文章编号:1005-0523(2023)04-0112-15

融合迭代和问题维度的速度约束粒子群算法

王子航[1,2],刘建华[1,2],薛醒思[1,2],朱 剑[1,2],陈宇翔[1,2]

(1. 福建工程学院计算机科学与数学学院,福建 福州 350118;
2. 福建工程学院福建省大数据挖掘与应用技术重点实验室,福建 福州 350118)

摘要:粒子群算法广泛应用于工程、科学与管理等领域实际问题中的复杂优化问题求解,设计新的策略以应对算法的性能和效率瓶颈是该领域的研究热点。针对传统粒子群算法速度约束策略比较单一,容易导致算法收敛速度慢,性能低等问题,提出一种融合算法迭代和问题维度的速度约束策略。通过分析算法种群进化状态评估值与迭代次数及问题维度的关系,设计计算进化状态评估值的公式,使其受算法迭代次数和问题维度影响,最后根据进化状态评估值计算算法的速度约束范围,得到一种融合迭代和问题维度的速度约束粒子群算法。新的速度约束策略使粒子群算法的种群状态受到迭代次数和问题维度的影响,具有自适应性,并对不同维度问题求解具有扩展性,提高了粒子群算法的收敛速度和求解精度,仿真实验证明了算法的有效性。

关键词:粒子群优化算法;速度约束策略;进化状态预估;迭代次数;问题维度

中图分类号:TP18 **文献标志码**:A

本文引用格式:王子航,刘建华,薛醒思,等. 融合迭代和问题维度的速度约束粒子群算法[J]. 华东交通大学学报,2023,40(4):112-126.

DOI:10.16749/j.cnki.jecjtu.2023.04.003

Particle Swarm Optimization with Velocity Limit Combining Iteration and Problem Dimension

Wang Zihang[1,2], Liu Jianhua[1,2], Xue Xingsi[1,2], Zhu Jian[1,2], Chen Yuxiang[1,2]

(1. College of Computer Science and Mothematics, Fujian University of Technology, Fuzhou 350118, China;
2. Fujian Provincial Key Laboratory of Big Data Mining and Applications, Fujian University of Technology, Fuzhou 350118, China)

Abstract: PSO is widely used to solve complex optimization problems in practical problems in the fields of engineering, science and management. Designing new strategies to deal with the performance and efficiency bottlenecks of the algorithm is a research hotspot in this field. In order to solve the problem that the original velocity limit strategy of PSO is relatively simple, which may easily lead to slow convergence speed and low performance of the algorithm, this paper proposes a new velocity limit strategy combining iteration and problem dimension. By analyzing the relationship of the algorithm evolutionary state evaluation to iterations and the dimension of problem for particle swarm optimization, a formula was designed to calculate the ESE influenced by the iterations and problem dimension, and calculated the velocity limit on the basis of the ESE, so a particle swarm optimization with velocity limit combining iteration and problem dimension was obtained. Finally, the algorithm was affected by iteration and problem dimensions, adaptive and scalable for solving problems in different dimensions. The re-

收稿日期:2022-07-24

基金项目:福建省心理健康人机交互技术研究中心项目(2020L3024);福建工程学院发展基金(GY-Z20046);福州市科技创新平台项目(2021-P-052)

sults show that the strategy improves the convergence speed and accuracy. Experimental results prove the effectiveness of the algorithm.

Key words: particle swarm optimization; velocity limit strategy; evolutionary state evaluation; iteration; problem dimension

Citation format: WANG Z H, LIU J H, XUE X S, et al. Particle swarm optimization with velocity limit combining iteration and problem dimension[J]. Journal of East China Jiaotong University, 2023, 40(4): 112–126.

粒子群优化算法（particle swarm optimization, PSO）最初是由 Eberhart 等[1]和 Kennedy 等[2]于 1995 年提出，通过模拟动物群体的社会行为，比如鸟群的捕食行为，构建一种群体智能优化算法。与其他遗传算法[3]和蚁群算法[4]等智能算法相比 PSO 算法具有实现简单且收敛速度快的优势，广泛应用于很多领域，比如函数优化[5]，神经网络训练[6]，轨迹优化[7]等。PSO 算法在运算前期效果较好，在后期存在易陷入局部最优值问题，因此学者们提出了各种各样的改进方法。例如，邓浩等[8]提出了自适应权重参数调整的 PSO 算法；张德华等[9]提出了具有不同学习策略的 PSO 算法；Wang 等[10]则将其他优化算法与粒子群算法结合，得到新的 PSO 算法。各种 PSO 算法从 PSO 不同元素中改进，提升了算法精度与性能。

速度约束是 PSO 算法运行过程中的一个步骤操作，目的是防止粒子跳出搜索空间，避免出现不可行解。最早的速度约束策略采用固定值[11]，然后出现对 PSO 速度边界约束的研究和改进策略，例如，Helwig 等[12]提出并比较了多种用于粒子群优化的速度边界约束处理技术；Jiang 等[13]研究了速度约束策略解决高维问题存在的缺陷并提出了几种解决方案。近年来，有学者提出一些自适应速度约束策略，并提高了算法的性能。例如，Barrera 等[14]提出速度边界随迭代次数而几何变化的策略；Adewumi 等[15]提出在粒子每代计算速度最高值和最低值的绝对值，据此计算速度边界。

随迭代次数而变化的速度约束策略具有自适应性，可以更好地提高算法性能。但是，由于 PSO 算法的种群状态不确定，其局部搜索有可能发生在初始阶段，而全局搜索也可能发生在后期，因此速度边界不仅受迭代次数影响，还应该与粒子群的状态有关。Li 等[16]提出一种基于种群状态的自适应速度约束粒子群算法（particle swarm optimization with state-based adaptive velocity limit strategy，PSOSAVL），通过评估种群进化状态值而设置速度约束范围，提高算法性能。但 PSOSAVL 算法只是根据粒子之间的距离评估种群进化状态，并没有考虑迭代次数，所以还是存在陷入局部最优的风险。

同时，PSO 算法与求解问题的维度有关，根据维度提出改进算法的策略，可以提高其性能。例如，蒋晓岫等[17]提出，算法粒子每一维采用不同衰减权重，提高了多样性和局部搜索能力；邓志诚等[18]提出一种具有动态子空间随机单维变异粒子群算法，当某维退化且算法性能下降时，该维采用变异，有效地提升算法收敛速度和求解质量。然而，对于 PSO 算法速度约束与求解问题维度的关系，目前并没有发现文献研究。

基于上述的分析，本文提出一种融合迭代和问题维度的速度约束粒子群算法（PSO with velocity limit combining iteration and problem dimension，VLPSOID）。该算法不仅计算种群进化状态评估值（evolutionary state evaluation，ESE），而且考虑了迭代次数和求解问题的维度。算法首先设计受迭代次数和维度影响的计算公式，将其融合进化状态评估值计算中，再根据进化状态评价估值确定速度约束范围，使用算法速度约束既受迭代变化影响，也受到问题维度影响，具有自适应性，提高了粒子群算法的性能。

1　相关知识介绍

1.1　粒子群优化算法

在每次迭代过程中，粒子群算法计算每个粒子速度，更新自身位置，并且速度受到群体最优位置和自身历史最优位置的影响。标准的 PSO 算法的数学描述如下。

假设粒子种群规模为 N,问题维度为 D;第 i 个粒子位置向量 $\boldsymbol{x}_i=(\boldsymbol{x}_i^1,\boldsymbol{x}_i^2,\cdots,\boldsymbol{x}_i^D)$,速度向量 $\boldsymbol{v}_i=(\boldsymbol{v}_i^1,\boldsymbol{v}_i^2,\cdots,\boldsymbol{v}_i^D)$;第 i 个粒子第 d 维的速度和位置在 t 代的更新如下

$$\boldsymbol{v}_{id}^{t+1}=\boldsymbol{\omega}\cdot\boldsymbol{v}_{id}^{t}+c_1r_1(\boldsymbol{p}_{id}^{t}-\boldsymbol{x}_{id}^{t})+c_2r_2(\boldsymbol{g}_{d}^{t}-\boldsymbol{x}_{id}^{t}) \tag{1}$$

$$\boldsymbol{x}_{id}^{t+1}=\boldsymbol{x}_{id}^{t}+\boldsymbol{v}_{id}^{t+1} \tag{2}$$

式中:t 为当前迭代数;$\boldsymbol{\omega}$ 为惯性权重;c_1 和 c_2 为加速因子;$\boldsymbol{v}_{id}^{t}$ 为第 t 代第 i 个粒子第 d 维的速度;r_1 和 r_2 为介于 0 和 1 之间的随机数;$\boldsymbol{x}_{id}^{t}$ 为第 t 代第 i 个粒子第 d 维的位置;$\boldsymbol{g}_{d}^{t}$ 为第 i 个粒子第 d 维的个体历史最优位置;$\boldsymbol{p}_{id}^{t}$ 为整个粒子群第 t 代第 d 维的全局最优位置。

标准 PSO 算法速度 $\boldsymbol{v}_{id}^{t}$ 在按照式(1)更新后,要限制在$[-VL,VL]$范围内,$-VL$ 为速度下边界,VL 为速度上边界,$VL\geqslant 0$。标准的速度约束策略如式(3)所示,速度约束值 VL 采用固定值策略。本文提出一种自适应策略,使速度约束值 VL 受迭代次数和问题维度的影响而变化。

$$\boldsymbol{v}_{id}^{t}=\begin{cases}VL, & \text{if } \boldsymbol{v}_{id}^{t}\geqslant VL,\\ -VL, & \text{if } \boldsymbol{v}_{id}^{t}\leqslant -VL,\\ \boldsymbol{v}_{id}^{t}, & \text{else}\end{cases} \tag{3}$$

1.2 进化状态评估策略

进化状态评估策略是评估计算算法粒子种群状态值,判断算法当前搜索状态并改进算法,使用 PSO 算法具有自适应性。Zhan 等[19]提出一种自适应粒子群优化算法(adaptive particle swarm optimization, APSO),首次采用进化状态评估(evolutionary state evaluation, ESE)策略改进 PSO 算法;后来也有文献提出改进的 ESE 策略[20],提高了算法效率。ESE 策略考虑了种群粒子分布状态及其适应度值,根据进化状态评估值将种群状态分为 4 种类型:探索,开发,收敛以及跳出。种群进化状态评估值定义如下,假定 f 为种群进化状态评估值,则 f 值通过式(4)和式(5)两步计算得到。

$$\boldsymbol{L}_i=\frac{1}{N-1}\sum_{j=1,j\neq i}^{N}\sqrt{\sum_{d=1}^{D}(\boldsymbol{x}_i^d-\boldsymbol{x}_j^d)^2} \tag{4}$$

$$f=\frac{\boldsymbol{L}_g-\boldsymbol{L}_{\min}}{\boldsymbol{L}_{\max}-\boldsymbol{L}_{\min}} \tag{5}$$

式中:$\boldsymbol{L}_i$ 为第 i 个粒子到其他粒子的平均距离,$\boldsymbol{L}_g$ 为全局最佳粒子到其他粒子的平均距离,$\boldsymbol{L}_{\min}$ 和 $\boldsymbol{L}_{\max}$ 分别为 $\boldsymbol{L}_i$ 的最大值和最小值。根据式(5),$\boldsymbol{L}_i$ 值为 0 和 1 之间的值,式(4)是体现一个粒子到其他粒子之间平均距离,式(5)将种群最优粒子平均距离归一化处理,得到种群 ESE 值 f,然后用于控制粒子速度约束。

但是,式(5)评估种群进化状态值,只是采用全局最优粒子的平均距离,存在陷入局部最优风险。本文提出一种融合迭代和问题维度的自适应进化状态评估策略,改进种群状态评估方法,改进粒子群算法的速度约束,形成一种融合迭代和问题维度的速度约束粒子群算法。

2 融合迭代和问题维度的速度约束粒子群算法

本算法采用了融合迭代和问题维度的自适应进化状态评估策略,其方法是先由式(4)和式(5)计算 f 值;然后让其受当前迭代次数 t 和求解问题维度 D 影响,得到新的状态评估值 $\hat{f}$,如式(6)。这里 $s(t)$ 和 $h(D)$ 分别表示受到迭代次数 t 和问题维度 D 影响的公式,各自采用了不同影响策略,影响着原始种群状态评估值 f。$s(t)$ 为迭代策略,$h(D)$ 为维度策略

$$\hat{f}=f\eta s(t)h(D) \tag{6}$$

式中:η 为影响系数,$\eta\in(0,1)$,控制两个策略对原始种群状态评估值 f 的影响程度。

2.1 迭代策略

PSO 算法要尽可能地在前期处于探索状态,在后期处于开发状态。当种群状态评估值采用式(5)计算 f 值时,应考虑迭代次数 t 对其影响,目的是使 f 值在前期相对较大,在后期慢慢变小。这样算法在前期全局搜索能力较强,能够搜寻更大空间,容易跳出局部最优点;算法在后期局部搜索能力增强,重点搜索有希望的局部区域。因此,本小节设计一个迭代策略,使其影响种群状态评估值 f,使其值随着迭代递减。

本文设计迭代策略 $s(t)$如式(7),t_{max} 为最大迭代次数。图 1 则表示 $s(t)$是随迭代 t 指数递减的图像。如图 1 所示,$s(t)$在迭代前期,值相对较大,而随迭代变化,$s(t)$值缓慢下降,最后值为 1。指数递减比线性递减下降更快,容易使 f 值突变,全局搜索能力下降快,其变化范围从 e 到 1。

$$s(t)=e^{\frac{t_{max}-t}{t_{max}}} \tag{7}$$

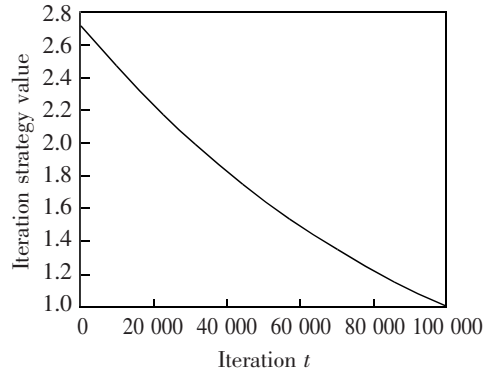


图 1　迭代策略值随迭代次数的变化

Fig.1　Change of iteration strategy value with iteration

2.2　维度策略

随着问题维度增大,PSO 算法计算成本增加,搜索效率降低。由于在更新高维的粒子位置时,粒子每个维度都同时到达最佳位置的概率降低,粒子更可能远离最佳位置,降低算法性能。因此 PSO 算法会受到问题维度的影响,本节设计维度策略,使种群状态评估值 f 受求解问题维度 D 影响,其值随维度 D 增加而增加,增强算法的全局搜索能力。

本文设计策略函数 $h(D)$如式(8)所示,其变化如图 2 所示。在低维时,h 值变化比较大,而随着维度增加,h 值变化慢慢变平缓,最终趋于稳定。

$$h(D)=1+e^{-\frac{1}{D}} \tag{8}$$

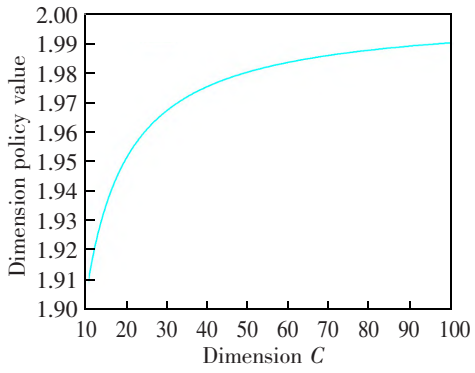


图 2　维度策略值随问题维度的变化

Fig.2　The change of dimension policy value with the problem dimension

2.3　种群进化状态评价方法

在式(6)的基础上,融合迭代策略和维度策略到 ESE 值 f 中,得到新 ESE 值 $\hat{f}$,其计算如下

$$\hat{f}=f\eta e^{\frac{t_{max}-t}{t_{max}}}\left(1+e^{-\frac{1}{D}}\right) \tag{9}$$

式中:η 为影响系数,用于控制两种策略影响 ESE 值的程度,$\eta\in(0,1)$,具体取值由实验方法决定;t 为迭代次数;t_{max} 为最大迭代次数;D 为求解问题维度。

根据式(9),$\hat{f}$值受迭代次数 t 影响,在前期相对较大,在后期慢慢减小;同时,$\hat{f}$值还会受到问题维度的影响,随着维度的增加,$\hat{f}$值也增大,全局搜索能力增强。

2.4　进化状态评估值决定速度约束的策略

本节根据式(9)的 ESE 值 $\hat{f}$,设计一个 PSO 算法自适应速度约束策略,计算速度约束值 VL。

2.4.1　速度约束的计算方法

假定 μ 为 PSO 速度约束值(VL)与其搜索最大位置(x_{max})比例值,$\mu\in(0,1)$,而速度约束 VL 限制在速度下边界和上边界$[VL_{min},VL_{max}]$之间。

为了使速度边界随 ESE 值 $\hat{f}$ 自适应调整,μ 设置为$\hat{f}$的 sigmoid 函数值,如式(10)所示,使$\mu\in(0,1)$,最终得到速度约束值 VL

$$\mu=\text{sigmoid}(\hat{f})=\frac{1}{1+\alpha e^{-\beta\hat{f}}} \tag{10}$$

$$VL=\frac{1}{1+\alpha e^{-\beta\hat{f}}}x_{max} \tag{11}$$

2.4.2　超参数设置方法

α 和 β 为两个超参数设置基于原则为,当$\hat{f}$值越大时,PSO 算法种群倾向于全局搜索,粒子速度约束边界变大;而当$\hat{f}$越小时,种群倾向于局部搜索,粒子速度约束边界收缩。当$\hat{f}$=1 时,VL 达到最大值 VL_{max},此时 $\mu=\mu_{max}$;当$\hat{f}$=0 时,VL 达到其最小值 VL_{min},此时 $\mu=\mu_{max}$。μ_{min},μ_{max} 分别为 VL 与 x_{max} 的最大比例

和最小比例。假设 $\mu_{\min}$ 和 $\mu_{\max}$ 已知，根据式(11)可得式(12)，推导求解 α 和 β 值。当给定速度约束 VL 的比例最大值 $\mu_{\max}$ 和比例最小值 $\mu_{\min}$ 时，可以根据式(12)设置式(11)的超参数 α 和 β 值。

$$\begin{cases} \text{sigmoid}(\hat{f}=0)=\dfrac{1}{1+\alpha}=\mu_{\min} \\ \text{sigmoid}(\hat{f}=1)=\dfrac{1}{1+\alpha e^{-\beta}}=\mu_{\max} \\ \alpha=\dfrac{1}{\mu_{\min}}-1 \\ \beta=-\ln\left[\left(\dfrac{1}{\mu_{\max}}-1\right)/\alpha\right] \end{cases} \tag{12}$$

2.5 融合迭代和问题维度的速度约束 PSO 算法

本节 PSO 算法速度约束公式为式(3)，其速度边界值 VL 采用式(11)计算。式(11)VL 由进化状态评估 ESE 值 $\hat{f}$ 决定，而 $\hat{f}$ 值融合了迭代次数和问题维度的影响，得到一种融合迭代和问题维度的速度约束 PSO 算法。

当 PSO 算法求解的问题维度 D 和其当前迭代次数 t 已知，并设置好 $\mu_{\min}$ 和 $\mu_{\max}$ 时，可以用本文提出的相关策略及其公式，求解速度约束值 VL 及其范围值 $[-VL,VL]$，其算法具体过程的伪代码见算法 Calc_VL，本文提出 PSO 算法流程的伪代码见算法 Calc_Best。

```
算法 Calc_VL
输入：迭代次数 t，问题维度 D，种群规模 N
输出：速度约束值 VL
BEGIN
通过式(12)设置参数 α 和 β
FORi=1：N
通过式(4)计算 L_i
END FOR
通过式(9)计算 ESE f̂
按式(11)计算速度约束值 VL
END BEGIN
```

```
算法 Calc_Best
输入：种群规模 N 最大迭代次数 t_max，
输出：全局最优位置 g
BEGIN
初始化：种群位置 x，速度 v，个体最优位置 p，全局最优位置 g
FORt=1：t_max
通过算法 Calc_VL 获取速度约束值 VL
FORi=1：N
FORd=1：D
按式(1)更新粒子速度 v_id^t
根据 VL 并通过式(3)调整粒子速度 v_id^t
按式(2)更新粒子位置 x_id^t
END FOR
END FOR
计算种群适应度值
更新 p 和 g
END FOR
END BEGIN
```

2.6 时间复杂度分析

根据算法 Calc_VL 和算法 Calc_Best，VLPSOID 算法的时间成本主要是：计算种群状态，更新粒子的速度和位置。对于算法 Calc_VL，时间成本主要为式(4)，需要计算粒子之间的距离，涉及粒子之间每个维度，N 个粒子之间要相互计算，所以时间复杂度为 $O(D*N^2)$。对于算法 Calc_Best，由于每次迭代调用算法 Calc_VL，最后时间复杂度为 $O(t_{\max}*D*N^2)$。算法 Calc_Best 在更新粒子速度和位置时，时间复杂度为 $O(t_{\max}*D*N)$。综上所述，VLPSOID 算法的最终时间复杂度为 $O(t_{\max}*D*N^2)$，主要时间成本花费在计算粒子之间的距离，原因是需要计算算法种群的状态评估值(ESE)。

3 实验分析

本节实验分析方法用 PSO 算法求解 CEC2013 中的 28 个基准测试函数[21]，测试对比 VLPSOID 算法与其他相关 PSO 算法的性能。28 个函数的信息如表 1 所示，共分为 3 类：f_1~f_5 为单峰函数；f_6~f_{20} 为多峰函数；f_{21}~f_{28} 为复合函数。与之对比算法有：基于种群状态的自适应速度约束粒子群算法(PSOSAVL)，基于分层自主学习的改进粒子群优化算法(HCPSO)[22]，具有拓扑时变和搜索扰动的混合粒子群优化算法(HPSO)[23]，以及标准 PSO 算法；各算法的实验参数设置采用原始论文的数据，如表 2 所示。

表 1　CEC 2013 测试函数
Tab.1　CEC 2013 benchmark functions

No.	Function	Best
f_1	Sphere Function	−1 400
f_2	Rotated High Conditioned Elliptic Function	−1 300
f_3	Rotated Bent Cigar Function	−1 200
f_4	Rotated Discus Function	−1 100
f_5	Different Powers Function	−1 000
f_6	Rotated Rosenbrocks Function	−900
f_7	Rotated Schaffers F7 Function	−800
f_8	Rotated Ackleys Function	−700
f_9	Rotated Weierstrass Function	−600
f_{10}	Rotated Griewanks Function	−500
f_{11}	Rastrigins Function	−400
f_{12}	Rotated Rastrigins Function	−300
f_{13}	Non−Continuous Rotated Rastrigins Function	−200
f_{14}	Schwefels Function	−100
f_{15}	Rotated Schwefels Function	100
f_{16}	Rotated Katsuura Function	200
f_{17}	Lunacek Bi_Rastrigin Function	300
f_{18}	Rotated Lunacek Bi_Rastrigin Function	400
f_{19}	Expanded Griewanks plus Rosenbrocks Function	500
f_{20}	Expanded Scaffers F6 Function	600
f_{21}	Composition Function 1 (n=5,Rotated)	700
f_{22}	Composition Function 2 (n=3,Unrotated)	800
f_{23}	Composition Function 3 (n=3,Rotated)	900
f_{24}	Composition Function 4 (n=3,Rotated)	1 000
f_{25}	Composition Function 5 (n=3,Rotated)	1 100
f_{26}	Composition Function 6 (n=5,Rotated)	1 200
f_{27}	Composition Function 7 (n=5,Rotated)	1 300
f_{28}	Composition Function 8 (n=5,Rotated)	1 400

表 2　用于比较的算法以及参数
Tab.2　Algorithms and parameters for comparison

Algorithm	Parameter
PSOSAVL	ω=0.9~0.4,c_1=c_2=2.05,v_{max}=100
HPSO	ω=0.9~0.4,c_1=c_2=2,v_{max}=100
HCPSO	ω=0.729 6,c_1=c_2=2,v_{max}=100
PSO	ω=0.9~0.4,c_1=c_2=2,v_{max}=100

3.1　影响系数 η 的敏感性分析

VLPSOID 算法式(9)的影响系数 η 控制迭代次数和问题维度对原始 ESE 值 f 的影响强度,$\eta\in(0,1)$。但 η 具体取值表现为影响敏感程度,本节采用实验方法分析,根据 η 取不同值时,分析 VLPSOID 算法优化部分函数的实验性能,寻求一个最优 η 取值。

实验的种群粒子数为 30 个,最大迭代次数为 20 000 次,实验选择的优化函数是 f_3,f_8,f_{13},f_{18},f_{23},f_{28} 6 种函数,函数维度分别为 20,40,60,80,100 维;η 取[0,1]之间均衡地取不同的 10 个值,实验比较同一函数在同一个维度,其函数适应度值的大小。实验结果如表 3 所示,在 η 取不同值时,算法计算函数在不同维度时的最优值。

算法处于不同维度时,采用不同的 η 值对结果的影响也是不同的。最终统计了所选 6 个函数的结果,得到总结果表。由总结果表得到,当 η 值分别取 0.6,0.7,0.8 时,取得较好解的次数分别为 6 次,7 次,10 次。相比较其他值,当 η 值为 0.6,0.7,0.8 时,算法取得的结果更为突出。因此,为了能够整体提升算法的性能,本文建议影响系数 η 取值范围在[0.6,0.8]。

3.2　对 μ_{min} 和 μ_{max} 的敏感性分析

算法的 μ_{min} 和 μ_{max} 是分别表示 VL 与位置最大值 μ_{max} 的最小比例和最大比例,决定式(12)计算超参数 α 和 β 值。本节针对部分测试函数,采用实验方法,分析其取值的敏感度。实验的测试函数为 f_{12},f_{18},f_{23},f_{26};函数的维度为 50 维,算法种群大小为 30 个,最大迭代次数为 50 000 次。实验方法是,先固定 μ_{min} 为 0.5,测试 μ_{max} 取[0.4,1.0]范围内不同值时,算法求解函数最优值,实验结果如表 4 及图 3 左侧所示。然后,固定 μ_{max} 为 0.7,测试 μ_{min} 取[0.1,0.7]范围内不同值时,算法求解函数最优值,实验结果如表 5 及图 3 右侧所示。

观察表 4 数据,本文算法对 μ_{max} 的取值不敏感并且不同的 μ_{max} 取值都提供了较为满意的结果,因此可以得出结论,本文算法对于变化的 μ_{max} 具有健壮性。根据表 4 以及图 3 左侧的结果,建议 μ_{max} 取值范围[0.6,0.7]。观察表 5 数据,算法对 μ_{min} 取值有一定的敏感性,当 μ_{min} 取过小时,算法的优化效果变差,从图 3 各子图的右侧图,也可得到验证。因此,应避免 μ_{min} 取 0.1 和 0.2 值。观察表 5 和图 3 左侧,μ_{min} 取值范围应该在[0.5,0.6]之间最佳。综上所述,VLPSOID 对于变化的 μ_{max} 具有鲁棒性,而对 μ_{min} 值具有敏感性。本文建议在 [0.6,0.7] 内选择 μ_{max},在[0.5,0.6]内选择 μ_{min}。

表3 不同维度的最优值结果

Tab.3 The best result with different η in different dimension

Function	η	Dimension					Best
		20	40	60	80	100	
f_3	0.1	5.01e+08	1.33e+09	1.79e+10	2.36e+10	1.04e+13	1
	0.2	5.43e+07	5.27e+08	2.15e+10	1.35e+11	2.52e+13	1
	0.3	4.08e+08	2.30e+09	1.09e+10	7.80e+10	2.28e+12	0
	0.4	4.54e+07	3.64e+09	9.45e+09	6.15e+10	1.36e+12	0
	0.5	3.50e+07	2.72e+09	3.22e+10	3.23e+10	1.11e+13	0
	0.6	1.75e+06	1.54e+09	4.49e+10	2.33e+11	1.66e+12	1
	0.7	1.37e+08	2.63e+09	9.39e+09	6.79e+10	1.28e+12	1
	0.8	4.16e+06	1.10e+09	1.16e+10	4.47e+10	7.50e+12	0
	0.9	1.55e+08	6.47e+08	1.10e+10	4.14e+10	2.26e+14	0
	1.0	2.13e+06	1.65e+09	1.40e+10	6.47e+10	3.75e+11	1
f_8	0.1	20.708	20.962	21.139	21.244	21.337	0
	0.2	20.697	20.803	21.075	21.214	21.358	0
	0.3	20.685	20.918	21.186	21.215	21.338	0
	0.4	20.551	20.836	21.097	21.177	21.307	1
	0.5	20.757	20.887	21.136	21.181	21.347	0
	0.6	20.693	20.72	21.135	21.197	21.274	2
	0.7	20.67	21.001	21.036	21.257	21.308	1
	0.8	20.717	20.888	21.129	21.134	21.339	1
	0.9	20.581	20.919	21.093	21.274	21.331	0
	1.0	20.704	20.998	20.918	21.204	21.305	0
f_{13}	0.1	7.07e+01	2.62e+02	6.50e+02	8.81e+02	1.38e+03	1
	0.2	9.61e+01	2.89e+02	6.48e+02	1.01e+03	1.24e+03	0
	0.3	8.41e+01	2.13e+02	6.51e+02	7.36e+02	1.18e+03	0
	0.4	1.12e+02	2.74e+02	3.64e+02	7.03e+02	1.27e+03	1
	0.5	7.54e+01	2.16e+02	5.71e+02	9.89e+02	1.31e+03	0
	0.6	1.03e+02	3.61e+02	4.01e+02	8.43e+02	1.10e+03	0
	0.7	9.72e+01	3.17e+02	5.32e+02	6.65e+02	1.27e+03	1
	0.8	9.70e+01	2.00e+02	6.13e+02	8.01e+02	1.17e+03	2
	0.9	1.20e+02	2.73e+02	5.57e+02	8.74e+02	1.37e+03	0
	1.0	7.32e+01	3.05e+02	5.90e+02	7.65e+02	1.36e+03	0
f_{18}	0.1	8.69e+01	2.87e+02	5.66e+02	1.01e+03	1.53e+03	0
	0.2	1.16e+02	3.26e+02	5.73e+02	1.06e+03	1.30e+03	0
	0.3	6.40e+01	2.45e+02	7.05e+02	9.71e+02	1.38e+03	0
	0.4	7.29e+01	3.14e+02	6.35e+02	7.58e+02	1.17e+03	1
	0.5	8.18e+01	3.20e+02	4.83e+02	9.24e+02	1.39e+03	0
	0.6	7.26e+01	2.37e+02	4.79e+02	7.49e+02	1.39e+03	1
	0.7	8.78e+01	1.89e+02	5.33e+02	1.03e+03	1.33e+03	0
	0.8	4.71e+01	2.30e+02	4.76e+02	7.70e+02	1.26e+03	2
	0.9	1.17e+02	1.84e+02	5.05e+02	1.15e+03	1.52e+03	1
	1.0	1.01e+02	3.01e+02	5.91e+02	9.79e+02	1.53e+03	0
f_{23}	0.1	3.07e+03	1.43e+04	9.22e+03	1.48e+04	3.31e+04	0
	0.2	3.62e+03	1.38e+04	9.26e+03	1.56e+04	3.21e+04	0
	0.3	3.30e+03	1.46e+04	1.16e+04	1.39e+04	3.08e+04	0
	0.4	3.46e+03	1.49e+04	1.03e+04	1.49e+04	3.15e+04	0
	0.5	3.59e+03	1.34e+04	1.12e+04	1.25e+04	3.17e+04	0
	0.6	3.06e+03	1.42e+04	8.56e+03	1.07e+04	3.37e+04	2
	0.7	2.50e+03	1.33e+04	1.18e+04	1.30e+04	3.24e+04	1
	0.8	2.18e+03	1.53e+04	1.12e+04	1.35e+04	2.97e+04	2
	0.9	3.60e+03	1.45e+04	1.26e+04	1.77e+04	3.17e+04	0
	1.0	3.26e+03	1.44e+04	1.01e+04	1.39e+04	3.21e+04	0
f_{28}	0.1	1.42e+03	8.47e+02	4.57e+03	5.34e+03	1.22e+04	0
	0.2	1.77e+03	8.75e+02	5.06e+03	4.43e+03	1.11e+04	0
	0.3	1.28e+03	1.95e+03	4.44e+03	4.00e+03	1.19e+04	0
	0.4	1.77e+03	8.75e+02	3.70e+03	3.46e+03	1.07e+04	0
	0.5	1.23e+03	1.75e+03	2.65e+03	7.76e+03	9.86e+03	0
	0.6	1.37e+03	8.87e+02	3.11e+03	5.75e+03	1.12e+04	0
	0.7	3.07e+02	9.13e+02	2.03e+03	3.74e+03	8.66e+03	2
	0.8	1.36e+03	8.39e+02	3.64e+03	3.15e+03	1.06e+04	3
	0.9	1.50e+03	9.82e+02	2.29e+03	4.02e+03	9.93e+03	0
	1.0	1.48e+03	8.64e+02	3.87e+03	4.53e+03	1.12e+04	0

表 4　不同 μ_{max} 值求解函数的最优值(μ_{min}=0.5)

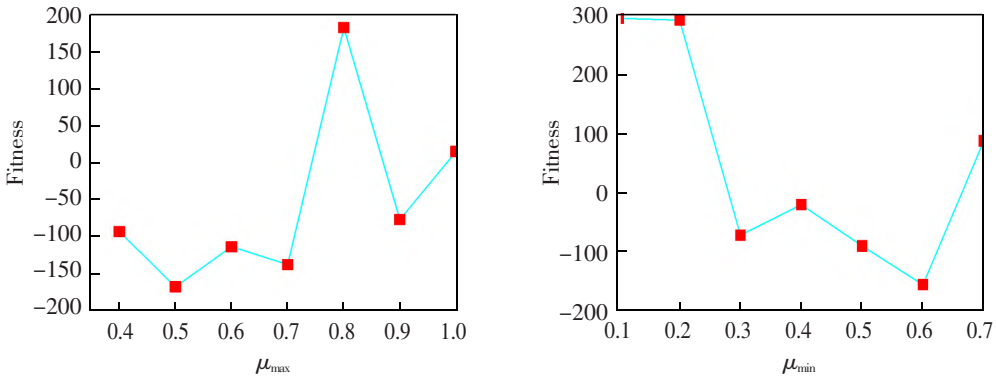
Tab.4　The best value for different μ_{max}(μ_{min}=0.5)

μ_{max}	f_5	f_{10}	f_{15}	f_{20}	f_{25}
0.4	4.73e−01	8.67e+00	1.22e+04	2.13e+01	4.44e+02
0.5	6.46e−01	9.61e+00	1.23e+04	2.19e+01	4.46e+02
0.6	1.83e−01	8.38e+00	1.33e+04	2.08e+01	4.40e+02
0.7	7.81e−01	5.22e+00	1.30e+04	1.94e+01	4.72e+02
0.8	5.33e−01	8.49e+00	1.22e+04	1.95e+01	4.42e+02
0.9	4.73e−01	8.67e+00	1.22e+04	2.13e+01	4.44e+02
1	6.46e−01	9.61e+00	1.23e+04	2.19e+01	4.46e+02

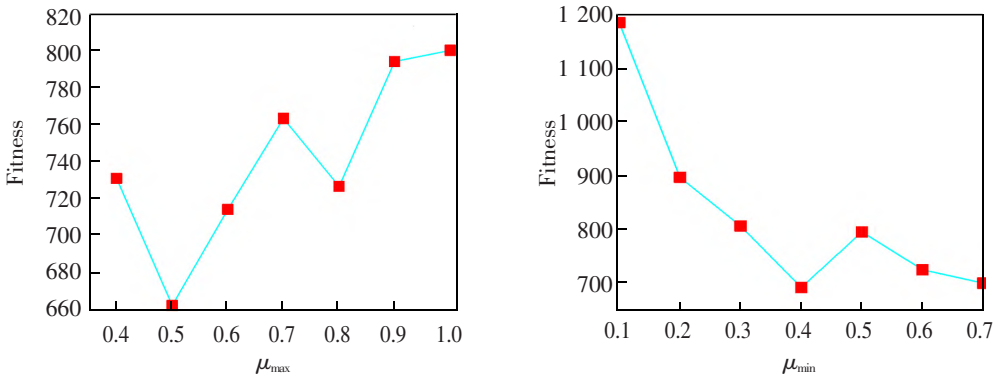
表 5　不同 μ_{min} 值求解函数的最优值(μ_{max}=0.7)

Tab.5　The best value for different μ_{min}(μ_{max}=0.7)

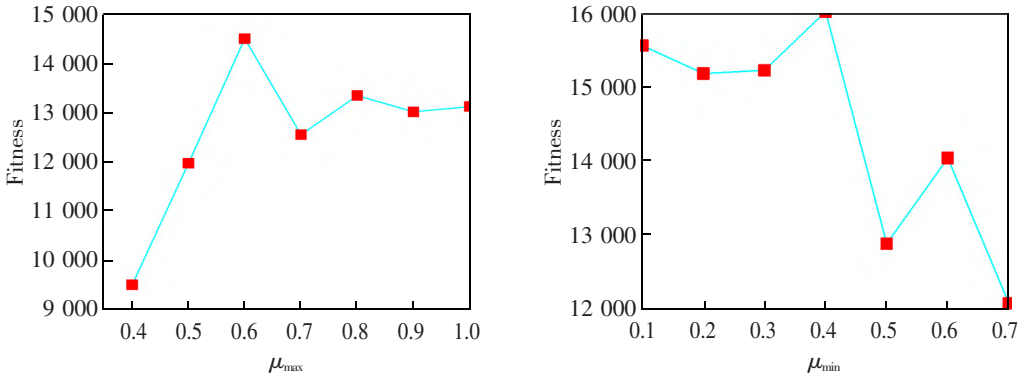
μ_{min}	f_5	f_{10}	f_{15}	f_{20}	f_{25}
0.1	6.08e+02	8.26e+02	1.32e+04	2.25e+01	4.54e+02
0.2	6.35e+00	1.83e+02	1.29e+04	2.24e+01	4.47e+02
0.3	9.61e−01	1.39e+01	1.37e+04	2.21e+01	4.25e+02
0.4	7.62e−01	1.02e+01	1.34e+04	2.11e+01	4.58e+02
0.5	4.85e−01	7.08e+00	1.25e+04	2.16e+01	4.73e+02
0.6	6.63e−01	1.87e+01	1.08e+04	2.01e+01	4.56e+02
0.8	7.31e−01	2.81e+01	1.35e+04	2.08e+01	4.51e+02



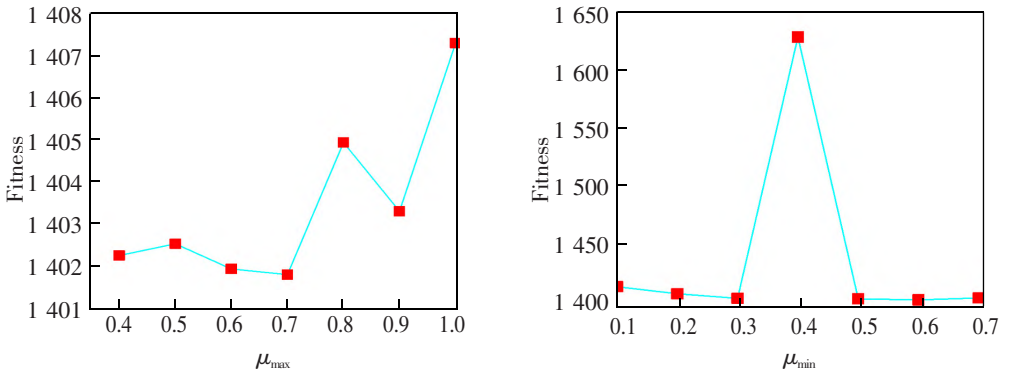
(a) f_{12}



(b) f_{18}



(c) f_{23}



(d) f_{26}

图 3 函数适应度随 μ_{max} 和 μ_{min} 变化的曲线

Fig.3 Curve of fitness as a function of μ_{max} and μ_{min}

3.3 迭代策略影响 VLPSOID 算法的分析

本小节将式(6)的迭代策略 $s(t)$式去除,称算法为 VLPSOD,并与 VLPSOID 的性能比较分析。实验对 28 个基准函数优化计算, 两个算法分别独立运行 10 次,比较两个算法求解函数的平均最优适应度。实验设置种群粒子数为 30 个,函数维度为 30 维,算法迭代次数为 10 000 次。实验结果如表 6 所示。

根据表 6,与 VLPSOID 算法相比,去除迭代策略的 VLPSOD 算法性能出现降低。28 个测试函数,VLPSOD 在 18 个函数中的表现要差于 VLPSOID 算法(全部的 5 个单峰函数,15 个多峰函数的 9 个,8 个复合函数的 4 个), 表明 VLPSOID 要优于 VLPSOD。这说明迭代策略能使算法在前期增强全局搜索能力,后期增强局部搜索能力,促进算法寻优能力。因此,迭代策略能提高算法优化效果和性能。

3.4 维度策略影响 VLPSOID 算法的分析

针对维度策略影响 VLPSOID 算法性能分析,本节将式(6)的维度策略 $h(D)$去除,称算法为

表 6 VLPSOD 算法与 VLPSOID 实验对比

Tab.6 Comparison between VLPSOD and VLPSOID

Function	VLPSOID	VLPSOD
f_1	2.49e+00	4.05e+00
f_2	1.89e+07	1.90e+07
f_3	2.70e+09	4.07e+09
f_4	9.51e+03	9.83e+03
f_5	2.08e+00	3.33e+00
f_6	1.61e+02	1.65e+02
f_7	8.08e+01	8.03e+01
f_8	2.09e+01	2.09e+01
f_9	3.01e+01	3.05e+01
f_{10}	3.94e+01	6.05e+01
f_{11}	5.64e+01	5.13e+01
f_{12}	1.27e+02	1.50e+02
f_{13}	1.92e+02	1.81e+02
f_{14}	1.48e+03	1.65e+03

VLPSOI,并与 VLPSOID 的性能比较分析。实验选择的优化函数是 f_3,f_8,f_{13},f_{18},f_{23},f_{28} 6 种函数，两个算法分别独立运行 10 次，比较两个算法在求解不同维度的函数的平均最优适应度。实验设置种群粒子数 30 个，函数维度分别设置为 30,60,90 维,算法迭代次数为 50 000 次。实验结果如表 7 所示。

观察表 7 可知,与完整 VLPSOID 算法相比,去除维度策略的 VLPSOI 性能出现下降的趋势。能够明显地看出 VLPSOI 在这 6 个测试函数的表现中差于 VLPSOID,这说明,去除了维度策略,算法在高维空间寻优能力变差,导致性能降低。维度策略能提高算法的优化效果和性能。

表 7　VLPSOID 算法与 VLPSOI 实验对比
Tab.7　Comparison between VLPSOID and VLPSOI

Function	Algorithm	Dimension		
		30	60	90
f_3	VLPSOID	5.15e+08	8.23e+09	2.72e+11
	VLPSOI	1.15e+08	8.30e+09	2.96e+10
f_8	VLPSOID	2.08e+01	2.11e+01	2.12e+01
	VLPSOI	2.09e+01	2.11e+01	2.12e+01
f_{13}	VLPSOID	1.47e+02	4.28e+02	7.43e+02
	VLPSOI	1.79e+02	3.07e+02	7.60e+02
f_{18}	VLPSOID	1.73e+02	3.77e+02	9.29e+02
	VLPSOI	1.77e+02	4.24e+02	1.06e+03
f_{23}	VLPSOID	4.26e+03	1.06e+04	1.37e+04
	VLPSOI	5.68e+03	1.01e+04	1.90e+04
f_{28}	VLPSOID	3.00e+02	1.80e+03	1.37e+03
	VLPSOI	3.02e+02	3.46e+03	1.64e+03

3.5　VLPSOID 算法的性能分析

针对分析本文提出的 VLPSOID 算法性能,将其与表 2 列出的 4 种算法在 28 个测试函数上实验比较,每个测试函数独立运行 50 次,得到其平均最优适应度值(Mean)及其标准方差(standard deviation, Std.)。实验参数设置如表 8 所示,实验结果如表 9 所示。

根据表 9,相比其他算法,VLPSOID 在 28 个函数中有 14 个函数均值和方差表现为最好，其中 5 个单峰函数有 4 个,15 个多峰函数中有 6 个,8 个复合函数中有 4 个；其他算法表现最好的 HCPSO 也只在 28 个函数中有 7 个;因此,在相比较的算法中,VLPSOID 的性能为最佳的算法。

表 8　实验的参数设置
Tab.8　Experimental parameters

Population	Dimension	Iteration	Replication number
30	30	150 000	50

虽然在 28 个测试函数中,只有 14 个函数效果优于其他算法，采用弗里德曼检测方法可知,VLPSOID 算法无论是在单峰函数、多峰函数和复合函数的检验值,还是在综合的检验值,在比较的算法中排名最好,说明其综合性能最佳。

为了更好地验证 VLPSOID 算法性能有效性,针对表 9 的实验数据,做 t 检验方法验证。t 检验自由度和显著性水平设置分别为 49 和 0.05，最终结果如表 10 所示。其中,粗体表示本文算法好于对比算法,灰体表示本文算法表现较差。

从表中可以看出，与 PSOSAVL 算法相比,算法性能只有在 f_{14},f_{16},f_{21},f_{22},f_{25},f_{26} 这 6 个函数上处于劣势,其余函数则明显处于优势;而对比 HPSO 算法,可以看出 VLPSOID 算法效果显著;与 HCPSO 算法相比,本文算法在 7 个函数上效果差于对比算法。最后,与 PSO 算法对比可以看出,本文算法只有在 f_8,f_{15},f_{16} 函数上处于劣势。总体结果是,在 28 个函数的对比中，本文算法好于 PSOSAVL、HPSO、HCPSO 和 PSO 的函数个数分别为 22、28、21、25。综上所述,本文算法在对比算法中有一定的竞争能力。

3.6　算法的收敛性分析

为分析算法的收敛状态，采用 VLPSOID 与其他几个算法优化 6 个基准测试函数,演示了每个算法计算测试函数的适应度随迭代变化的曲线,结果如图 4。其中,6 个函数分别是,单峰函数选取一个函数,多峰函数选取 3 个函数以及复合函数选取两个函数。实验参数设置种群维度为 30 维,种群大小为 30 个,迭代次数为 150 000 次。

由图 4(a)可知,VLPSOID 在单峰函数中与其他算法中表现相当。由图 4(b)、图 4(c)和图 4(d)中可知,VLPSOID 算法在多峰函数中表现良好，大约迭代到一万代时,就能找到较好的解,收敛速度快且

表9 基准测试函数的实验结果

Tab.9 Experimental results of function tests

Function	Standard	VLPSOID	PSOSAVL	HPSO	HCPSO	PSO
f_1	Mean	5.35e−06	5.68e−06	2.35e+03	1.44e−01	5.53e+03
	Std.	3.65e−06	5.96e−06	2.82e+02	5.88e−02	5.25e+03
f_2	Mean	8.39e+06	1.02e+07	4.86e+07	1.07e+07	7.66e+07
	Std.	5.76e+06	7.85e+06	9.24e+06	5.89e+06	9.25e+07
f_3	Mean	1.15e+08	2.09e+08	7.89e+09	3.70e+08	3.65e+13
	Std.	1.39e+08	2.87e+08	1.77e+09	6.26e+08	2.01e+14
f_4	Mean	2.77e+02	4.49e+02	1.18e+04	1.24e+02	1.33e+04
	Std.	2.00e+02	2.86e+02	1.83e+03	3.90e+01	2.42e+04
f_5	Mean	7.68e−04	7.73e−04	8.75e+02	2.32e−01	6.13e+03
	Std.	5.87e−04	4.71e−04	1.63e+02	4.43e−02	6.20e+03
f_6	Mean	1.02e+02	1.15e+02	3.30e+02	1.00e+02	6.57e+02
	Std.	3.28e+01	5.36e+01	4.67e+01	3.75e+01	6.03e+02
f_7	Mean	4.52e+01	5.90e+01	8.34e+01	4.23e+01	5.71e+02
	Std.	1.48e+01	1.59e+01	1.20e+01	1.39e+01	1.31e+03
f_8	Mean	2.08e+01	2.08e+01	2.08e+01	2.08e+01	2.08e+01
	Std.	5.42e−02	5.05e−02	4.58e−02	5.02e−02	5.30e−02
f_9	Mean	2.37e+01	2.46e+01	3.67e+01	2.26e+01	2.85e+01
	Std.	3.14e+00	3.10e+00	1.03e+00	2.37e+00	3.10e+00
f_{10}	Mean	8.75e−02	1.33e+00	3.70e+02	1.88e+00	1.14e+03
	Std.	5.62e−02	8.38e+00	6.23e+01	3.23e−01	7.56e+02
f_{11}	Mean	6.59e+00	6.99e+00	2.45e+02	7.82e+01	1.42e+02
	Std.	3.43e+00	3.26e+00	1.28e+01	2.20e+01	7.25e+01
f_{12}	Mean	8.99e+01	1.00e+02	2.44e+02	8.83e+01	2.11e+02
	Std.	3.24e+01	2.95e+01	1.03e+01	2.85e+01	9.99e+01
f_{13}	Mean	1.53e+02	1.64e+02	2.41e+02	1.59e+02	2.55e+02
	Std.	3.05e+01	2.74e+01	1.33e+01	2.77e+01	7.01e+01
f_{14}	Mean	4.29e+02	4.13e+02	6.78e+03	2.45e+03	3.51e+03
	Std.	1.87e+02	1.81e+02	2.18e+02	7.13e+02	3.51e+03
f_{15}	Mean	4.78e+03	5.33e+03	6.80e+03	5.15e+03	4.52e+03
	Std.	8.84e+02	8.84e+02	2.57e+02	1.25e+03	8.89e+02
f_{16}	Mean	1.63e+00	1.61e+00	1.94e+00	1.93e+00	1.05e+00
	Std.	3.55e−01	2.86e−01	2.05e−01	1.75e−01	2.44e−01
f_{17}	Mean	5.32e+01	5.37e+01	3.11e+02	1.88e+02	1.87e+02
	Std.	6.34e+00	8.68e+00	1.59e+01	3.51e+01	1.29e+02
f_{18}	Mean	1.09e+02	1.25e+02	3.10e+02	1.92e+02	2.14e+02
	Std.	2.03e+01	2.38e+01	1.48e+01	3.25e+01	1.34e+02
f_{19}	Mean	4.33e+00	4.84e+00	5.55e+01	1.48e+01	5.86e+04
	Std.	1.78e+00	1.87e+00	1.06e+01	3.16e+00	3.92e+00
f_{20}	Mean	1.11e+01	1.14e+01	1.27e+01	1.11e+01	1.48e+01
	Std.	8.92e−01	1.39e+00	2.26e−01	7.85e−01	7.73e−01
f_{21}	Mean	3.13e+02	3.05e+02	8.71e+02	3.21e+02	5.96e+02
	Std.	9.02e+01	8.63e+01	1.42e+02	7.48e+01	4.76e+02
f_{22}	Mean	4.77e+02	4.45e+02	7.13e+03	2.56e+03	3.34e+03
	Std.	2.48e+02	2.38e+02	2.62e+02	6.13e+02	1.02e+03
f_{23}	Mean	5.03e+03	5.24e+03	7.12e+03	5.45e+03	5.12e+03
	Std.	8.98e+02	8.53e+02	2.61e+02	1.35e+03	9.58e+02
f_{24}	Mean	2.54e+02	2.57e+02	2.88e+02	2.73e+02	3.03e+02
	Std.	1.28e+01	1.27e+01	9.91e+00	8.44e+00	1.32e+01
f_{25}	Mean	2.99e+02	2.96e+02	3.17e+02	2.92e+02	3.15e+02
	Std.	1.86e+01	2.12e+01	4.15e+00	9.01e+00	1.28e+01
f_{26}	Mean	2.00e+02	2.00e+02	2.03e+02	2.00e+02	3.53e+02
	Std.	2.92e−01	1.68e−01	6.39e−01	2.60e−01	6.71e+01
f_{27}	Mean	8.78e+02	8.96e+02	1.04e+03	9.21e+02	1.18e+03
	Std.	1.33e+02	9.57e+01	1.09e+02	6.23e+01	1.16e+02
f_{28}	Mean	3.35e+02	3.61e+02	1.51e+03	4.22e+02	2.84e+03
	Std.	2.85e+02	3.18e+02	8.59e+01	3.84e+02	7.14e+02

表 10　*t*–检验实验结果

Tab.10　The *t*–test experimental results

Function	VLPSOID	PSOSAVL	HPSO	HCPSO	PSO
f_1	*t*–value	**3.62e–01**	**5.90e+01**	**1.73e+01**	**7.44e+00**
	p–value	**7.19e–01**	**3.38e–47**	**1.75e–22**	**1.36e–09**
f_2	*t*–value	**1.63e+00**	**2.51e+01**	**1.89e+00**	**5.27e+00**
	p–value	**1.10e–01**	**1.16e–29**	**6.47e–02**	**3.00e–06**
f_3	*t*–value	**2.02e+00**	**3.08e+01**	**2.81e+00**	**1.28e+00**
	p–value	**4.89e–02**	**1.07e–33**	**7.03e–03**	**2.05e–01**
f_4	*t*–value	**3.19e+00**	**4.43e+01**	–5.54e+00	**3.80e+00**
	p–value	**2.46e–03**	**3.48e–41**	1.00e–06	**4.02e–04**
f_5	*t*–value	**4.97e–02**	**3.79e+01**	**3.71e+01**	**7.00e+00**
	p–value	**9.61e–01**	**6.12e–38**	**1.63e–37**	**6.70e–09**
f_6	*t*–value	**1.28e+00**	**3.10e+01**	–2.31e–01	**6.53e+00**
	p–value	**2.06e–01**	**7.76e–34**	8.18e–01	**3.60e–08**
f_7	*t*–value	**4.60e+00**	**1.48e+01**	–9.48e–01	**2.84e+00**
	p–value	**3.00e–05**	**1.18e–19**	3.48e–01	**6.52e–03**
f_8	*t*–value	**1.96e–01**	**2.79e+00**	**2.11e+00**	–4.21e–01
	p–value	**8.46e–01**	**7.45e–03**	**3.99e–02**	6.76e–01
f_9	*t*–value	**1.28e+00**	**2.75e+01**	–1.97e+00	**7.42e+00**
	p–value	**2.06e–01**	**1.92e–31**	5.47e–02	**1.50e–09**
f_{10}	*t*–value	**1.05e+00**	**4.20e+01**	**3.90e+01**	**1.07e+01**
	p–value	**2.99e–01**	**4.21e–40**	**1.54e–38**	**1.96e–14**
f_{11}	*t*–value	**5.88e–01**	**1.24e+02**	**2.22e+01**	**1.31e+01**
	p–value	**5.59e–01**	**6.07e–63**	**3.39e–27**	**1.24e–17**
f_{12}	*t*–value	**1.50e+00**	**3.07e+01**	–2.69e–01	**8.26e+00**
	p–value	**1.39e–01**	**1.12e–33**	7.89e–01	**7.68e–11**
f_{13}	*t*–value	**1.76e+00**	**1.69e+01**	**1.00e+00**	**9.44e+00**
	p–value	**8.42e–02**	**4.57e–22**	**3.22e–01**	**1.32e–12**
f_{14}	*t*–value	–4.07e–01	**1.52e+02**	**1.94e+01**	**1.81e+01**
	p–value	6.85e–01	**3.64e–67**	**1.12e–24**	**2.47e–23**
f_{15}	*t*–value	**2.97e+00**	**2.97e+00**	**1.77e+00**	–1.58e+00
	p–value	**4.65e–03**	**4.65e–03**	**8.29e–02**	1.21e–01
f_{16}	*t*–value	–3.08e–01	**5.07e+00**	**5.24e+00**	–8.75e+00
	p–value	7.59e–01	**6.00e–06**	**3.00e–06**	1.38e–11
f_{17}	*t*–value	**3.27e–01**	**1.00e+02**	**2.69e+01**	**7.34e+00**
	p–value	**7.45e–01**	**2.34e–58**	**4.94e–31**	**2.00e–09**
f_{18}	*t*–value	**3.57e+00**	**5.87e+01**	**1.41e+01**	**5.61e+00**
	p–value	**8.05e–04**	**4.65e–47**	**8.01e–19**	**9.18e–07**
f_{19}	*t*–value	**1.36e+00**	**3.40e+01**	**1.87e+01**	**3.13e+00**
	p–value	**1.81e–01**	**9.31e–36**	**5.67e–24**	**2.98e–03**
f_{20}	*t*–value	**1.18e+00**	**1.24e+01**	–2.03e–01	**2.15e+01**
	p–value	**2.45e–01**	**8.92e–17**	8.40e–01	**1.28e–26**
f_{21}	*t*–value	–4.93e–01	**2.31e+01**	**4.63e–01**	**4.40e+00**
	p–value	6.24e–01	**5.67e–28**	**6.45e–01**	**5.80e–05**
f_{22}	*t*–value	–6.09e–01	**1.29e+02**	**2.41e+01**	**1.84e+01**
	p–value	5.45e–01	**1.00e–63**	**8.61e–29**	**1.16e–23**
f_{23}	*t*–value	**1.24e+00**	**1.58e+01**	**1.97e+00**	**4.62e–01**
	p–value	**2.21e–01**	**7.99e–21**	**5.46e–02**	**6.46e–01**
f_{24}	*t*–value	**1.20e+00**	**1.63e+01**	**9.27e+00**	**1.93e+01**
	p–value	**2.37e–01**	**2.18e–21**	**2.38e–12**	**1.57e–24**
f_{25}	*t*–value	–6.75e–01	**6.35e+00**	–2.25e+00	**6.06e+00**
	p–value	5.03e–01	**6.75e–08**	2.87e–02	**1.91e–07**
f_{26}	*t*–value	–5.41e–01	**2.63e+01**	**4.23e+00**	**1.61e+01**
	p–value	5.91e–01	**1.56e–30**	**1.02e–04**	**3.50e–21**
f_{27}	*t*–value	**8.60e–01**	**6.81e+00**	**1.94e+00**	**1.12e+01**
	p–value	**3.94e–01**	**1.31e–08**	**5.79e–02**	**3.76e–15**
f_{28}	*t*–value	**4.25e–01**	**2.87e+01**	**1.24e+00**	**2.60e+01**
	p–value	**6.73e–01**	**2.49e–32**	**2.20e–01**	**2.61e–30**
Good		22	28	21	25
Bad		6	0	7	3

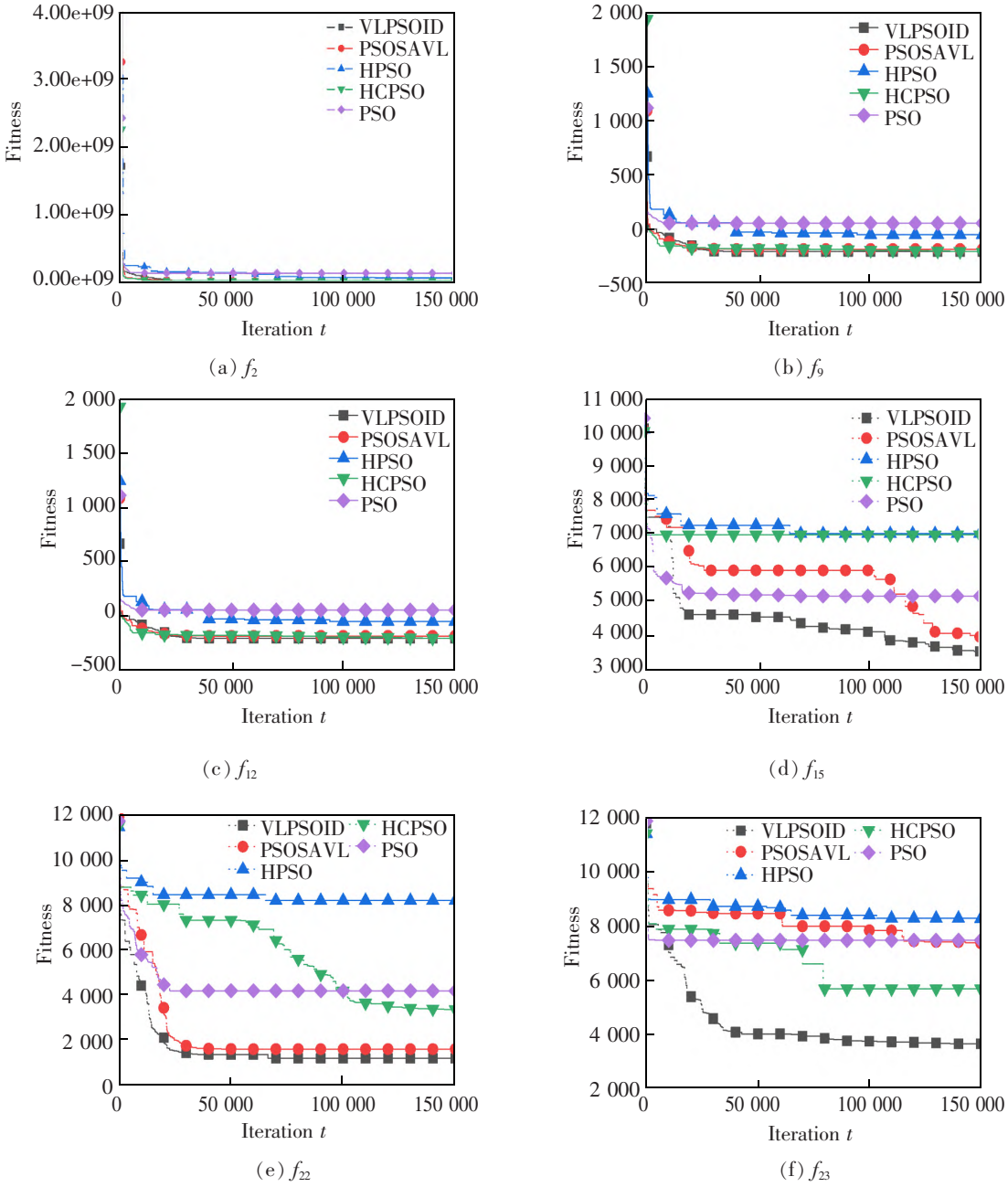


图4 不同算法的函数适应度随迭代变化图

Fig.4 Convergence curves of six test functions

精度较高。从图4(e)和图4(f)能明显得出,对于复合函数,VLPSOID算法在前期下降速度相对于其他算法快,且在后期其精度也能得到保证。综上所述,VLPSOID不管在单峰函数还是多峰函数,其收敛速度较快且精度较高。

3.7 算法的运算时间评估

为了进一步验证VLPSOID在寻优速度上的优势,在相同环境下,将各算法在CEC2013测试集28个函数中分别独立运行10次,记录达到指定精度时算法的平均运行时间,并规定,如果迭代次数超过200 000次后仍未达到指定精度,则用空白表示。最终结果如表11所示。

由表11看出,相较于对比算法,VLPSOID的在5个单峰函数和复合函数中表现稍差,而在多峰函数中,VLPSOID的总体排名较为靠前。在本文中,由于VLPSOID在每一代都需要对粒子的距

表 11　算法达到指定精度的运算时间

Tab.11　Running time of algorithm to achieve specified accuracy

s

Function	Algorithm					Function	Algorithm				
	VLPSOID	PSOSAVL	HCPSO	HPSO	PSO		VLPSOID	PSOSAVL	HCPSO	HPSO	PSO
f_1	18.53	14.48	8.70	63.45		f_{15}	23.40	25.61	43.29	30.05	45.77
f_2	35.30	49.88				f_{16}	21.39	33.79	32.39	35.39	28.41
f_3	26.44	18.08		43.91		f_{17}	21.72	27.83	40.97	38.96	37.13
f_4	66.53	79.48	38.58		41.64	f_{18}	23.53	34.62	24.91	28.87	30.46
f_5	15.89	20.58	16.09			f_{19}	19.59	32.83	32.625	41.13	45.91
f_6	11.66	17.19	7.17	36.34		f_{20}	22.64	23.94	26.84		31.94
f_7	23.11	85.50		15.06		f_{21}	13.23	17.78	24.76	22.84	26.81
f_8	8.19	13.36	10.59	13.03	13.16	f_{22}	57.99	45.17			
f_9	42.78	82.09	74.11		315.03	f_{23}	95.38	117.24		57.96	
f_{10}	12.67	19.12	7.19			f_{24}	34.28	20.38	57.34	13.64	
f_{11}	14.27	25.94	12.83	27.37	30.75	f_{25}	72.13	84	100.26	34.86	
f_{12}	20.28	11.22	26.29	30.06		f_{26}	82.24	48.15	62.98		
f_{13}	16.36	30.80	40.12	72.86	65.44	f_{27}	65.93	76.95	241.52		
f_{14}	22.75	19.05	45.98	33.88	47.25	f_{28}	56.17	87.09	175.57		

离进行计算,从而计算算法种群的状态评估值,因此在此阶段需要额外的计算时间,会导致算法在部分测试函数中没有表现出其寻优速度的优势。但从表 10 结果中可以看出,VLPSOID 在收敛精度有着出色的表现,因此一些时间上的损耗是可以接受的。

4　结论

1) VLPSOID 算法在前期具有较好的全局搜索能力,在后期具有较强的局部搜索能力,算法对问题维度具有扩展性。

2) 在 CEC2013 测试函数上, 算法无论在单峰函数、多峰函数,还是复杂函数上,具有更好性能,更具有适用性,并具有解决不同维度问题的能力。

参考文献:

[1] EBERHART R,KENNEDY J. Proceedings of the Sixth International Symposium on Micro Machine and Human Science,October 4–6,1995[C]. Piscataway:IEEE,1995.

[2] KENNEDY J,EBERHART R. Proceedings of IEEE International Conference on Neural Network,June 27–July 2,1994 [C]. Perth:IEEE,2002.

[3] URRACA R,SODUPE–ORTEGA E,ANTONANZAS J,et al. Evaluation of a novel GA–based methodology for model Structure selection[J]. Neurocomputing,2018,271:9–17.

[4] WU Y,GONG M G,MA W P,et al. High–order graph matching based on ant colony optimization[J]. Neurocomputing,2019,328(7):97–104.

[5] DING S W,LU R R,XI Y,et al. Efficient well placement optimization coupling hybrid objective function with particle swarm optimization algorithm[J]. Applied Soft Computing,2020,95:106511.

[6] KIM T Y,CHO S B. Optimizing CNN–LSTM neural networks with PSO for anomalous query access control[J]. Neurocomputing,2021,456(2):666–677.

[7] 徐尤南,刘志强,陈洁. 基于粒子群算法的码垛机器人时间轨迹优化研究[J]. 华东交通大学学报,2021,38(3):75–81.
XU Y N,LIU Z Q,CHEN J. Time trajectory optimization of palletizing robot based on particle swarm optimization[J]. Journal of East China Jiaotong University,2021,38(3):75–81.

[8] 邓浩,李均利,胡凯,等. 学习邻域参数的粒子群算法[J]. 小型微型计算机系统,2021,42(5):996–1002.
DENG H,LI J L,HU K,et al. Particle swarm optimization with learning neighborhood parameter [J]. Journal of Chinese

Mini-Micro Computer Systems,2021,42(5):996-1002.

[9] 张德华,郝昕源,张妮娜,等. 一种融合优化选择策略的差分粒子群算法[J]. 西安电子科技大学学报,2022,49(2):218-227.
ZHANG D H,HAO X Y,ZHANG N N,et al. PSO-DE algorithm based on the optimal selection strategy[J]. Journal of Xidian University,2022,49(2):218-227.

[10] WANG F,WANG X J,SUN S L. A reinforcement learning level-based particle swarm optimization algorithm for large scale optimization[J]. Information Sciences,2022(602):298-312.

[11] SHI Y,EBERHART R C. Evolutionary programming VII [M]. Berlin:Springer Berlin Heidelberg,1998:591-600.

[12] HELWIG S,BRANKE J. MOSTAGHIM S M. Experimental analysis of bound handling techniques in particle swarm optimization[J]. IEEE Transactions on Evolutionary Computation,2013,17(2):259-271.

[13] JIANG S J,SHI J J,ZHANG Y M,et al. Automatic test data generation based on reduced adaptive particle swarm optimization algorithm[J]. Neurocomputing,2015,158:109-116.

[14] BARRERA J,LVAREZ-BAJO O,FLORES J J,et al. Limiting the velocity in the particle swarm optimization algorithm[J]. Computaciony Sistemas,2016,20(4):635-645.

[15] ADEWUMI A O,ARASOMWAN M A. Improved particle swarm optimizer with dynamically adjusted search space and velocity limits for global optimization[J]. International Journal of Artificial Intelligence Tools,2015,24(5):15020 5174241005.

[16] LI X Z,MAO K Z,LIN F F,et al. Particle swarm optimization with state-based adaptive velocity limit strategy[J]. Neurocomputing,2021,447:64-79.

[17] 蒋晓岫,任佳,顾敏明. 多维度惯性权重衰减混沌化粒子群算法及应用[J]. 仪器仪表学报,2015,36(6):1333-1341.
JIANG X C,REN J,GU M M. Multi-dimensional descending chaotic inertia weight based PSO and its application [J]. Chinese Journal of Scientific Instrument,2015,36(6):1333-1341.

[18] 邓志诚,孙辉,赵嘉,等. 具有动态子空间的随机单维变异粒子群算法[J]. 计算机科学与探索,2020,14(8):1409-1426.
DENG Z C,SUN H,ZHAO J,et al. Stochastic single-dimensional mutated particle swarm optimization with dynamic subspac[J]. Journal of Frontiers of Computer Science & Technology,2020,14(8):1409-1426.

[19] ZHAN Z H,ZHAN J,LI Y,et al. Proceedings of the 6th International Conference on Ant Colony Optimization and Swarm Intelligence,October 12,2009[C]. Scotland:IEEE,2009.

[20] ZHAN Z H,WANG Z J,JIN H,et al. Adaptive distributed differential evolution[J]. IEEE Transactions on Cybernetics,2019,50(11):4633-4647.

[21] LIANG J J,QU B Y,SUGANTHAN P N,et al. Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization[R]. Zhengzhou:Computational Intelligence Laboratory,2013.

[22] 袁小平,蒋硕. 基于分层自主学习的改进粒子群优化算法[J]. 计算机应用,2019,39(1):148-153.
YUAN X P,JIANG S. Improved particle swarm optimization algorithm based on hierarchical autonomous learning [J]. Journal of Computer Applications,2019,39(1):148-153.

[23] 周文峰,梁晓磊,唐可心,等. 具有拓扑时变和搜索扰动的混合粒子群优化算法[J]. 计算机应用,2020,40(7):1913-1918.
ZHOU W F,LIANG X L,TANG K X,et al Hybrid particle swarm optimization algorithm with topological time-varying and search disturbance[J]. Journal of Computer Applications 2020,40(7):1913-1918.

第一作者:王子航(1998—),男,硕士研究生,CCF会员(J8815G)研究方向为智能计算。E-mail:972582408@qq.com。

通信作者:刘建华(1967—),男,教授,博士,硕士生导师,研究方向为智能计算大数据分析物联网技术。E-mail:jhliu@fjnu.edu.cn。

(责任编辑:熊玲玲)