文章编号：2095-2163(2023)02-0006-09 中图分类号：TP391 文献标志码：A

基于权值分配策略的聚类天牛群优化算法

郭晓语，高 鹰，李 宁，周灿基，严基杰
（广州大学 计算机科学与网络工程学院，广州 510006）

摘 要：为改进天牛群优化算法在种群更新阶段存在的社会信息利用不足，及其在多峰函数中易陷入局部极值的情况，提出了一种基于权值分配策略的聚类天牛群优化算法。算法使用 k 均值聚类算法配合轮廓系数法，将天牛种群分为 k 个最佳聚类子群；分别选取各子群中适应度值最优的个体，并通过给定策略分配影响权值；最后使用多个最优个体共同决策的方式处理原算法中的社会学习部分，从而降低全局最优个体对种群位置更新的影响。实验选取了 15 个常用基准测试函数对所提算法进行仿真测试。实验结果表明，所提算法能够适应不同类型的优化问题，相较于天牛群优化算法及 3 个经典的智能优化算法拥有更好的寻优精度和稳定性。

关键词：天牛群优化算法；轮廓系数法；k 均值聚类；权值分配；社会学习

K-means clustering Beetle Swarm Optimization algorithm based on weight distribution strategy

GUO Xiaoyu，GAO Ying，LI Ning，ZHOU Canji，YAN Jijie
（School of Computer Science and Cyber Engineering，Guangzhou University，Guangzhou 510006，China）

【Abstract】 To improve the social information utilization rate of the Beetle Swarm Optimization（BSO）algorithm in the population regeneration stage and enhance the global optimization ability of the BSO algorithm in multimodal functions，a K-means clustering Beetle Swarm Optimization（KMBSO）algorithm based on weight distribution strategy is proposed. Firstly，this algorithm uses the k-means clustering algorithm and the silhouette coefficient method to divide the population into k-optimal clustering subgroups. Then the optimal individual in each subgroup is selected and their influence weight is allocated according to the corresponding fitness value. Finally，the social learning part of the original algorithm is optimized by a joint decision-making method of multi-optimal individuals to reduce the impact of the global optimal individual when the population location is updated. The proposed algorithm is simulated in 15 different benchmark functions and the experimental results show that it has better optimization accuracy and stability than the BSO algorithm and three classical intelligent optimization algorithms.

【Key words】 Beetle Swarm Optimization；silhouette coefficient method；K-means clustering；weight distribution；social learning

0 引 言

智能优化算法是通过模拟某一自然现象或过程而建立起来的元启发式算法，与传统的数学规划法相比，其依靠简单的规则和随机性可以实现复杂的多峰极值问题求解。对初始方案的弱依赖性以及不需要提供梯度信息等方面，成功地弥补了传统优化算法的不足，为复杂优化问题提供了一种全新的解决方法，众多文献都已证明智能优化算法在函数寻优上的高效性。目前，流行的智能优化算法包括：模拟鸟群觅食的粒子群优化算法（PSO）[1]、源于蚂蚁释放信息素行为的蚁群优化算法（ACO）[2]、受蜂群行为启发的人工蜂群算法（ABC）[3]，以及以化学物质的冷却和结晶为基础的模拟退火算法（SA）[4]等等。

天牛群优化算法（Beetle Swarm Optimization Algorithm，BSO）[5]是 2018 年由 Wang 等人提出的一种混合型智能优化算法。BSO 算法借鉴粒子群优化算法的思想，将天牛须搜索算法（Beetle Antennae Search Algorithm，BAS）[6]从单体智能转变为群体智

基金项目：广东省研究生教育创新计划项目（2020JGXM084）；大学生创新训练计划项目（202111078029，S202111078052）。

作者简介：郭晓语（1996-），男，硕士研究生，CCF 会员，主要研究方向：智能优化算法及知识图谱；高 鹰（1963-），男，博士，教授，CCF 会员，主要研究方向：智能优化算法及盲信号处理；李 宁（1998-），女，硕士研究生，主要研究方向：智能优化算法；周灿基（1999-），男，本科生，主要研究方向：智能优化算法；严基杰（2000-），男，本科生，主要研究方向：智能优化算法及自然语言处理。

通讯作者：高 鹰 Email：csgy@gzhu.edu.cn

收稿日期：2022-10-10

能，保留了所需调整参数少的优点，同时一定程度上改善了 BAS 算法在多峰函数上寻优表现不佳的问题。目前，已有许多 BSO 算法的应用案例。如：三维路径规划[7]、投资组合[8]、蓄电负荷控制[9]等。尽管 BSO 算法提出时间不久，但却吸引了众多研究人员对其进行改进。其中，胥松奇等[10]为强化天牛左右须的探测行为，提出了量子天牛群算法（QBSO），使得天牛个体在量子空间和实体空间中同时搜寻食物，以提高算法的收敛能力；Yang 等[11]通过在速度更新公式中引入自适应变异因子，提出自适应变异甲虫群优化算法（MBSO），以防算法过早收敛并提高算法多样性；王永贵等[12]提出引入变异策略的混沌天牛群搜索算法（CMBSOA），将混沌映射用于算法种群初始化阶段，以达到个体均匀分布的效果，并在位置更新上引入变异因子来增加算法跳出局部最优解的概率。

鉴于 BSO 算法存在收敛过快、精度低且在多峰问题上易陷入局部最优解的问题，本文提出了一种基于权值分配策略的聚类天牛群优化算法（K-means Clustering Beetle Swarm Optimization Algorithm Based on Weight Distribution Strategy，KMBSO）。算法以加强群体之间的信息交流为目的，利用 K-means 聚类算法配合轮廓系数法，将种群分为若干子群，并对每个子群中的最优个体分配一定权值以代替全局最优个体，从而实现对位置更新的联合决策。仿真实验结果表明，改进后的算法在给定的单、多峰函数上的表现明显优于 BSO 算法以及其它对比算法，具备较好的鲁棒性，并能有效地应用于优化问题。

1　算法原理

1.1　天牛群优化算法

在天牛群优化算法中，将包含有 n 只天牛个体的种群在 S 维中的表现形式描述为 $X=\{X_1,X_2,\cdots,X_n\}$，第 i 只天牛在 S 维空间中的位置用 $X_i=(X_{i1},X_{i2},\cdots,X_{iS})^{\mathrm{T}}$ 表示，天牛个体在空间中移动所需的速度变量被表示为 $V_i=(V_{i1},V_{i2},\cdots,V_{iS})^{\mathrm{T}}$。天牛群优化算法引用粒子群优化算法的思想，保留种群个体迭代过程中的个体最优位置和全局最优位置。其中，将 $P_i=(P_{i1},P_{i2},\cdots,P_{iS})^{\mathrm{T}}$ 表示为第 i 只天牛迭代过程中的最优位置，将 $P_g=(P_{g1},P_{g2},\cdots,P_{gS})^{\mathrm{T}}$ 表示为天牛种群在解空间中迄今搜索到的全局最优位置。天牛位置更新可通过式（1）实现：

$$X_{is}^{t+1}=X_{is}^{t}+\alpha V_{is}^{t}+(1-\alpha)\xi_{is}^{t} \tag{1}$$

其中，t 表示当前迭代次数，s 表示 S 维解空间中的第 s 维。V_{is} 和 ξ_{is} 分别表示第 i 只天牛在第 s 维空间中的移动速度和位置增量，算法主要通过更新两者的值，来控制天牛个体在解空间中搜寻最优解；α 表示设置的权重系数，取自 [0,1] 中的一个常数。

天牛个体移动速度 V 的计算采用粒子群中的速度更新策略，具体表示为：

$$V_{is}^{t+1}=\omega V_{is}^{t}+c_0r_0(P_{is}^{t}-X_{is}^{t})+c_1r_1(P_{gs}^{t}-X_{is}^{t}) \tag{2}$$

$$\omega=\omega_{\min}+(\omega_{\max}-\omega_{\min})\left(\frac{\mathrm{Max\_}iter-t}{\mathrm{Max\_}iter}\right) \tag{3}$$

$$c_0=d_1+1.2\cos(\pi t)/\mathrm{Max\_}iter \tag{4}$$

$$c_1=d_2-1.2\cos(\pi t)/\mathrm{Max\_}iter \tag{5}$$

其中，ω 称为惯性因子，调整其值可以改变算法全局寻优和局部寻优能力的比重；c_0 和 c_1 为加速因子，前者称为个体学习因子，后者称为社会学习因子；r_0、r_1 为两个单位随机变量；Max_*iter* 表示设定的算法最大迭代次数。

天牛个体的移动依赖于天牛须搜索算法所提供的位置增量 ξ，具体表示为：

$$\xi_{is}^{t+1}=\delta^{t}V_{is}^{t}sign(f(X_{ir}^{t})-f(X_{il}^{t})) \tag{6}$$

$$\begin{cases}X_{irs}^{t}=X_{is}^{t}+V_{is}^{t}d^{t}/2\\X_{ils}^{t}=X_{is}^{t}-V_{is}^{t}d^{t}/2\end{cases} \tag{7}$$

$$\delta^{t+1}=\eta\delta^{t} \tag{8}$$

$$d^{t}=\delta^{t}/c \tag{9}$$

$$\eta=\delta_{\min}(\delta_{\max}/\delta_{\min})^{1/(1+10t/\mathrm{Max\_}iter)} \tag{10}$$

其中，δ 是天牛的移动步长；$\delta_{\min}$ 和 $\delta_{\max}$ 为初始设置的范围值；d 是天牛左右两须的长度；η 是与 t 和 Max_*iter* 相关的自适应变量；c 是 [0,2] 内的一个常数值。δ 的初始赋值则取搜索空间范围值的一半，通过式（8）和式（9）进行迭代变化，进而影响 d 值。X_{ir}^{t} 和 X_{il}^{t} 表示天牛左右两须在解空间中的位置，$f(X_{ir}^{t})$ 和 $f(X_{il}^{t})$ 则表示两者对应的适应度值大小。

1.2　K-means 聚类算法

聚类算法起源于分类学，其主要作用是将相似的样本自动归类。在众多的聚类方法中，常见的方法有 K-means 聚类[13]、Hierarchical 聚类[14]、基于密度的噪声应用空间聚类[15]以及使用高斯混合模型的期望最大化聚类[16]等。其中，K-means 聚类算法因其简洁和高效率，成为使用最广泛、最著名的聚类算法。

对于 n 个处于 m 维空间的个体，K-means 聚类算法可以将这 n 个个体依据相似性分别聚集到指定

的 k 个簇中。在确定了 k 个初始化聚类中心点后，依据式(11)的欧氏距离公式，将每个个体依次分配到最近的簇中：

$$dis(X_i, C_j) = \sqrt{\sum_{t=1}^{m} (X_{it} - C_{jt})^2} \tag{11}$$

其中，X_i 表示第 i 个个体 $(1 \leqslant i \leqslant n)$；$C_j$ 表示第 j 个簇中心点 $(1 \leqslant j \leqslant k)$；$X_{it}$ 和 C_{jt} 分别表示对应个体与中心点在第 t 维上的属性值 $(1 \leqslant t \leqslant m)$。

每进行一次聚类操作后，则更新每个簇的中心点，而后按照式(11)中的规则继续操作，直到前后两次的聚类中心点的每个维度差值不超过给定值。更新中心点的方式为计算各个簇中个体在每个维度上的均值，具体的计算方式表示为

$$C_j = \frac{\sum_{Y_i \in S_j} Y_i}{|S_j|} \tag{12}$$

其中，S_j 表示第 j 个个体簇；$|S_j|$ 表示该簇中的个体数；Y_i 表示第 i 个个体 $(1 \leqslant i \leqslant |S_j|)$。

1.3 轮廓系数法

K-means 聚类算法作为无监督学习方法的一种，可以很好地对样本进行分类，但是对于簇的个数却较难确定。针对于此，常用解决方法有：手肘法[17]、轮廓系数法[18]、Calinski criterion 法[19]以及 Gap Statistic 法[20]等。本文使用轮廓系数法作为确定种群最优聚类数的方法。

在确定了聚类个数为 k 的前提下，轮廓系数可以作为衡量个体在簇内不相似度和簇间不相似度的指标。通过对不同聚类个数所对应的种群轮廓系数值进行比较，可以确定最优的聚类个数。具体的计算公式如下：

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{13}$$

$$T_k = \frac{\sum_{i=1}^{n} S(i)}{n} \tag{14}$$

其中，$a(i)$ 表示第 i 个个体与簇内其它个体间的平均距离；$b(i)$ 表示第 $i(1 \leqslant i \leqslant n)$ 个个体与其它簇中个体的平均距离的集合取最小值；n 为种群个体数量。$T_k(1 \leqslant k \leqslant n)$ 表示当聚类个数为 k 时，种群中个体的轮廓系数平均值，T_k 的值在 $[-1, 1]$ 中，其值越趋近于 1 则表示分类的情况越好。

2 改进天牛群优化算法

与许多智能优化算法一样，BSO 算法以当前全局最优个体作为个体移动的主要导向，但忽略搜寻过程中得到的局部最优解这一有用信息。本节针对该种情况，提出以下具体改进策略。

2.1 确定天牛子群

为确定最佳聚类子群数，需要计算在不同聚类数下的种群轮廓系数值，其中得到的最大轮廓系数值所代表的聚类个数便是最佳聚类数。这需要循环计算 n 次的种群轮廓系数值，其计算过程如下：

步骤 1 初始聚类个数 $k = 1$；

步骤 2 $k = k + 1$；通过式(11)进行聚类操作，并通过式(12)重新确定每个簇的中心点，重复本步骤直至达到设定的迭代次数；

步骤 3 通过式(13)计算每个天牛个体的轮廓系数，并通过式(14)计算得到所有天牛的轮廓系数均值，即天牛种群的总轮廓系数值；

步骤 4 重复执行 n 次步骤 2、3 的操作，比较各次得到的种群轮廓系数值以确定最佳聚类个数 k 及其对应的种群分类情况。

2.2 权值分配

在得到天牛群体的最佳分类方案后，选择各簇中的最优适应度个体作为簇代表。为分配不同簇代表对种群个体移动的影响权值，本文以其适应度值大小为指标进行权值分配。适应度值越小权值越大，相反适应度值越大则权值越小。依照适应度值可能出现的全为正、全为负和有正有负 3 种情况进行分类讨论，采用不同的计算方式以加快算法收敛，具体如式(15)所示：

$$W(x_j) = \begin{cases} \dfrac{1/f(x_j)}{\sum_{x_j \in K} (1/f(x_j))}, & \min(f(x_j)) \geqslant 0 \\ \dfrac{f(x_j)}{\sum_{x_j \in K} f(x_j)}, & \max(f(x_j)) < 0 \\ \dfrac{y_{\max} - f(x_j)}{\sum_{x_j \in K} (y_{\max} - f(x_j))}, & \text{else} \end{cases} \tag{15}$$

其中，集合 K 包含有 k 个不同的簇代表；x_j 表示 K 中的第 j 个个体；$f(x_j)$ 为其适应度值 $(1 \leqslant j \leqslant k)$；$y_{\max}$ 表示集合 K 中所有个体的最大适应度值；$W(x_j)$ 为该簇代表 x_j 所对应分配的权值大小。此外，为避免出现 $f(x_j)$ 为 0 而导致权值不能计算的情况，将使用最小浮点数精度 $eps = 2.22e{-}16$ 进行替代。

2.3 多子群融合

如式(1)所示，BSO 算法主要通过更新速度 V

和位置增量 ξ 的值来控制天牛个体在解空间中的移动，而其中天牛个体两须的位置同样会受到速度 V 的影响，从而间接影响位置增量。

本文将使用 $Q=\{Q_1,Q_2,\cdots,Q_k\}$ 来表示不同的簇代表，在保持个体学习部分不变的情况下，使用 Q 代替全局最优个体，并通过加权的方式更新社会学习部分，在更改速度公式的同时也间接地改变位置增量。具体的改进方法如下：

$$V_{is}^{t+1}=\omega V_{is}^{t}+c_0r_0(P_{is}^{t}-X_{is}^{t})+c_1r_1\sum_{j=1}^{k}W_j(Q_{js}^{t}-X_{is}^{t}) \tag{16}$$

其中，W_j 表示得到的第 j 个簇代表分配到的影响权值（$1\leqslant j\leqslant k$）；Q_{js}^{t} 表示第 j 个簇代表在第 s 维上的值；惯性权重 ω 以及加速因子 c_0、c_1 均采用自适应策略；r_0 和 r_1 为取值在 [0,1] 上的随机量。

通过对天牛群进行聚类操作可以优化天牛速度更新模块的社会学习部分，增强天牛个体之间的信息交流，使得算法迭代过程中得到的局部信息得以充分的利用，从而达到增强寻优能力的作用。在 BSO 算法中，社会学习部分只受到全局最优个体的影响，而改进后的 KMBSO 算法在该部分将受到 k 个子群的最优个体的影响，因此前者为后者在 $k=1$ 时的特例。

2.4　KMBSO 算法

通过上述对算法改进的描述，KMBSO 算法的具体实现步骤如下：

步骤 1　种群初始化。随机产生一个天牛种群 $G=\{X_1,X_2,\cdots,X_n\}$，设定速度 V、惯性权重 ω 和天牛移动步长 δ 等变量的范围值，以及最大迭代次数 Max_*iter* 和权重 α 等常量的值；

步骤 2　对 V、ω、δ 等自适应变量进行更新，随后使用 K-means 聚类算法配合轮廓系数法对天牛种群进行分类，并得到分类后的子群集合 $S=\{S_1,S_2,\cdots,S_k\}$；

步骤 3　在每个子群 S_j 中挑选出适应度值最优的簇代表 $Q_j=(Q_{j1},Q_{j2},\cdots,Q_{jS})^{\mathrm{T}}$，并依据 Q_j 对应适应度值，通过式（15）计算得到应分配的权重值 W_j；

步骤 4　确定每个天牛个体迄今搜索到的最优位置 $P=\{P_1,P_2,\cdots,P_n\}$。利用步骤 3 得到的权值 W_j 配合 Q_j 带入式（16）中，计算得到新的速度更新量 $V=\{V_1,V_2,\cdots,V_n\}$；

步骤 5　依据速度更新量 V_i 以及天牛左右两须的位置对位置增量 $\xi=\{\xi_1,\xi_2,\cdots,\xi_n\}$ 进行更新，并通过式（1）对两者进行加权处理，以此获得新一代的天牛种群；

步骤 6　对比前后两代天牛种群的适应度值，更新天牛个体最优位置以及全局最优位置；

步骤 7　判断算法是否达到初始设定的最大迭代次数。若未达到，转步骤 2 重新进行聚类操作；否则结束迭代过程，输出全局最优位置，即全局最优解。

3　仿真实验及结果分析

为验证改进策略的有效性，实验选取了 4 种不同的算法与改进后的算法进行比较。其中包括 BSO 算法、自适应粒子群优化算法（Adaptive Particle Swarm Optimization algorithm，APSO）、正弦余弦算法（Sine Cosine Algorithm，SCA）以及樽海鞘群优化算法（Salp Swarm Algorithm，SSA）。各算法在不同维度的测试函数中运行，并对所得实验结果进行 Wilcoxon 符号秩检验。

实验测试环境为：Inter（R）Core（TM）i5-4210M CPU @ 2.60 GHz，8 G 运行内存，Windows 7 操作系统，MATLAB R2018a。

3.1　测试函数

依照多样性的原则，本文选取 15 个国际基准测试函数对算法进行性能检验，函数详情见表 1。其中，$f_1\sim f_5$ 为单峰测试函数，用于测试算法的局部开发能力；$f_6\sim f_{10}$ 为多峰测试函数，侧重于测试算法的全局探索能力以及跳出局部最优解的能力；$f_{10}\sim f_{15}$ 为定维多峰测试函数，用于测试算法在低维复杂函数中的表现情况；D 表示搜索空间的维度。

表 1　标准测试函数

Tab. 1　Standard test functions

函数	函数名称	搜索范围	最优值
f_1	Sphere	$[-100,100]^D$	0
f_2	Schwefel 2.22	$[-10,10]^D$	0
f_3	Schwefel 1.2	$[-100,100]^D$	0
f_4	Step	$[-100,100]^D$	0
f_5	Quartic with Noice	$[-1.28,1.28]^D$	0
f_6	Rastrigrin	$[-5.12,5.12]^D$	0
f_7	Ackley	$[-32,32]^D$	0
f_8	Griewank	$[-600,600]^D$	0
f_9	Pentalty #1	$[-50,50]^D$	0
f_{10}	Pentalty #2	$[-50,50]^D$	0
f_{11}	Shekel's Foxhole	$[-65,65]^2$	0.998 0
f_{12}	Kowalik	$[-5,5]^4$	0.000 3
f_{13}	Six-hump Camel	$[-5,5]^2$	-1.031 6
f_{14}	Goldstein-Price	$[-2,2]^2$	3
f_{15}	Shekel 5D	$[0,10]^4$	-10.153 2

3.2 参数设定及结果分析

在实验参数的设定上，KMBSO 算法与 BSO 算法、APSO 算法保持一致。其中，对于天牛移动步长 δ 和左右两须距离 d 的影响因子 η 和 c，设定 $\delta_{\min}=0.2$，$\delta_{\max}=0.9$，$\eta=0.95$，$c=2$；对于权重系数 α，设定 $\alpha=0.4$；对于惯性权重和加速因子，设定 $\omega_{\min}=0.4$，$\omega_{\max}=0.9$，$d_1=1.3$，$d_2=2$。

为充分了解算法的寻优过程，实验设定种群规模为 300，运行代数为1 000，搜索空间的维度分别取 5 维、10 维和 30 维。所有算法在不同类型的测试函数上独立执行 30 次，实验数值结果见表 2～表 4。

表 2 5 种算法在单峰测试函数上的表现

Tab. 2 Performance comparison of the five algorithms on unimodal test functions

f	D	BSO			APSO			SCA			SSA			KMBSO	
		Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std
f_1	5	**5.51e−181**	0.00e+00	=	5.07e−120	9.93e−120	+	1.3e−100	5.9e−100	+	4.38e−11	2.56e−11	+	1.00e−180	**0.00e+00**
	10	3.95e−56	1.65e−55	+	1.59e−60	5.67e−60	=	2.15e−46	8.39e−46	+	4.14e−10	9.45e−11	+	**2.64e−62**	**9.84e−62**
	30	1.47e−02	3.08e−02	+	1.23e+04	8.83e+03	+	1.31e−07	2.83e−07	−	**4.92e−09**	**9.72e−10**	−	5.09e−04	8.77e−04
f_2	5	1.62e−30	8.73e−30	+	**1.96e−60**	**2.54e−60**	=	7.07e−55	1.22e−54	−	1.11e−06	3.14e−07	+	4.83e−52	2.25e−51
	10	1.10e−03	3.18e−03	+	2.00e+00	4.00e+00	=	**3.61e−29**	**7.09e−29**	−	4.96e−06	8.73e−07	+	3.62e−07	1.24e−06
	30	1.17e+00	1.77e+00	=	2.50e+01	1.36e+01	+	**2.59e−08**	**3.99e−08**	−	8.39e−02	1.82e−01	−	7.78e−01	6.28e−01
f_3	5	**9.7e−181**	0.00e+00	=	3.6e−119	1.1e−118	+	1.62e−77	8.52e−77	+	4.06e−11	1.92e−11	+	3.10e−180	**0.00e+00**
	10	**4.83e−45**	**2.6e−44**	−	1.50e+03	5.8e−78	=	1.95e−24	1.03e−23	+	5.19e−10	1.56e−10	+	3.63e−41	1.71e−40
	30	4.31e−01	1.01e+00	−	5.36e+04	1.01e+04	+	2.85e+02	5.90e+02	+	**9.92e−06**	**2.87e−05**	−	1.11e+01	1.11e+01
f_4	5	0.00e+00	0.00e+00	=	0.00e+00	0.00e+00	=	4.34e−03	1.95e−03	+	4.45e−11	2.12e−11	+	**0.00e+00**	**0.00e+00**
	10	0.00e+00	0.00e+00	=	0.00e+00	0.00e+00	=	1.21e−01	7.36e−02	+	3.67e−10	8.46e−11	+	**0.00e+00**	**0.00e+00**
	30	5.67e−03	7.29e−03	+	1.25e+04	6.73e+03	+	3.40e+00	2.46e−01	+	**4.68e−09**	**8.64e−10**	=	5.52e−04	1.08e−03
f_5	5	7.72e−05	7.53e−05	+	7.06e−05	7.29e−05	+	7.06e−05	7.29e−05	+	1.30e−04	8.60e−05	+	**1.46e−05**	**1.17e−05**
	10	2.64e−04	2.95e−04	+	4.34e−04	3.99e−04	+	1.63e−04	1.83e−04	+	6.94e−04	4.60e−04	+	**3.78e−05**	**2.36e−05**
	30	2.60e−03	1.84e−03	+	4.68e−01	1.97e+00	+	5.73e−03	4.58e−03	+	8.11e−03	3.91e−03	+	**1.90e−04**	**8.90e−05**
Total			8/5/2			9/6/0			11/0/4			11/1/3			+/=/−

表 3 5 种算法在多峰测试函数上的表现

Tab. 3 Performance comparison of the five algorithms on multimodal test functions

f	D	BSO			APSO			SCA			SSA			KMBSO	
		Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std
f_6	5	6.63e−01	1.10e+00	+	2.65e−01	4.40e−01	+	0.00e+00	0.00e+00	=	1.59e+00	9.47e−01	+	**0.00e+00**	**0.00e+00**
	10	4.98e+00	3.89e+00	+	1.07e+01	9.95e−01	+	**0.00e+00**	**0.00e+00**	−	8.89e+00	4.34e+00	+	4.33e−01	1.09e+00
	30	2.12e+01	1.30e+01	+	1.37e+02	4.98e+01	+	**1.41e+00**	**5.17e+00**	−	3.17e+01	1.16e+01	+	1.39e+01	1.13e+01
f_7	5	3.85e−15	1.61e−15	+	2.19e−15	1.71e−15	+	8.88e−16	9.86e−32	=	3.73e−06	1.13e−06	+	**8.88e−16**	**9.86e−32**
	10	7.16e−15	1.50e−15	−	7.67e+00	8.88e−16	+	**4.09e−15**	**1.07e−15**	−	5.49e−02	2.96e−01	+	5.40e−11	1.77e−10
	30	**9.31e−04**	**1.09e−03**	−	1.96e+01	1.44e+01	+	6.98e+00	9.20e+00	=	6.27e−01	7.08e−01	+	5.10e−01	4.24e−01
f_8	5	1.17e−01	6.92e−02	+	3.01e−02	4.67e−02	+	**0.00e+00**	**0.00e+00**	−	9.07e−02	4.55e−02	+	2.05e−03	4.67e−03
	10	2.25e−01	1.91e−01	+	1.58e−01	4.43e−02	+	**1.30e−09**	**6.93e−09**	−	2.40e−01	1.02e−01	+	9.97e−03	8.98e−03
	30	4.21e−02	2.44e−02	−	1.42e+02	3.46e−02	+	1.89e−02	5.71e−02	−	**9.68e−03**	**9.76e−03**	−	5.73e−01	1.67e−01
f_9	5	9.42e−32	6.57e−47	=	9.42e−32	6.57e−47	=	1.66e−03	7.84e−04	+	1.25e−12	6.01e−13	+	**9.42e−32**	**6.57e−47**
	10	5.18e−02	1.16e−01	=	2.07e−02	**4.71e−32**	=	3.22e−02	1.49e−02	+	5.18e−02	1.81e−01	+	**4.51e−23**	2.43e−22
	30	3.14e−01	3.23e−01	+	8.53e+06	3.57e+00	+	3.18e−01	6.33e−02	+	8.78e−01	9.08e−01	+	**3.71e−03**	**1.91e−02**
f_{10}	5	3.66e−04	1.97e−03	=	1.35e−32	5.47e−48	=	4.93e−03	2.07e−03	+	2.77e−12	1.50e−12	+	**1.35e−32**	**5.47e−48**
	10	4.03e−03	8.74e−03	=	7.32e−04	1.35e−32	=	1.13e−01	5.63e−02	+	**1.55e−11**	**4.95e−12**	−	3.66e−04	1.97e−03
	30	8.15e−01	7.69e−01	+	4.10e+07	1.07e−01	+	1.95e+00	1.84e−01	+	**2.93e−03**	**4.86e−03**	=	1.75e−02	2.19e−02
Total			8/4/3			11/4/0			6/3/6			12/1/2			+/=/−

表 4　5 种算法在定维多峰测试函数上的表现

Tab. 4　Performance comparison of the five algorithms on fixed-dimensional multimodal test functions

f	D	BSO			APSO			SCA			SSA			KMBSO	
		Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std	W	Mean	Std
f_{11}	2	9.98e−01	3.33e−16	=	9.98e−01	3.33e−16	=	9.98e−01	4.60e−07	+	9.98e−01	3.33e−16	=	**9.98e−01**	**3.33e−16**
f_{12}	4	7.69e−04	4.54e−04	=	1.20e−03	**1.73e−04**	+	5.78e−04	4.18e−04	=	**5.15e−04**	3.41e−04	=	6.14e−04	4.52e−04
f_{13}	2	−1.03e+00	0.00e+00	=	−1.03e+00	**0.00e+00**	=	−1.03e+00	2.66e−06	+	−1.03e+00	4.72e−15	=	**−1.03e+00**	1.13e−07
f_{14}	2	3.00e+00	4.44e−16	=	3.00e+00	4.44e−16	=	3.00e+00	3.07e−07	+	3.00e+00	2.01e−14	+	**3.00e+00**	**4.44e−16**
f_{15}	4	−1.02e+00	8.91e−01	+	−2.41e+00	1.12e+00	+	−5.19e+00	1.65e+00	+	−8.97e+00	2.15e+00	=	**−9.86e+00**	**5.73e−01**
Total			1/4/0			2/3/0			4/1/0			1/4/0			+/=/−

在定量分析方面，实验结果的对比采用 3 个指标分别是：重复多次实验后得到结果的平均值、标准差以及不同算法结果间以显著性水平 $\alpha = 0.05$ 为前提的 Wilcoxon 符号秩检验。其中，平均值可以衡量算法在求解问题上得到结果的质量好坏，平均值越小，结果越好；标准差可以衡量算法稳定性，表现出结果在平均值附近的震荡程度，标准差越小，算法越稳定。符号秩检验用于判断两组数据之间是否存在显著性差异，在表中使用“+”、“=”和“−”分别表示 KMBSO 算法的表现优于、相当和劣于其它算法。表中使用加粗的方式来表示 5 种算法在这两个对比指标中的最优数值结果，并在表尾处给出了检验结果的初步数量统计。

在表 2 的单峰测试函数问题中，KMBSO 算法在函数 f_4 和 f_5 中的表现优于其它算法，平均值和标准差上都拥有较高的寻优精度。但由于单峰函数本身只存在一个因函数值持续下降导致的极值点，所以不需要使用聚类策略，只利用全局最优点作为移动导向也有机会搜索到一个较好的结果（如：BSO 与 APSO 算法在 f_1 和 f_2 上得到的最小值结果），但当问题维度提高时，KMBSO 算法的性能便明显优于两者。此外，从符号秩检验的结果也可以看出，KMBSO 算法的局部开发能力明显优于其它算法，特别是与 SCA 算法和 SSA 算法的比较，皆为 11 个显著优胜。

在表 3 的多峰测试函数问题中，因为迭代次数和种群数量足够大，不同的算法在一些测试函数的求解中会出现相同的结果（如：5 维下的 f_6、f_9 和 f_{10} 函数）。KMBSO 算法的寻优性能相较于未改进前的 BSO 算法仍有提升，检验结果为 8 个显著优胜，4 个无显著差异，3 个显著劣解；相较于 APSO 算法和 SSA 算法，检验结果表现出显著的优势；对于 SCA 算法，两者的性能表现相当，检验结果为 6 个显著优胜，3 个无显著差异，6 个显著劣解。

在表 4 中的定维多峰测试函数下，KMBSO 算法表现良好，与其它算法的数值检验结果皆未出现显著劣解，且寻优表现优于 SCA 算法。

在算法从探索阶段转向开发阶段时，KMBSO 算法能够不断地跳出局部最优，得益于利用多个极值点共同进行决策，并不失侧重地依据适应度值分配权值。因此，会相应延长算法的全局搜索过程，有利于对解空间进行充分的探索，同时在开发阶段能够将种群个体导向更优的位置。例如：定维多峰函数 f_{15}，全局共有 5 个局部极小值点，其纵深较大，若算法只利用搜索到的全局最优点作为导向，很容易出现过早收敛以及陷入局部最优的情况，从而导致得到的实验结果较差。

表 5 对 5 种算法在不同维度下的检验结果进行了统计。可以看出，KMBSO 算法在绝大多数测试函数中的表现均优于其它对比算法，特别是在 5 维和 10 维空间中，整体表现优于 BSO 算法，这表明改进策略是可行有效的；但当维度升至 30 维时，KMBSO 算法与 SCA 算法、SSA 算法的表现相当。

表 5　5 种算法在不同维度上的 Wilcoxon 符号秩检验统计

Tab. 5　Wilcoxon signed-rank test statistics of five algorithms in different dimensions

Algorithm	D	+	=	−	Algorithm	D	+	=	−
KMBSO−BSO	≤5	6	9	0	KMBSO−APSO	≤5	8	7	0
	10	5	3	2		10	4	6	0
	30	6	1	3		30	10	0	0
Total				17/13/5	Total				22/13/0
KMBSO−SCA	≤5	10	3	2	KMBSO−SSA	≤5	11	4	0
	10	6	0	4		10	9	0	1
	30	5	1	4		30	4	2	4
Total				21/4/10	Total				24/6/5

在定性分析方面,图 1 中给出了不同的测试函数在给定维度下的算法数值结果收敛曲线对比结果,可以直观地了解到不同算法在迭代过程中的收敛情况。

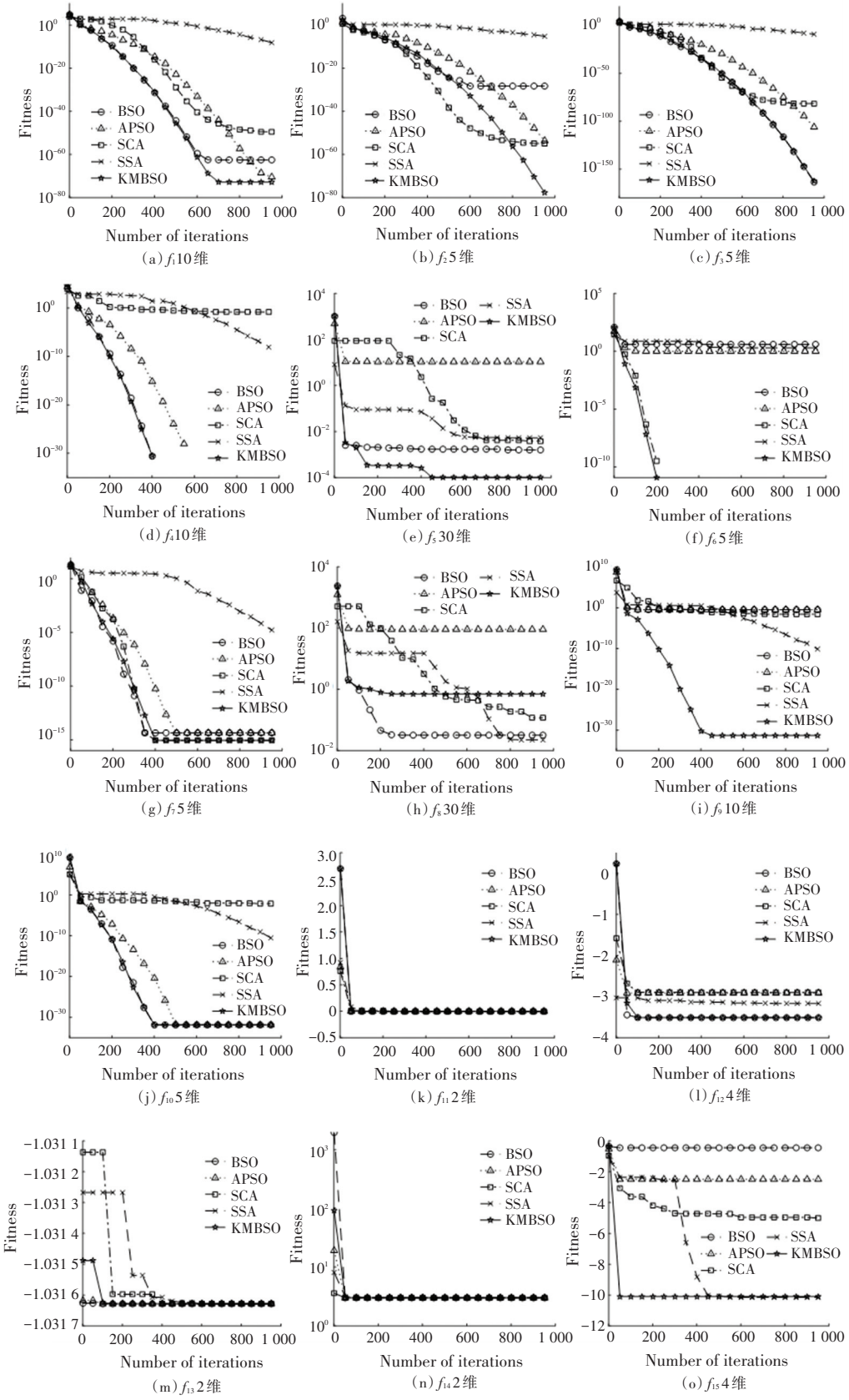

图 1 5 种算法的收敛曲线对比

Fig. 1 Comparison of convergence curves of the five algorithms

从图 1(a)~图 1(e)可知，KMBSO 算法在对应的单峰函数中的收敛速度相较于 BSO 和 APSO 算法稍慢一点，但却具有远优于 BSO 和 APSO 算法的收敛精度。在图 1(d)中，KMBSO 算法收敛却最快，迭代未到 400 次就得到最优值 0。而 APSO 和 BSO 算法会出现早熟现象，主要原因是搜索过程中全局最优个体对其它个体的强吸引力，同时缺少必要的变异手段，导致种群个体过早聚集；SCA 和 SSA 算法在多个函数的收敛过程中出现速度慢，精度低的现象，这表明两者的局部开发能力较弱。

从图 1(f)~图 1(o)可知，5 种算法的收敛过程主要集中在前中期，多数图中 BSO 和 APSO 算法的收敛同样早于 KMBSO 算法，但当其陷入局部最优时，KMBSO 算法依旧可以跳出并找到更优的解。其中，KMBSO 算法收敛的速度较慢也是因为聚类策略在帮助算法不断地跳出局部最优。SCA 和 SSA 算法的图像多处呈阶梯型，表明其同样具备跳出局部最优的能力，其中 SCA 算法出现较早收敛的现象，而 SSA 算法在迭代后期仍未收敛，呈现出可持续开发的能力。

综上所述，APSO 算法结构简单，但易出现早熟的现象，在 30 维的空间中往往易陷入局部最优；BSO 算法的性能优于 APSO 算法，特别是在低维空间中，但仍存在前期收敛过快和中期全局寻优能力较弱的问题；KMBSO 算法收敛较慢，但寻优性能强于两者，具备不断跳出局部最优的能力，且在 5 维和 10 维下能达到相较于 SCA 算法更高的数值精度；SSA 算法收敛速度较慢，局部开发能力有待提升，但在 30 维空间中的表现较好。

4　结束语

针对天牛群优化算法存在的不足，本文提出了一种新的基于聚类的天牛群优化算法，将聚类的思想融合到天牛种群中。通过将一个基础种群分为若干子群，使用多个子群中的最优个体代替全局最优个体，从而扩展局部极值在社会学习部分的影响范围。其次，本文提出了一种新的权值分配策略，该策略在利用群体智能求解极值问题上，可对影响群体移动的关键个体，按其对应的适应度值进行相应的权值分配。仿真实验结果表明，改进策略有利于算法跳出局部极值，并能够提高算法的寻优精度及稳定性，在单、多峰函数中的表现说明其具备较好的鲁棒性。

此外，在研究的过程中发现，使用其他的聚类算法进行种群分类时，在一些基准函数中的表现会优于使用 K-means 聚类算法。因此在接下来的工作中，将对这一方面开展进一步的探究。

参考文献

[1] KENNEDY J, EBERHART R C. Particle swarm optimization [C]//Proceeding of IEEE International Conference on Neural Networks. New York: IEEE Press, 1995:1942-1948.

[2] DORIGO M, BIRATTARI M, STUTZLE T. Ant colony optimization [J]. IEEE computational intelligence magazine, 2006, 1(4):28-39.

[3] KARABOGA D. Artificial bee colony algorithm [J].Scholarpedia, 2010, 5(3):6915-6945.

[4] KIRKPATRICK S, GELATT C D, VECCHI M P. Optimization by simulated annealing[J]. Science, 1983, 220(4598):671-680.

[5] WANG T T, YANG L, LIU Q. Beetle swarm optimization algorithm: theory and application [J]. Filomat, 2020, 34(15): 5121-5137.

[6] JIANG X, LI S. BAS: beetle antennae search algorithm for optimization problems [J]. Inter-national Journal of Robotics and Control, 2018, 1(1):1-5.

[7] MU Y, LI B, AN D, et al. Three-dimensional route planning based on the beetle swarm optimization algorithm [J]. IEEE Access, 2019, 7:117804-117813.

[8] CHEN T T, YIN H, JIANG H L, et al. Particle swarm optimization algorithm based on beetle antennae search for solving portfolio problem [J]. Computer Systems & Applications, 2019, 28(2):171-176.

[9] HE H, ZHOU S, ZHANG L, et al. Beetle swarm optimization algorithm-based load control with electricity storage [J]. Journal of Control Science and Engineering, 2020 (17):1-8.

[10] XU S Q, ZHOU S L, PI D Q. Damage identification of high-piled wharf′s lateral bent structure based on quantum beetle swarm optimization algorithm [J]. Port & Waterway Engineering, 2020 (8):91-99.

[11] YANG H W, XUE F, ZHU H M, et al. Web service composition optimization based on adaptive mutant beetle swarm[C]//Journal of Physics: Conference Series. Bristol UK: IOP Publishing, 2020, 1651(1):012061-012069.

[12] WANG Y G, LI S. A chaotic beetle swarm search algorithm with mutation strategy: China, CN111310885A[p].2020-06-19.

[13] MACQUEEN J. Some methods for classification and analysis of multivariate observations[C]//Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. Berkeley: University of California Press, 1967, 1(14):281-297.

[14] JOHNSON S C. Hierarchical clustering schemes [J]. Psychometrika, 1967, 32(3):241-254.

[15] BÄCKLUND H, HEDBLOM A, NEIJMAN N. A density-based spatial clustering of application with noise [J]. Data Mining TNM033, 2011:11-30.

[16] SHENTAL N, BAR-HILLEL A, HERTZ T, et al. Computing Gaussian mixture models with EM using equivalence constraints [J]. Advances in neural information processing systems, 2004, 16(8):465-472.

[17] JENSSEN R, ELTOFT T. A new information theoretic analysis of

sum-of-squared-error kernel clustering [J]. Neurocomputing, 2008, 72(1-3): 23-31.
[18] ROUSSEEUW P J. Silhouettes: a grap - hical aid to the interpretation and validation of cluster analysis [J]. Journal of computational and applied mathematics, 1987, 20:53-65.
[19] CALIŃSKI T, HARABASZ J. A dendrite method for cluster analysis [J]. Communications in Statistics-theory and Methods, 1974, 3(1):1-27.
[20] TIBSHIRANI R, WALTHER G, HASTIE T. Estimating the number of clusters in a data set via the gap statistic [J]. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 2001, 63(2): 411-423.

(上接第 5 页)

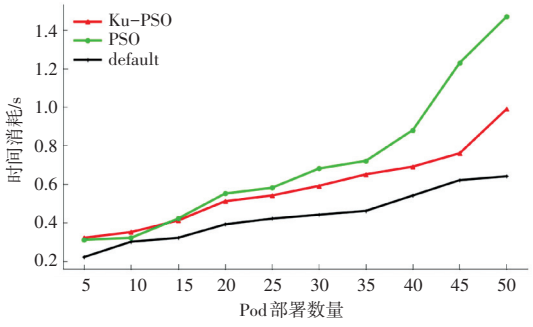

图 2 调度时间消耗对比

Fig. 2 Comparison of scheduling time consumption

对比 3 种调度方法的时间消耗,默认的调度算法由于调度过程较为简单,时间消耗最少;PSO 由于过程复杂,时间消耗最长;Ku-PSO 由于改进了后期的收敛速度,使得时间消耗在多 Pod 调度时得到迅速收敛,较传统的 PSO 算法消耗时间更短。

4 结束语

本文针对 Kubernetes 的默认调度算法中的不足进行改进,通过改进粒子群算法(PSO)得到 Ku-PSO 算法,使用二进制离散粒子群映射的方式将 Ku-PSO 应用于 K8s 多 Pod 部署过程中。Ku-PSO 极大程度改善了经典粒子群算法中容易陷入局部最优的问题,通过调整惯性因子非线性变化,给予前期适应值较低的粒子更高的速度,防止陷入局部最优,使得迭代过程个体学习因子和社会学习因子权值动态调整,前期的全局寻优能力和后期的局部收敛速度均得到保证。实验结果表明,使用 Ku-PSO 在保证集群负载均衡度和调度时间上都有非常不错的表现。

参考文献

[1] 武志学. 云计算虚拟化技术的发展与趋势[J]. 计算机应用, 2017,37(4):915-923.
[2] LI Chunlin, SUN Hezhi, TANG Hengliang, et al. Adaptive resource allocation based on the billing granularity in edge-cloud architecture[J]. Computer Communications, 2019, 145:29-42.
[3] Rafael Fayos-Jordan, Santiago Felici-Castell, Jaume Segura-Garcia, et al. Elastic Computing in the Fog on Internet of Things to Improve the Performance of Low Cost Nodes[J]. Electronics, 2019,8(12):1489.
[4] 张玉芳, 魏钦磊, 赵膺. 基于负载权值的负载均衡算法[J]. 计算机应用研究, 2012, 29(12): 4711-4713.
[5] 李华东, 张学亮, 王晓磊, 等. Kubernetes 集群中多节点合作博弈负载均衡策略[J]. 西安电子科技大学学报, 2021, 48(6): 16-22,122.
[6] 聂清彬, 潘峰, 吴嘉诚, 等. 基于改进蚁群算法的自适应云资源调度模型研究[J]. 激光与光电子学进展, 2020, 57(1): 90-96.
[7] KAEWKASI C, CHUENMUNEEWONG K. Improvement of container scheduling for docker using ant colony optimization [C]//2017 9th international conference on knowledge and smart technology (KST). IEEE, 2017: 254-259.
[8] DING J, XUE N, LONG Y, et al. Learning roi transformer for oriented object detection in aerial images[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019: 2849-2858.
[9] 王悦悦, 谢晓兰, 郭杨, 等. 基于自适应神经网络的云资源预测模型[J]. 科学技术与工程, 2021, 566(25): 10814-10819.
[10] 李蓉,沈云波,刘坚. 改进的自适应粒子群优化算法[J]. 计算机工程与应用,2015,51(13):31-36.
[11] 张鑫,邹德旋,肖鹏,等. 自适应简化粒子群优化算法及其应用[J]. 计算机工程与应用,2019,55(8):250-263.
[12] 翁理国,王骥,夏旻,等. 自适应种群更新策略的多目标粒子群算法[J]. 计算机工程与应用,2017,53(15):181-186.
[13] 刘建华, 杨荣华, 孙水华. 离散二进制粒子群算法分析[J]. 南京大学学报(自然科学版), 2011, 47(5): 504-514.