\boldsymbol{W}_i=(x_w^i,y_w^i,z_w^i)$。每个传感器的感知半径为 R_s,则感知范围为以节点为中心、半径为 R_s 的球形区域。节点覆盖问题被建模为:在给定三维空间中存在均匀分布的 M 个待检测点 $(K_1,K_2,\cdots,K_m)$,坐标表示为 $\boldsymbol{K}_i=(x_k^i,y_k^i,z_k^i)$,当监测点 K_i 与传感器节点 $\boldsymbol{W}_i$ 之间的欧氏距离 $D(\boldsymbol{K}_i,\boldsymbol{W}_i)$ 小于感知半径 R_s 时表明该监测点 K_i 被传感器覆盖。其中,欧氏距离的计算式如下:

$$D(\boldsymbol{K}_i,\boldsymbol{W}_i)=\sqrt{(x_k^i-x_w^i)^2+(y_k^i-y_w^i)^2+(z_k^i-z_w^i)^2}$$

其单个节点$\boldsymbol{K}_i$被传感器$\boldsymbol{W}_i$感知到的概率为：

$$P(\boldsymbol{K}_i,\boldsymbol{W}_i)=\begin{cases}1, & D(\boldsymbol{K}_i,\boldsymbol{W}_i)\\ 0, & \text{else}\end{cases}$$

然而，在实际场景中单个监测点可能被多个传感器感知，则该监测点被检测到的概率可通过联合概率定义：

$$P(\boldsymbol{K}_i,\boldsymbol{W}_i)=1-\prod_{n=1}^{M}(1-(P(\boldsymbol{K}_n,\boldsymbol{W}_i)))$$

网络覆盖率C_r被表示为传感器覆盖的节点与总检测点之比，同时C_r也是求解的适应度函数。

$$C_r=\frac{\sum_{i=1}^{M}P(\boldsymbol{W},\boldsymbol{K}_i)}{M}$$

本文设计了两个实验，以证明JRBMO在WSN节点覆盖问题的性能。在两个实验中，三维空间区域的长宽高都设置为20。为了证明同维度下JRBMO的性能，在实验1中感知半径R_s设置为4和5，传感器数量M为30，最大迭代次数为100，实验结果如表7所列。

表7　覆盖率统计

Table 7　Coverage rate statistics

$M=30$	$R_s=4$		$R_s=5$	
算法	Best	Avg	Best	Avg
GWO	0.7444	0.7309	0.9611	0.9503
HHO	0.6876	0.6601	0.9079	0.8890
GJO	0.7254	0.6860	0.9544	0.9327
DBO	0.7110	0.6848	0.9268	0.9153
RIME	0.7396	0.7166	0.9671	0.9421
CPO	0.6628	0.6493	0.8974	0.8772
RBMO	0.7155	0.6954	0.9501	0.9295
JRBMO	**0.7848**	**0.7584**	**0.9834**	**0.9725**

从表7可以得出，JRBMO在不同感知半径R_s下，平均覆盖率和最优覆盖率均取得了最高值，分别达到了75.84%，78.48%，97.25%，98.34%。从平均覆盖率来看，在$R_s=4$时平均覆盖率比排名第二的GWO高出2.75%，比原算法高出6.3%。在$R_s=5$时，平均覆盖率比原算法高出3.33%。结果证明了在同维度下JRBMO的性能，表明JRBMO在实际应用问题中的适应性。

为了进一步证明JRBMO面对高维问题时的性能，在实验2中，取在实验1中综合排名前4的算法(JRBMO，GWO，RIME，GJO)进行不同传感器数下的节点覆盖实验。具体来说，将R_s设置为4，传感器数量M分别设置为30，35，40，45，即问题维度分别为90，105，120，135。图6给出了不同传感器数下不同算法的平均覆盖率对比图。

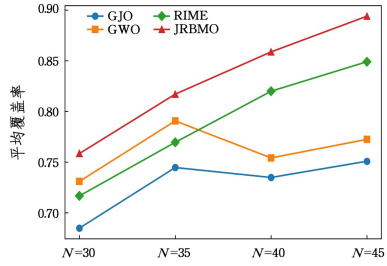


图6　不同传感器数下的平均覆盖率对比

Fig. 6　Comparison of average coverage under different sensor numberss

对图6进行横向分析，随着传感器数量M的增加，JRBMO和RIME算法优化的覆盖率均取得了提升，这是因为传感器数量越多，其覆盖率也随之增加。从增长的幅度来看，JRBMO均优于其他算法。GJO和GWO的增长幅度较小，甚至在$M=40$时，覆盖率减小，表明GJO和GWO在处理高维问题时性能较差。而JRMO的覆盖率呈线性增长，表明JRBMO具备较强的鲁棒性。从纵向分析看，JRBMO的覆盖率精度均高于对比算法，表明JRBMO的部署质量更高。综合实验1的结果，可以得出JRBMO在面对实际问题时，具备强大的适应能力，展现出了JRBMO卓越的性能。

结束语　本文为了解决RBMO多样性迅速退化、寻优精度差、易陷入局部最优的问题，提出了一种基于混合策略的自适应红嘴蓝鹊优化算法——JRBMO。并将JRBMO与经典及近期提出的算法在CEC2017和CEC2019测试集和典型应用中进行研究对比，实验结果表明，在CEC2017和CEC2019测试集上，JRBMO表现出更好的适应性、寻优精度和鲁棒性，证明了JRBMO的有效性。此外，在拉(压)弹簧设计问题、WSN节点覆盖问题上JRBMO均取得了最好的寻优结果，证明了JRBMO在实际工程问题中的优越性。未来研究应着重于JRBMO种群分布多样性评判，进行进一步自适应动态调整开发与探索的过程，同时采用个体参照小组或集群位置取均值的方法，应该进一步考虑自适应权重的方法进行加权求和，以增强算法的灵活性、鲁棒性。

参 考 文 献

[1] TALBI E G. Metaheuristics: From Design to Implementation [J]. John Wiley & Sons Google Schola, 2009, 2: 268-308.

[2] YU H, LI W, CHEN C, et al. Dynamic Gaussian bare-bones fruit fly optimizers with abandonment mechanism: method and analysis[J]. Engineering with Computers, 2020: 1-29.

[3] GHAREHCHOPOGH F S, IBRIKCI T. An improved African vultures optimization algorithm using different fitness functions for multi-level thresholding image segmentation[J]. Multimedia Tools and Applications, 2024, 83(6): 16929-16975.

[4] SOOD M, VERMA S, PANCHAL V K. Optimal path planning using swarm intelligence based hybrid techniques[J]. Journal of Computational and Theoretical Nanoscience, 2019, 16(9): 3717-3727.

[5] RAJWAR K, DEEP K, DAS S. An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges[J]. Artificial Intelligence Review, 2023, 56(11): 13187-13257.

[6] KATOCH S, CHAUHAN S S, KUMAR V. A review on genetic algorithm: past, present, and future[J]. Multimedia Tools and Applications, 2021, 80: 8091-8126.

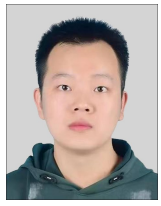
[7] PRICE K V. Differential evolution[M]// Handbook of Optimization: From Classical to Modern Approach. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013: 187-214.

[8] SU H, ZHAO D, HEIDARI A A, et al. RIME: A physics-based optimization[J]. Neurocomputing, 2023, 532: 183-214.

[9] MIRRASHID M, NADERPOUR H. Incomprehensible but Intelligible-in-time logics: Theory and optimization algorithm[J]. Knowledge-Based Systems, 2023, 264: 110305.

[10] MIRJALILI S, MIRJALILI S M, LEWIS A. Grey wolf optimizer [J]. Advances in Engineering Software, 2014, 69: 46-61.

[11] MAFARJA M, MIRJALILI S. Whale optimization approaches for wrapper feature selection[J]. Applied Soft Computing, 2018, 62: 441-453.
[12] CHOPRA N, ANSARI M M. Golden jackal optimization: A novel nature-inspired optimizer for engineering applications[J]. Expert Systems with Applications, 2022, 198: 116924.
[13] ABDEL-BASSET M, MOHAMED R, ABOUHAWWASH M. Crested Porcupine Optimizer: A new nature-inspired metaheuristic[J]. Knowledge-Based Systems, 2024, 284: 111257.
[14] WOLPERT D H, MACREADYW G. No free lunch theorems for optimization[J]. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 67-82.
[15] WU L, CHEN E, GUO Q, et al. Smooth Exploration System: A novel ease-of-use and specialized module for improving exploration of whale optimization algorithm[J]. Knowledge-Based Systems, 2023, 272: 110580.
[16] ZAMANI H, NADIMI-SHAHRAKIM H. An evolutionary crow search algorithm equipped with interactive memory mechanism to optimize artificial neural network for disease diagnosis[J]. Biomedical Signal Processing and Control, 2024, 90: 105879.
[17] WANG Z, MO Y, CUI M, et al. An improved golden jackal optimization for multilevel thresholding image segmentation[J]. PloS Pne, 2023, 18(5): e0285211.
[18] FU S, LI K, HUANG H, et al. Red-billed blue magpie optimizer: a novel metaheuristic algorithm for 2D/3D UAV path planning and engineering design problems[J]. Artificial Intelligence Review, 2024, 57(6): 1-89.
[19] MIRJALILI S, GANDOMIA H. Chaotic gravitational constants for the gravitational search algorithm[J]. Applied Soft Computing, 2017, 53: 407-419.
[20] HEIDARI A A, MIRJALILI S, FARIS H, et al. Harris hawks optimization: Algorithm and applications[J]. Future Generation Computer Systems, 2019, 97: 849-872.
[21] XUE J, SHEN B. Dung beetle optimizer: A new meta-heuristic algorithm for global optimization[J]. The Journal of Supercomputing, 2023, 79(7): 7305-7336.
[22] CHAUHAN D, YADAV A. An adaptive artificial electric field algorithm for continuous optimization problems[J]. Expert Systems, 40, 9(2023), e13380.
[23] REZAEI F, SAFAVI H R, ABD ELAZIZ M, et al. An enhanced grey wolf optimizer with a velocity-aided global search mechanism[J]. Mathematics, 2022, 10(3), 351.
[24] SEYYEDABBASI A, KIANI F, ALLAHVIRANLOO T, et al. Optimal data transmission and pathfinding for WSN and decentralized IoT systems using I-GWO and Ex-GWO algorithms[J]. Alexandria Engineering Journal, 2023, 63: 339-357.
[25] LI Y, HAN T, ZHOU H, et al. A novel adaptive L-SHADE algorithm and its application in UAV swarm resource configuration problem[J]. Information Sciences, 2022, 606: 350-367.
[26] DURDEV M, DESNICA E, PEKEZ J, et al. Modern swarm-based algorithms for the tension/compression spring design optimization problem[J]. Annals of the Faculty of Engineering Hunedoara, 2021, 19(2): 55-58.
[27] SHAIKH F K, ZEADALLY S. Energy harvesting in wireless sensor networks: A comprehensive review[J]. Renewable and Sustainable Energy Reviews, 2016, 55: 1041-1054.



DUAN Bowen, born in 1999, postgraduate, is a student member of CCF (No. V1576G). His main research interests include deep learning and computational intelligence

YIN Jinbin, born in 1976, Ph.D, associate professor. His main research interests include human-computer, deep learning and computational intelligence.