工程科学学报，第 39 卷，第 3 期：462－473，2017 年 3 月
Chinese Journal of Engineering，Vol. 39，No. 3：462－473，March 2017
DOI：10.13374/j.issn2095－9389.2017.03.020；http://journals.ustb.edu.cn

基于逐层演化的群体智能算法优化

张水平，王　碧✉，陈　阳

江西理工大学信息工程学院，赣州 341000
✉通信作者，E-mail：happyeveryday_386@163.com

摘　要　为能彻底解决群体智能算法早熟问题的同时保持原算法主体不变且可与现有优化理论协同优化，在前期仿真实验和理论证明的基础上，提出了一种逐层演化的改进策略．利用在原算法中构建基于搜索空间压缩理论的自适应系统，通过逐层的压缩、选择、再初始化的操作，以包括压缩后搜索空间在内的社会信息作为遗传知识，指导寻优过程，从而实现最终解精度的提升、避免早熟问题的出现．对基准函数进行仿真实验可以看出该策略在提升算法精度，增强后期个体活性方面具有良好的表现．

关键词　群体智能；搜索空间；逐层演化；早熟
分类号　TP301.6

Optimization for swarm intelligence based on layer-by-layer evolution

ZHANG Shui-ping[1)]，*WANG Bi*[1)]✉，*CHEN Yang*[1)]

Faculty of Information Engineering，Jiangxi University of Science and Technology，Ganzhou 341000，China
✉ Corresponding author，E-mail：happyeveryday_386@163.com

ABSTRACT　A layer-by-layer evolution strategy was proposed to deal with the premature convergence of swarm intelligence as a collaborator with other existing researches based on pre-experiments and simple proofs. For promoting the precision of solution and eviting the premature convergence，the self-adaption system was constructed on the basis of the primal algorithm，operations such as compression，selection and re-initialization using the technology of layer-by-layer，and the social information was used including the compressed research space and the optimal solution. The improvements of precision of solution and the vitality of terminal individuals can be found in results of simulation experiments with benchmark functions.

KEY WORDS　swarm intelligence；search space；layer-by-layer；premature convergence

群体智能算法是以多个体为探索主体，利用群组之间的有效社会信息对个体演化产生影响，从而进行最优解探索并保证算法收敛的一类优化算法，如遗传算法（GA）、粒子群优化算法（PSO）和差分进化（DE）等．智能算法体系中，各类算法通过不同的演化策略实现对全局最优解的定位，而在这过程中如何避免局部最优解和早熟问题是二十余年来改进算法所不断研究的重点．然而，由于算法具有收敛性及强随机性，早熟问题并未得到良好的解决，其表现为收敛曲线随迭代次数增加而趋于平滑．这是算法收敛性所带来的必然结果，同时也使得在需要获得更加精确的解时，只能增加种群内个体数量和迭代次数．随之而来的是更加巨大的计算开销．同时，由于现有对各算法的改进优化成果众多，且有良好效果．但策略之间存在明显的非兼容性．如果不考虑对大多数现有优化策略的兼容问题，无疑是对以往研究成果的一种巨大浪费．为此，如何 k 在不改变原有算法收敛性的前提下，尽可能的改变收敛曲线在算法末期保持不变或变化较小的现状，是本文希望解决的主要问题．

在材料领域，layer-by-layer（LBL）技术被用来对纳米薄膜进行组装[1]．通过利用浸泡（immersive）、离心旋转（spin）、喷洒（spray）、电解（electromagnetic）和液

收稿日期：2016－05－09

体流动(fluidic) 等方式对纳米薄膜进行处理. 为获得具有特定功能的薄膜材料,研究者们常常将薄膜依次通过不同的原液并利用上述方法使原液附着在薄膜表面从而形成新的功能材料. 这一思想与群体智能算法中的演化方式[2] 相吻合. 所不同的是,传统的智能算法是将"薄膜"一直浸泡在同一种液体之中.

本文采用搜索空间[3] 作为关键因素,来完成将"薄膜"经过多次浸泡从而获得拥有更加良好表现的智能算法. 搜索空间是对连续函数进行群体智能求解的必要参数,是与求解问题相关的,且会对算法最终输出解精度产生影响. 当"薄膜"每一次与"液体"充分接触后,将对其进行清洗,去除多余的"附着物". 之后再次与"液体"进行接触,依次循环下去. 直到获得预期的结果. "薄膜"便是我们最开始的搜索空间,也可以认为是当前的全局最优解; "液体"则是各算法的进化策略. 此外,还需要对"充分接触"所代表的迭代次数以及清洗策略做出具体定义.

为了将该思想进一步阐述清楚,文章在第 1 章将会对粒子群优化算法做一个简短的介绍. 在第 2 章中则介绍了相关准备工作,从基本实验中展示出逐层演化模型的可行性. 对于逐层演化在粒子群中的实现及其理论证明则在第 3 章中进行说明. 最后,作为结论,在第 4 章中对文中思想优势与不足做出了概述.

1 研究背景

1.1 群体智能与搜索空间压缩

群体智能算法是指以多个体为探索主体,利用群组之间的有效社会信息对个体演化产生影响,从而进行最优解探索并保证收敛的一类优化算法. 算法的输入为搜索空间与目标函数,输出则为搜寻得到的最优解. 在过去的二十余年中,这类优化算法得到了长足地发展,形成了如遗传算法(GA) 、粒子群算法(PSO) 和差分进化(DE) ,并广泛地应用于众多研究与实践领域. 以粒子群算法为例,位置更新是粒子群算法进化迭代的核心. 在本文中所提及的标准粒子群算法[4-5] ,采用下式来更新位置和速度:

$$\begin{cases} v^{t+1} = \chi\left(\omega v^t + c_1 r_1 (p_g - x) + c_2 r_2 (g - x)\right), \\ x^{t+1} = x^t + v^{t+1}. \end{cases} \tag{1}$$

式中,v 为速度,ω 为权重,c_1、C_2 为学习因子,x 为当前位置,p_g、g 分别为该粒子自身的历史最优位置和当前已知的全局最优解,t 为迭代次数,r_1,$r_2 \in (0,1)$ 为随机数,χ 为收缩因子. 在这一阶段,主要的优化策略包括,位置更新和边界约束. 其中位置更新部分又可分为: 参数优化[6-9] 、公式更新[10-11] 、拓扑结构[12-13] 改进. 这些优化策略是粒子群算法所特有的. 而更具通用性的则是边界约束[14-15] ,然而,在其他群体智能算法实施过程中,由于进化策略的不同,并不一定会出现个体越界的情况出现,因此,这一策略依旧仅在粒子群算法中存在广泛应用研究. 为了构建普适性更好的优化策略,这些传统粒子群中的优化方法将作为参考依据而非主要研究对象. 搜索空间对于绝大多数智能算法而言,与目标函数一样,是作为基本输入信息而存在的. 搜索空间压缩策略是指,通过以某一基准点(策略不同,所选取的点也不尽相同) 为中心,并完成对当前搜索空间的压缩,以此来提升算法效率的一种优化策略. 公式(2) 和公式(3) 在文献[16-17] 中被提出. 不难看出二者都是通过计算粒子之间的距离来判断种群的收敛程度. 除此之外,也有利用实验总结[18] ,得出种群收敛的大致节点. 这些研究在完成种群多样性判别后,都不约而同的以此作为标准,用来完成对粒子群速度更新公式或权重的控制. 而本文在后续论述中将公式(2) ~(3) 作为种群多样性判别标准进行对比讨论.

$$d(P) = \frac{1}{P \cdot K} \sum_{i=1}^{P} \sqrt{\sum_{j=1}^{D} (x_{ij} - \bar{x_j})^2}, \tag{2}$$

$$d(i) = \frac{1}{P-1} \sum_{j=1, j\neq i}^{P} \sqrt{\sum_{k=1}^{D} (x_{i,k} - x_{j,k})^2}. \tag{3}$$

其中,P、D 和 K 分别是种群中个体数量、目标问题维度和搜索空间边界最大距离.

1.2 层层组装技术

层层组装技术[1] 是指在构建特定功能材料的时候,采用的一种处理工艺. 其具体流程可由图 1 展示. 研究人员首先将一块薄膜介质作为基础,浸入功能溶

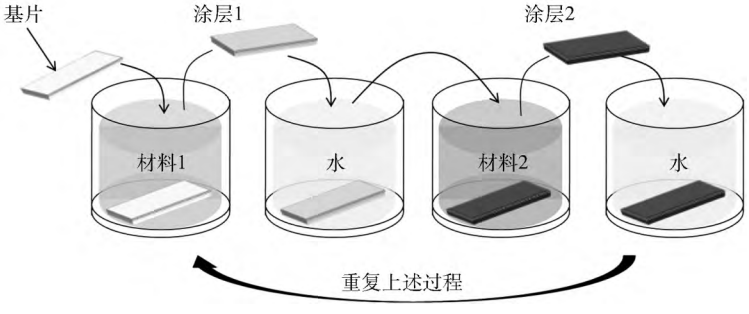

图 1 层层组装浸泡工艺流程示意图[1]

Fig. 1 Layer-by-layer assembly technologies[1]

液中. 等待一段时间至薄膜表面附着粒子后用清水洗掉多余的. 之后再浸入另一功能溶液中，如此往复，直至获得特定功能的薄膜. 若将该技术应用于智能算法之中，则主要问题在于切换不同溶液时，如何确定种群所需携带的社会信息. 虽然在整个寻优过程中，看似只有已知全局最优点在对种群进化产生较大影响. 但在实际过程中，整个群体内的个体对于算法而言都是至关重要的. 为此，当出现分层技术时如何在切换不同层的过程中尽可能保证社会信息中的有效部分是十分困难的；而不对信息进行删减，则等同于传统的求解过程，无法体现改进效果. 为此，则需要选择具有良好通用性、可继承性的社会信息作为遗传要素. 而搜索空间无疑是一个良好的选择.

2　相关问题研究

张水平和王碧[3]对粒子群算法可利用搜索空间压缩策略进行优化的基本条件做出了简单的描述. 最为关键的是需要被优化的算法具有良好的收敛性. 而收敛性的存在，可以为后续的压缩搜索空间提供良好的指导. 推广开来，与粒子群算法一般具有收敛性的群体智能算法对此优化策略也具有潜在可适用性. 构建压缩搜索空间自适应系统的关键是拟定良好的收敛状态判别条件. 此判别条件需要具有随着迭代次数增加而趋于稳定的特定；同时还需要在时间点上与收敛曲线拐点相一致. 为了选取合适的收敛判断，本文将对现有种群多样性判别曲线、最大迭代次数与收敛点的相关性问题进行实验研究. 实验设计如下：

(1) 利用标准粒子群算法，式(1)，作为核心演化策略，并选取固定权值、随机权值[19]、线性权重[20]、非线性权重[21](后分别记为 $w_1 \sim w_4$) 作为对比参数；

(2) 依次增加种群中粒子数量(由 20 增加到 200) 而其他条件保持不变(维度 $D=30$). 对于任意权重策略与种群数量的组合重复进行 50 次实验并记录迭代信息；

(3) 选取 Ackley 函数作为测试函数，进行寻优测试；

(4) 为防止出现由于粒子数量增加使得解精度提升、造成数据比对不便的原因，对最终结果进行归一化处理，形成最终结果.

2.1　种群多样性判别实验对比

策略 1 为式(2) 所示的判别策略[17]，策略 2 为式(4) 所示判别策略[3]，将实验中各组(按粒子数量分类) 实验中粒子在迭代过程中的具体位置作为输入，进行计算两个策略在迭代过程中的适应度，并将所得适应度归一化处理，获得如图 2 和图 3 所示曲线图. 由于在本文中需要稳定的判别标准作为空间压缩的重要指标，而文献 [16] 中的策略正如其文中所述一般，具有很强的随机性，并不能很好的作为指示标准，故在此处不对其进行展开讨论.

$$d(t)=\frac{\sum_{i=1}^{P}\left(\|x_i,g\|\leqslant\beta\|b_l,b_u\|\right)}{P}. \tag{4}$$

式中，g 为当前迭代中已知最优解；b_l、b_u 分别为搜索空间的上下限；$\beta=1\times10^{-6}$ 为阈值，与 $\|b_l,b_u\|$ 一同构成领域的判别标准.

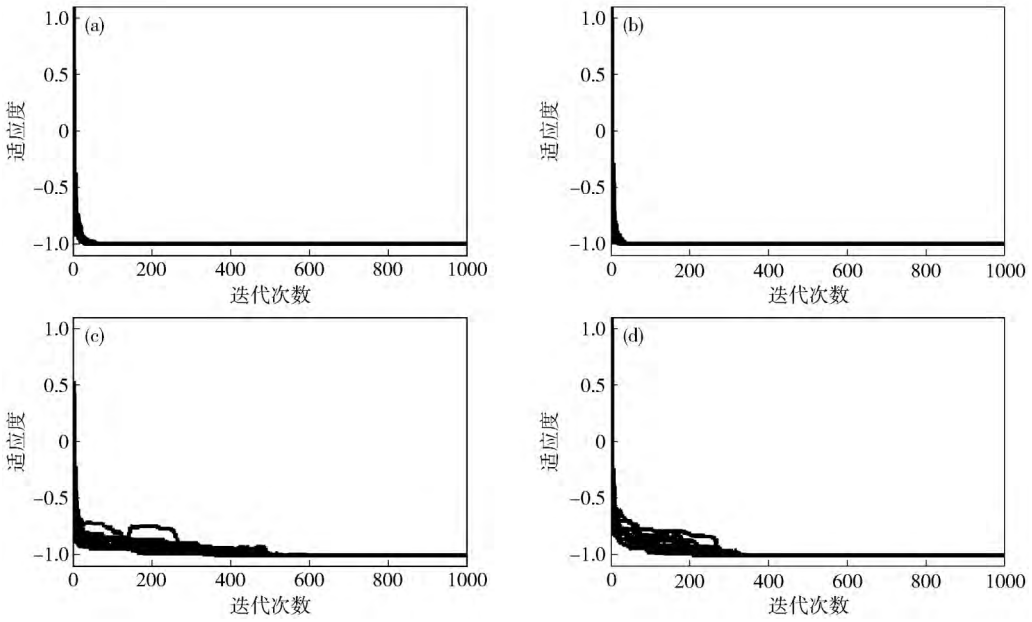

图 2　策略 1 种群多样性曲线 (Ackley). (a) 固定权值；(b) 随机权值；(c) 线性权重；(d) 非线性权重

Fig. 2　Illustrating the swarms diversity with strategy 1 (Ackley): (a) constant-weight; (b) random-weight; (c) linear-weight; (d) nonlinear-weight

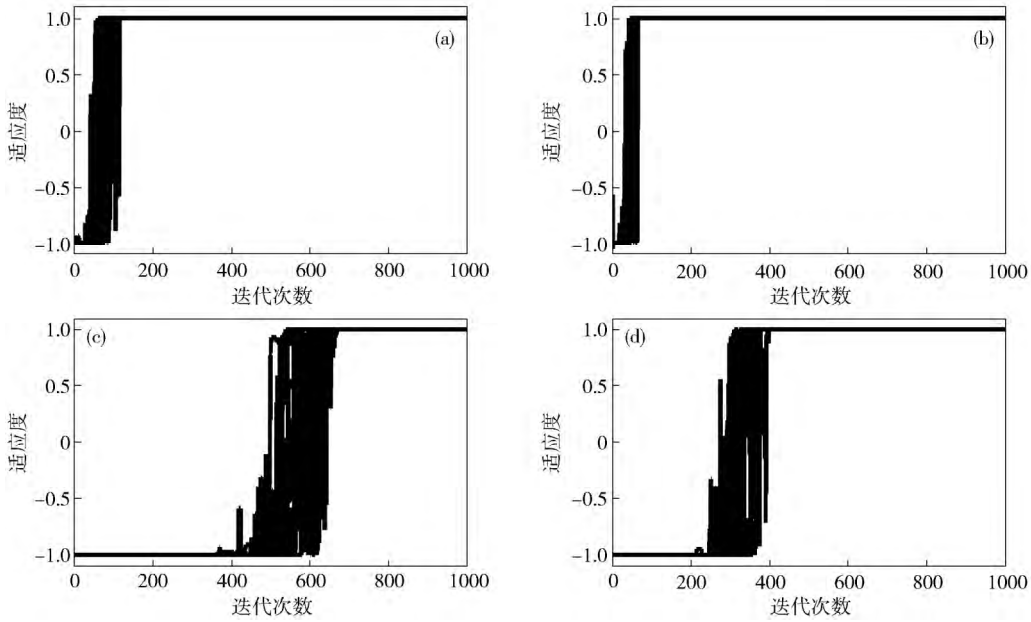

图 3　策略 2 种群多样性曲线 (Ackley) . (a) 固定权值; (b) 随机权值; (c) 线性权重; (d) 非线性权重

Fig. 3　Illustrating the swarms diversity with strategy 2 (Ackley) : (a) constant-weight; (b) random-weight; (c) linear-weight; (d) nonlinear-weight

如图 2 所示,策略 1 所生成的曲线可以清楚展示出判断标准所具有的稳定性,且各组种群粒子数量不同的实验曲线趋势一致,由此可知,该策略在稳定性这一点上是可以应用于后续研究的. 同样的性质也在策略 2 中得到体现,如图 3 所示. 策略 2 在收敛过程中的取值并不稳定,如图 3 中的黑色部分是由于取值的高低变化而造成的. 相对来说,策略 1 更加的平稳一些,基本保持非增的曲线. 由式(4) 可知,策略 2 计算获得的值 d 应是百分数. 在实际应用中,则需要设置一个成熟阈值 γ,其数值可以与种群清洗保留比例 α 相同,使得当满足 $d \geqslant \gamma$ 时,触发搜索空间压缩,为保证粒子的收敛状态,则 γ 应为较大值. 在此前提下,策略 2 的触发时间基本保证在策略 1 趋于稳定后. 这就使得采用策略 2 作为判别标准时,可能造成计算资源的浪费,又或者采用策略 1 作为判别标准时,可能整个种群尚未完全进入稳态. 为此,在后续实验中将会分别采用这两种策略对算法进行改进并对比优化. 其中,策略 1 将会设置较高的阈值,使得自适应系统在曲线近似收敛的情况下触发; 策略 2 则会在最先到达指定阈值的时候触发自适应系统,完成空间压缩,以期得到一个良好的判别标准来解决种群多样性的判别问题.

2.2　搜索空间与最终解精度

为进一步说明搜索空间大小与最终输出解精度的关系,利用式(5) 设计实验对比,使得搜索空间范围随着 k 值增大而逐渐减小. 选择函数 Ackley (f_4) 作为测试函数,在保证除搜索空间外其他条件不变的情况下,对 4 种不同的权重策略进行 50 次重复实验,取平均值并依此绘制曲线. 获得当适应度函数恒定但搜索空间规模不同的情况下输出解精度的变化,如图 4 所示. 可以清楚的看出,当搜索空间规模下降时,最终解的精度也会随之提升,即解精度与搜索空间规模成负相关,如式(6) 所示,其中 G_{real} 和 S 分别是理论最优解和搜索空间规模.

$$[b_l, b_u]^* = [b_l, b_u] / 10^k, \tag{5}$$

$$\| g - G_{\text{real}} \| \propto \frac{1}{S}. \tag{6}$$

由此可知,如果能保证在压缩搜索空间过程中,全局最优解并不会被丢弃,则压缩搜索空间的次数越多、搜索空间将越小的情况下,获得的解精度将会大大提升. 为此,后续研究均以这一特性为准则: 在尽可能保证不丢弃最优解的情况下,取得尽可能多的压缩次数和压缩幅度,以确定一个极小的搜索空间. 同时,策略 2 所需的计算复杂度为 $O(P)$ 而策略 1 的则为 $O(P^2)$,策略 2 所需的计算资源更少一些.

3　逐层演化的粒子群算法

3.1　理论证明

为了进一步说明搜索空间压缩对于命中高适应度解概率的提升作用,如图 5 所示,将 2 维搜索空间划分为 m 个较大空间(灰色所示) 并将每个较大空间再次划分为 n 个较小的空间. 不难得出,较小的空间在连续函数的情况下会包含更为精确的解. 假设 A 和 B 分别为较大空间和较小空间内搜寻的初始位置,O 为最优解所在位置.

假设粒子在空间内移动过程为齐次马尔科夫过

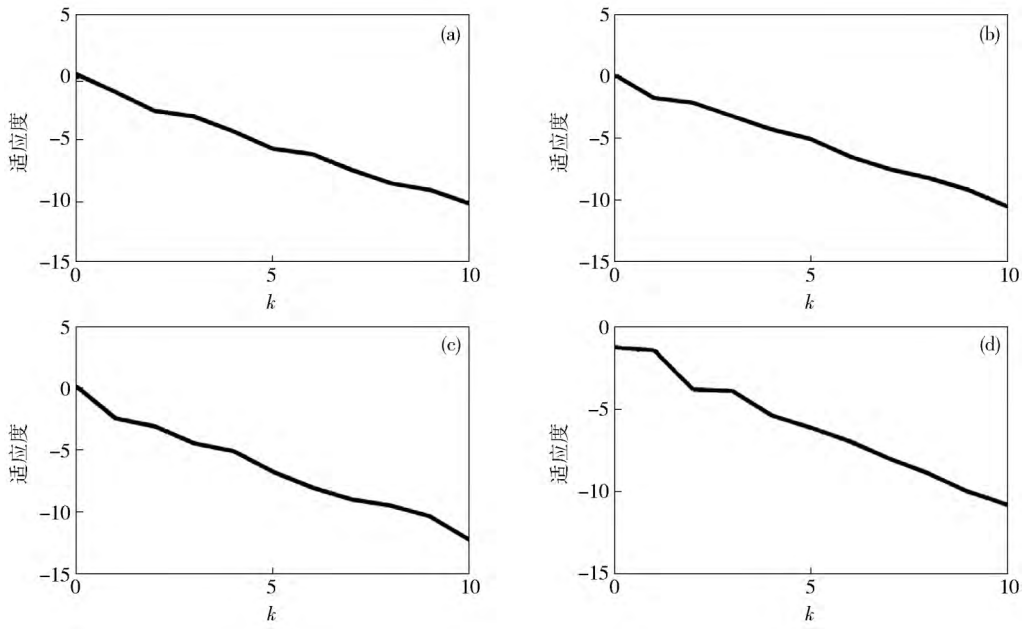

图 4　搜索空间大小与最终解精度相关函数图．(a) 固定权重；(b) 随机权重；(c) 线性权重；(d) 非线性权重

Fig. 4　Relations between the precision of the solution and the scale of the search space: (a) constant-weight; (b) random-weight; (c) linear-weight; (d) nonlinear-weight

程，一步概率转移矩阵如下式所示，为均匀分布.

$$\boldsymbol{p}_{m\times n}=\begin{pmatrix}\frac{1}{mn} & \cdots & \frac{1}{mn}\\ \vdots & \ddots & \vdots\\ \frac{1}{mn} & \cdots & \frac{1}{mn}\end{pmatrix}. \tag{7}$$

若第 $t+1$ 次首次命中最优解邻域 O（图 5 中所示），则分别计算无压缩空间机制即随机模式和压缩空间机制下的概率 p、q.

$$\begin{cases}p=\dfrac{1}{mn}\cdot\left(\dfrac{mn-1}{mn}\right)^{t},\\ q=\dfrac{1}{mn}\sum\limits_{i=0}^{t}\left[\left(\dfrac{m-1}{m}\right)^{i}\left(\dfrac{n-1}{n}\right)^{t-i}\right],\\ p-q=\dfrac{1}{mn}\cdot\left(\dfrac{mn-1}{mn}\right)^{t}-\dfrac{1}{mn}\sum\limits_{i=0}^{t}\left[\left(\dfrac{m-1}{m}\right)^{i}\left(\dfrac{n-1}{n}\right)^{t-i}\right].\end{cases} \tag{8}$$

现令 $m=n\to\infty$，则公式(8)可化简如下：

$$p-q=\frac{1}{mn}\left[\left(\frac{mn-1}{mn}\right)^{t}-\sum_{i=0}^{t}\left[\left(\frac{m-1}{m}\right)^{i}\left(\frac{n-1}{n}\right)^{t-i}\right]\right]=\frac{1}{m^{2}}\left(\frac{m-1}{m}\right)^{t}\left[\left(\frac{m+1}{m}\right)^{t}-t\right].$$

已知 $\dfrac{1}{m^{2}}\left(\dfrac{m-1}{m}\right)^{t}>0$，设：

$$f(t)=\left(\frac{m+1}{m}\right)^{t}-t,$$

$$f'(t)=\left(\frac{m+1}{m}\right)^{t}\ln\left(\frac{m+1}{m}\right)-1<0\ (m\to\infty).$$

即 $f(t)$ 是 t 的单调减函数，由此可知

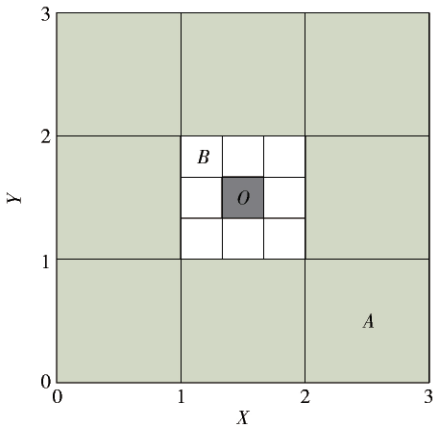

图 5　划分搜索空间下的命中概率示意图

Fig. 5　Schematic diagram of the hitting probability based on the divided search space

$$f(t)\leqslant f(1)=0. \tag{9}$$

因此

$$p-q=\frac{1}{m^{2}}\left(\frac{m-1}{m}\right)^{t}f(t)\leqslant 0. \tag{10}$$

由此可知，通过压缩搜索空间是可以在一定程度上提升命中更优解的概率. 同时这一改变在保证不会造成最优解丢失的情况下，随着压缩次数增加而提升. 加之上文中的具体实验，可以证明，搜索空间压缩策略是切实有效的. 同时由于逐层演化思想的引入，可以有效的组织各层之间对于所面临的不同问题，如是主要进行全局探索还是进行局部搜索等，通过调整不同层内的进化策略进而到达提升解精度的目的.

3.2 算法基本思想

多分组的粒子群在不同区域的搜索空间内,经过 t 步迭代探索局部最优解后保留大部分适应度高的局部最优解所在空间,并对其进行压缩; 而其他空间则抛弃,是该策略的主要思想.

(1) 首次初始化时将搜索空间 S 划分为多个子空间 S_i,并随机分配给各个独立的粒子群 p_i.

若按照某一规则对所有维度进行划分,则划分的子空间数随着维度增加而逐渐增加,以简单的对半划分为例,子空间的个数为 2^D,其中 D 为问题维度. 为此,在后续实验中,采用的是一种简单的随机划分空间策略,假设所需划分的子空间数量为 $e = e_w \cdot e_h$,其中 e_w 和 e_h 为两个自然数,则用公式(11)进行划分子空间. 其中,$i \leqslant e$,m,$n \leqslant D$,$m \neq n$ 为随机数,即随机挑选两个维度进行多项划分. 若需要划分的维度为其他值,则可以适当调整即可.

$$B_{U_i} = b_u \cdot \lceil y_p \rceil,\ y_{p\neq m,n} = 1,\ y_m = \frac{\lceil i/e_h \rceil}{e_w},$$
$$y_n = \frac{\mathrm{mod}(i, e_h) + 1}{e_h}. \tag{11}$$

(2) 在浸泡阶段独立的粒子群 p_i 进行空间探索,除粒子越过最初的搜索空间 S 外,不对各子空间进行越界处理.

(3) 清洗过程则仅仅保留寻优效果较好的粒子群组及其周围空间(已压缩),而其他搜索空间则被丢弃.

(4) 压缩过程将(3)所得空间进行压缩,保存新的空间大小.

(5) 再次初始化并不需要将剩余的空间再次划分为多个子空间,而是将被淘汰的群体中的粒子分配到其他群组中去. 除首次初始化外,整个空间中的群组数量是随清洗次数递减的. 如此直至最后输出.

为实现这一设想,需要进行下列设置:

(1) 粒子群组的递减策略: 保留比例 $\alpha = 0.7$,该值不固定.

保留机制采用 best-first 原则,保留最终适应度较高的一部分粒子群组. 每次淘汰群组数量由下式计算可得,

$$R^* = \max(1, R \cdot \alpha). \tag{12}$$

式中,R 为群组数量. 通过向上取整保证至少存在一组粒子群在进行寻优探索.

(2) 搜索空间压缩策略: 压缩比例 $\beta = 0.5$,可根据清洗次数而定.

由前文已然得出结论,更小的搜索空间可以提供更为精确的解,且算法收敛时粒子依附于当前已知最优解 g. 为此,可以导出搜索空间压缩式(13). 其中 L 为最小搜索空间距离限制,当确定不会发生最优解丢弃情况时,可以设为零,abs( ) 为绝对值函数.

$$\begin{cases} b_l^* = g - \max[\mathrm{abs}(g - b_l) \cdot \beta, L], \\ b_u^* = g + \max[\mathrm{abs}(g - b_u) \cdot \beta, L]. \end{cases} \tag{13}$$

(3) 再次初始化.

再次初始化时,可以将被淘汰群体中的粒子加入保留群组中,进一步提升解精度; 同样也可以直接放弃这部分粒子,从而提升算法的效率. 如文献中提出的策略一致,每次再初始化时需要保存必要的社会信息,在粒子群算法中即保存已知最优解 g,而初始化粒子则尽可能的进行均匀分配,也可以利用文献中提及的混沌遍历. 通过再初始化可以有效地提升粒子的“活力”,增加群内粒子多样性,改善依然趋于收敛的现状. 是利用搜索空间压缩进行优化的必要手段之一. 其伪代码实现如表 1 所示.

现对 2 维多峰值函数进行实验,并绘制各种群搜索空间范围,获得如图 6 所示结果,用来展示本文所提策略的优化细节. 其中,横纵坐标表示 2 维搜索空间范围,灰色黑边矩形表示一个独立种群的搜索子空间. 随着清洗过程的进行,子空间数量在逐渐减少,同时目标空间范围也越来越小,即所搜索的空间范围精度提高. 这一演化过程与之前的设想基本吻合,可用于提升算法精度.

3.3 实验设计与结果

选取了 11 个经典基准函数进行测试,在此处不对表达式进行详细展开[22],如表 2 所示. 其中 o 为偏移量,此处设定 $o = 10^D$,仅作偏移,可以设置为其他值. 函数 f_1、f_2 和 f_5 及其转换形式均为单峰函数,而其他则为多峰值函数.

由于逐层演化的策略是将每一个独立的群体智能算法看作一次浸泡过程而不去改变原有算法的结构流程,该策略的本质是如何去除在演化过程中无效或者效果较差的迭代、降低计算资源的浪费. 故此,一个良好的原算法由于具有良好的寻优能力,则在应用时可以增加裁剪搜索空间的力度,极大程度上缩小搜索空间范围,从而获得更高精度的解; 而相对于搜索能力较差的原算法,则需要注意搜索空间压缩力度,防止出现丢弃实际最优解的可能. 考虑到这一特点,加之粒子群算法优化策略众多且未有较为权威的标准,为此,选取 SPSO[5] 不仅可以保证原算法的易实现性. 同时也由于其常作为优化对照方案,查询效率低于大多数改进算法,可以作为一个标尺,展现出本文策略的提升效果. 而在应对更为精确的优化策略时,只需要相应的调整搜索空间压缩力度便可获得进一步提升.

利用本文所提及的用于种群多样性判别策略对 SPSO[5] 进行搭载测试,由于关于粒子群的具体更新策略并未发生变动,故此处不做特殊说明. 具体设置如下:

表 1　LBL–PSO 算法实现

Table 1　LBL–PSO approach

参数 T 为最大迭代次数；P 为初始状态每个种群的粒子个数；D 为问题维度；p_i 为第 $i(0\leqslant i\leqslant R)$ 组粒子群信息；R 为初始化种群数量；$d(p_i)$ 为第 i 个种群成熟度，对应文中策略 1 和 2；α 为种群清洗保留比例、β 空间压缩因子、γ 为成熟阈值，小于 1 的固定常数.	
1	随机初始化每组粒子群的初始信息 P_i；
2	For $t=1:T$
3	$c=0$//计数变量，用来记录，当种群到达清洗阈值，则加一
4	For each p_i
5	If $\gamma\leqslant d(p_i)$
6	$c++$
7	End
8	End
9	If $\gamma\leqslant\frac{c}{PR}$
10	按照式(12)保留优秀种群总数 R^*；
11	依据每个种群内的最优适应度进行排序，保留前 R^* 个种群；
12	按照式(13)计算每个种群对应的搜索空间 $[b_l,b_u]^*$；
13	根据新的搜索空间对各种群进行位置初始化，但需要保留最优信息，即遗传信息；
14	$R=R^*$；$[b_l,b_u]=[b_l,b_u]^*$；
15	Else
16	For each p_i
17	利用原群体智能策略进行粒子位置更新(此处可以为任意具有收敛性的群体智能策略)；
18	End
19	End
20	$t++$
21	End

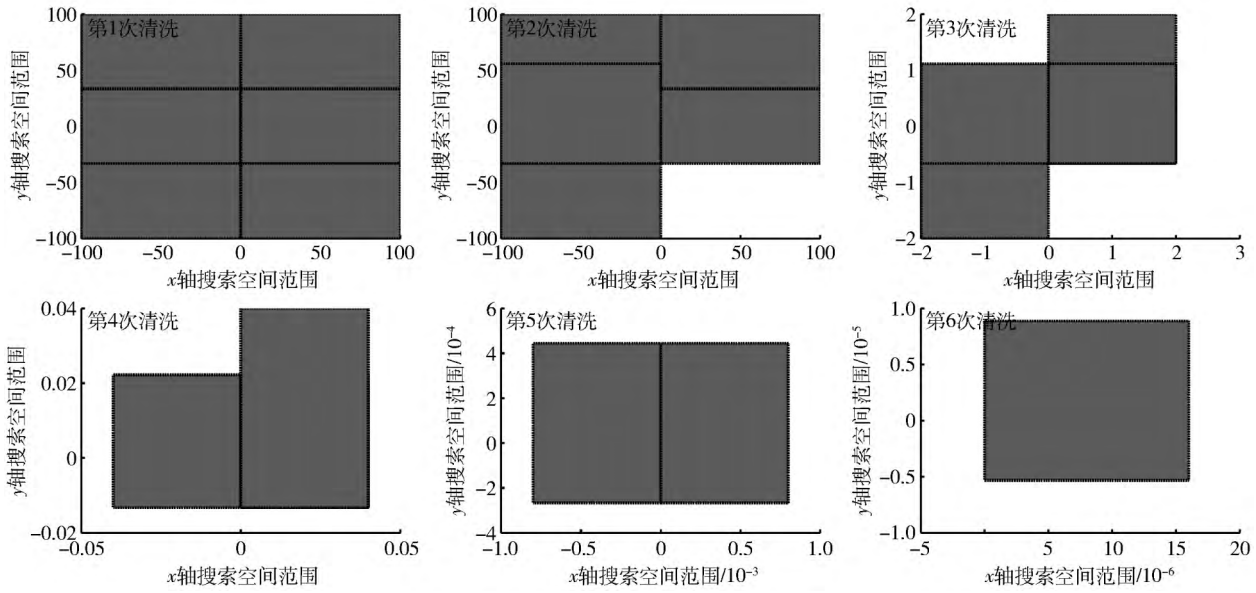

图 6　逐层演化效果示意图

Fig. 6　Diagram of LBL

$T=1000$；$P=10$；$R=5$；$\alpha=\gamma=0.7$；$\beta=0.5$；$L=1\times10^{-5}$，算法总参与计算的粒子数为 $PR=50$；α 作为种群清洗保留比例，可以辅助控制压缩次数，假设实验需要 t 次以上的压缩，则只需要保证 $R\alpha^t\leqslant1$ 即可；β 为

表 2 基准函数表

Table 2 Benchmark functions used in this paper

名称	取值范围	极值点	最优解
f_1: Sphere	$[-100,100]^D$	$g=0^D$	$f(g)=0$
f_2: Elliptic	$[-100,100]^D$	$g=0^D$	$f(g)=0$
f_3: Rastrigin	$[-5,5]^D$	$g=0^D$	$f(g)=0$
f_4: Ackley	$[-32,32]^D$	$g=0^D$	$f(g)=0$
f_5: Schwefel	$[-100,100]^D$	$g=0^D$	$f(g)=0$
f_6: Rosenbrock	$[-100,100]^D$	$g=1^D$	$f(g)=0$
f_7: Sft_Elliptic	$[-90,110]^D$	$g^*=o+g=10^D$	$f(g^*)=0$
f_8: Sft_Rastrigin	$[5,15]^D$	$g^*=o+g=10^D$	$f(g^*)=0$
f_9: Sft_Ackley	$[-22,42]^D$	$g^*=o+g=10^D$	$f(g^*)=0$
f_{10}: Sft_Schwefel	$[-90,110]^D$	$g^*=o+g=10^D$	$f(g^*)=0$
f_{11}: Sft_Rosenbrock	$[-90,110]^D$	$g^*=o+g=11^D$	$f(g^*)=0$

空间压缩比例,依据预期压缩次数而定,可以进行调整; 当所有种群中成熟种群数量超过一定比例 γ 后,表示整体成熟即浸泡完成,进行清洗; L 仅做阈值使用,一般实验过程中不会触及,若最大迭代次数提高,则可以降低该值.

图 7 记录了在权重为固定值,问题维度 $D=10$ 和问题维度 $D=30$ 的条件下,原粒子群算法与分别添加种群多样性策略 1 和 2 作为收敛性判别条件进而构建的逐层演化系统(图例中 10 和 30 为纬度,S1 和 S2 对应策略 1 和策略 2) ,各独立进行 100 次重复实验,取平均值获得的收敛曲线,其中,横轴为迭代次数, 和纵轴为最优解取对数后的值. 为了验证实验的公平性,标准粒子群算法和逐层演化策略搭载后的粒子群算法使用了相同数量的粒子,虽然这些粒子会分布到几个可以并行的独立种群中去,在一定程度上影响了改进后算法在初期的寻优效果,但也证实了改进策略的有效性.

以图 7(a) 为例,可以看出无论是在 10 维或者 30 维问题的处理时,逐层演化策略都能对结果的最终精度有一定提升. 然而,这一策略最主要的提升还是对存在于传统粒子群算法中早熟问题的处理. 可以看出两种不同的多样性判别标准所带来的曲线虽有一定的差异,但却都未出现停滞进化的现象. 同样的情况也可以在图 7(b) 中得到体现. 而在 7(c) 中则可以清楚的发现在 10 维问题求解的改进曲线上,存在明显的精度提升过程. 这一现象的产生原因是当算法在一段时间内没有获得新的最优解,则大多数个体由于进化策略的影响,必将收敛于最优解附近,从而达到种群多样新判断的阈值,进而触发空间压缩策略.

在图 7(d) 中同样可以发现曲线存在平缓后出现快速下滑的情况,这在其他的粒子群改进策略中并不多见. 这一现象的存在很好的克服了早熟问题,也可以在图 7(e) 、7(g) 、7(h) 和(j) 得到体现. 但是,正如之前所提及的,该策略对空间压缩中心点的定位具有较强的依赖性. 如图 7(f) 、7(i) 和 7(k) 所示,可以清楚的看到,如未能及早地确定最终解所在区域,将会出现进化停滞. 这是由于经过多次压缩空间使得搜索空间被控制在了一个错误的范围内,所探索到的只是在这个错误范围内的最优解. 虽然策略采用了多个群组的并行筛选策略,但由于初始粒子设置较少,所以无法很好地完成搜索工作. 但相较于原算法还是具有提升的.

此外,还采用相同的办法分别对采用的其他 3 种权重策略即随机权值、线性权重和非线性权重也进行了仿真实验. 通过图 8 可以进一步证明: 当算法初期寻优效果较好时,改进后的算法具有较强的进化能力. 且不易出现早熟现象. 表 3 中记录的是在问题维度等于 10 时,100 次实验解的相关数值特征. 从表 3 的方差项可以看出,改进效果对最终结果的稳定性(方差) 提升有较大帮助,且策略 2 的效果在大多数情况下要优于策略 1. 同样,在均值方面也处于领先地位. 进而在表 3 中的最大和最小值也可以看出逐层演化策略对算法的具体改进. 尤其是在 f_6 和 f_{11} 中的提升效果是十分显著的. 逐层演化策略的两种策略都能处于较好的保证对最终解的寻找,但结合图 7(f) 和图 7(k) 可以发现,最终收敛曲线的效果不足是由于算法不能稳定对该函数进行求解造成的.

表 4 记录了维度为 30 的时候,固定权重进行 100 次独立重复实验的数值特征. 相对于 10 维问题的求解来说,可以在表 4 中的方差一项清楚看出,求解效果的稳定性出现了较大的下滑. 但算法的改进效果依然十分明显. 同样可以发现在对函数 f_6 和 f_{11} 进行求解时

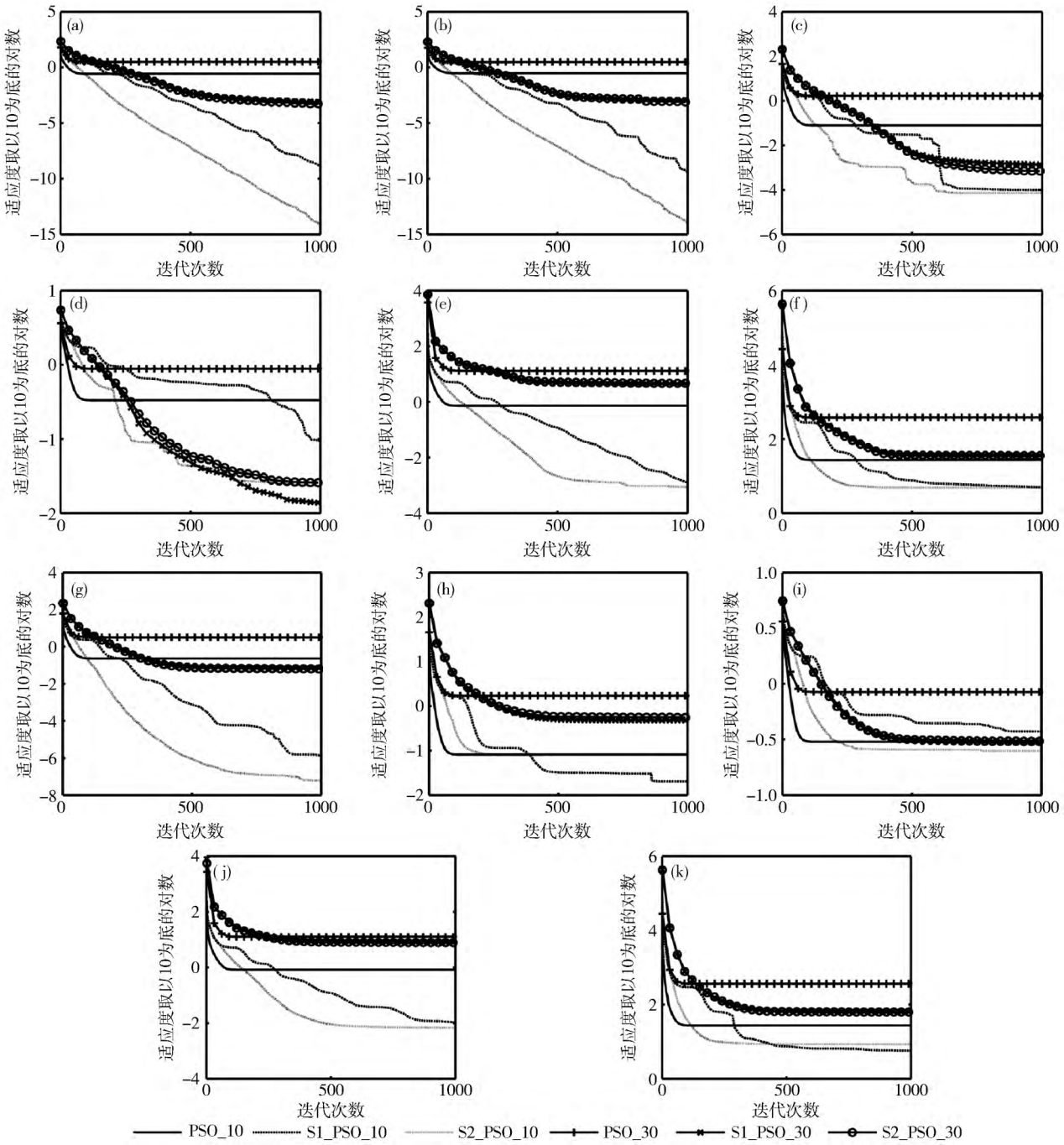

图 7　固定权重收敛曲线. (a) f_1; (b) f_2; (c) f_3; (d) f_4; (e) f_5; (f) f_6; (g) f_7; (h) f_8; (i) f_9; (j) f_{10}; (k) f_{11}

Fig. 7　Convergence performance of the constant weight: (a) f_1; (b) f_2; (c) f_3; (d) f_4; (e) f_5; (f) f_6; (g) f_7; (h) f_8; (i) f_9; (j) f_{10}; (k) f_{11}

存在着的问题. 进一步，表 4 中的最值则可以更直观地体现出算法的该井效果. 与之前表 3 中的不同，在表 4 中并未能完成对函数 f_6 和 f_{11} 的求解工作.

4　结论

(1) 压缩搜索空间切实可行，逐层演化的策略为相同或者不同类型算法之间的协作提供了良好的接口，是一种潜在的杂交算法模型框架.

(2) 在原算法可以完成基本寻优的条件下，逐层演化策略可以极大地提升算法效率. 由于整个改进措施并未对算法主体进行调整，不会造成原点或某个点的坍缩问题出现. 本文策略通过不断对邻域地压缩，逐渐搜寻到最优解.

(3) 该策略所带来的提升效果是具有广泛适用性的. 由于该策略在具体设计时未涉及任何粒子群算法所特有的演化策略，而是尽可能地考虑整个群体智能

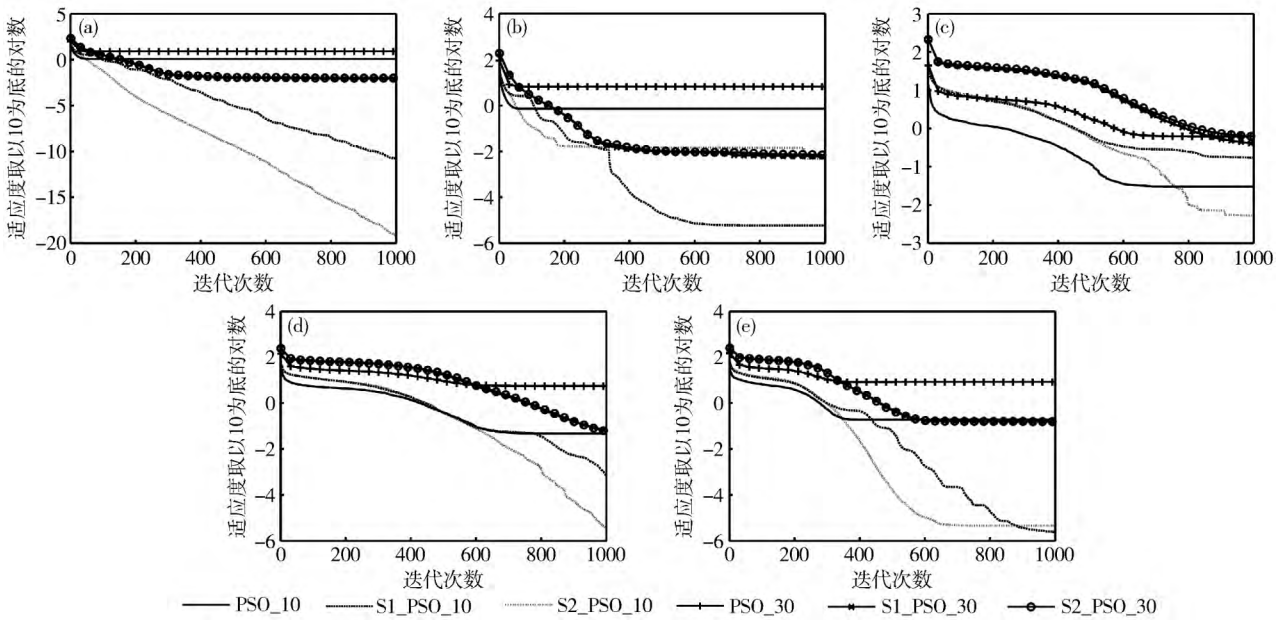

图 8　不同权重策略下的收敛曲线. (a) 随机权重下的f_1; (b) 随机权重下的f_3; (c) 线性权重下的f_3; (d) 线性权重下的f_7; (e) 非线性权重下的 f_7

Fig. 8　Convergence performance of different weights: (a) f_1 with random-weight; (b) f_3 with random-weight; (c) f_3 with linear-weight; (d) f_7 with linear-weight; (e) f_7 with nonlinear-weight

表 3　固定权重实验数据分析(10 维)

Table 3　Analysis of obtaining solutions under constant weight (D=10)

函数名	方差			均值		
	标准粒子群	策略 1	策略 2	标准粒子群	策略 1	策略 2
f_1	1.1282	6.0308×10^{-9}	2.7901×10^{-14}	1.0086	1.3991×10^{-9}	8.3702×10^{-15}
f_2	0.78058	1.4674×10^{-9}	6.048×10^{-14}	0.76403	4.0682×10^{-10}	1.1099×10^{-14}
f_3	1.3381	7.8044×10^{-4}	7.1379×10^{-4}	0.42132	$9.999e-05$	7.138×10^{-5}
f_4	0.57477	0.31842	0.12189	0.78178	0.095221	0.023389
f_5	2.0476	2.6102×10^{-3}	4.5881×10^{-3}	1.9656	0.0012553	0.00085644
f_6	161.0897	8.0314	3.5044	100.6025	5.0168	4.8764
f_7	0.85682	7.3719×10^{-6}	2.4297×10^{-7}	0.81468	1.3112×10^{-6}	6.0403×10^{-8}
f_8	1.3647	0.13995	0.4639	0.47642	0.020272	0.079874
f_9	0.60317	0.62451	0.52572	0.90764	0.37308	0.25052
f_{10}	2.2414	0.021445	0.010979	2.2676	0.010439	0.0067982
f_{11}	110.1334	4.9742	14.8816	101.1201	5.6683	8.4738

函数名	最小值			最大值		
	标准粒子群	策略 1	策略 2	标准粒子群	策略 1	策略 2
f_1	0.02147	8.8226×10^{-16}	1.2554×10^{-17}	7.2891	5.0411×10^{-8}	2.0414×10^{-13}
f_2	0.037832	1.1148×10^{-15}	1.3055×10^{-17}	3.8588	7.978×10^{-9}	5.9928×10^{-13}
f_3	0.00012295	5.6843×10^{-14}	0	10.0061	0.0076218	0.0071379
f_4	0.072415	5.0097×10^{-9}	1.4619×10^{-9}	2.6143	2.0137	1.1779
f_5	0.011101	2.5613×10^{-8}	2.5145×10^{-12}	11.255	0.018989	0.003329
f_6	8.4583	1.5749×10^{-6}	4.8059×10^{-6}	1437.9891	75.8412	9.0918
f_7	0.027766	2.0124×10^{-14}	3.9477×10^{-15}	5.0973	6.5277×10^{-5}	2.3163×10^{-6}
f_8	0.00098887	9.0381×10^{-10}	2.9715×10^{-11}	10.0468	0.995	0.9995
f_9	0.018175	3.9303×10^{-6}	2.2614×10^{-6}	2.417	2.0133	1.6464
f_{10}	0.052696	1.9219×10^{-5}	1.1726×10^{-5}	11.8583	0.12581	0.053114
f_{11}	3.7844	1.05×10^{-6}	9.2184×10^{-6}	575.4213	39.1231	81.8209

表 4　固定权重实验数据分析(30 维)

Table 4　Analysis of obtaining solutions under constant weight (D=30)

函数名	方差			均值		
	标准粒子群	策略 1	策略 2	标准粒子群	策略 1	策略 2
f_1	3.205	0.0019716	0.0013426	6.9363	0.00076592	0.00054158
f_2	3.8133	0.0042851	0.0027921	7.6343	0.0011544	0.0008001
f_3	5.7623	0.0069749	0.0015407	5.5119	0.0013778	0.00069827
f_4	0.47945	0.03693	0.091813	1.5833	0.013786	0.025675
f_5	22.9864	2.9231	2.9959	27.2878	4.482	4.6818
f_6	858.1021	14.6295	14.469	1048.2949	35.9452	35.6523
f_7	5.1435	0.057156	0.047947	7.6947	0.07127	0.064933
f_8	4.968	0.46587	0.80151	5.4284	0.47416	0.55524
f_9	0.53564	0.37398	0.36118	1.6354	0.30023	0.24424
f_{10}	28.4189	4.5261	3.5199	30.1211	8.1018	7.8266
f_{11}	930.9887	32.544	30.1093	1105.7393	64.056	62.5251

函数名	最小值			最大值		
	标准粒子群	策略 1	策略 2	标准粒子群	策略 1	策略 2
f_1	1.6355	1.2206×10^{-7}	6.6598×10^{-9}	21.8319	0.015959	0.0090582
f_2	2.8485	1.9937×10^{-8}	2.179×10^{-8}	23.295	0.035233	0.025575
f_3	0.99559	4.7299×10^{-7}	4.1461×10^{-8}	40.9301	0.064105	0.0081728
f_4	0.65022	2.4271×10^{-5}	0.00018403	2.8494	0.2004	0.80453
f_5	4.3958	0.37762	0.059936	131.4779	16.5186	14.507
f_6	124.9587	22.1206	25.8733	5366.0678	109.198	125.2924
f_7	2.1587	0.0017265	0.0016865	27.7803	0.32385	0.20228
f_8	1.0645	0.01745	0.03576	36.2923	2.5627	6.6635
f_9	0.68259	0.033952	0.01943	2.785	2.1191	1.7919
f_{10}	3.2344	0.98727	0.04448	141.5428	25.2545	16.3272
f_{11}	178.299	28.5085	18.5904	5020.2071	173.2692	167.2634

算法所共有的特性，因此，逐层演化策略可以适用于其他群体智能算法.

(4) 算法存在的不足: 逐层演化算法依赖于原算法的收敛速度与初期探索效果，如果原算法存在问题，会造成搜索空间被压缩于一个错误的区域内，或者无法进行任何压缩操作，造成改进策略的无效化; 算法可能会需要更多一些的初始成员来提升最终效果，当然并不是必须的.

参　考　文　献

[1] Richardson J J, Björnmalm M, Caruso F. Technology-driven layer-by-layer assembly of nanofilms. *Science*, 2015, 348(6233): aaa2491-1

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

[3] Zhang S P, Wang B. Dynamic search space for particle swarm optimization algorithm. *Appl Res Comput*, 2016, 33(7): 2047
(张水平，王碧. 动态搜索空间的粒子群算法. 计算机应用研究, 2016, 33(7): 2047)

[4] Eberhart R C, Shi Y. Comparing inertia weights and constriction factors in particle swarm optimization // *Proceedings of the* 2000 *Congress on Evolutionary Computation*. IEEE. La Jolla, 2000: 84

[5] Clerc M, Kennedy J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans Evol Comput*, 2002, 6(1): 58

[6] Shi Y, Eberhart R. A modified particle swarm optimizer // *The* 1998 *IEEE International Conference on Evolutionary Computation Proceedings*. IEEE. Anchorage, 1998: 69

[7] Nickabadi A, Ebadzadeh M M, Safabakhsh R. A novel particle swarm optimization algorithm with adaptive inertia weight. *Appl Soft Comput*, 2011, 11(4): 3658

[8] Zhan Z H, Zhang J, Li Y, et al. Orthogonal learning particle swarm optimization. *IEEE Trans Evol Comput*, 2011, 15(6): 832

[9] Xu G. An adaptive parameter tuning of particle swarm optimization algorithm. *Appl Math Comput*, 2013, 219(9): 4560

[10] De Oca M A M, Stützle T, Van Den Enden K, et al. Incremen-

tal social learning in particle swarms. *IEEE Trans Syst Man Cybern B*, 2011, 41(2): 368

[11] Xin B, Chen J, Zhang J, et al. Hybridizing differential evolution and particle swarm optimization to design powerful optimizers: a review and taxonomy. *IEEE Trans Syst Man Cybern C*, 2012, 42(5): 744

[12] Ni Q J, Zhang Z Z, Wang Z Z, et al. Dynamic probabilistic particle swarm optimization based on varying multi-cluster structure. *J Software*, 2009, 20(2): 339
(倪庆剑, 张志政, 王蓁蓁, 等. 一种基于可变多簇结构的动态概率粒子群优化算法. 软件学报, 2009, 20(2): 339)

[13] Mendes R, Kennedy J, Neves J. The fully informed particle swarm: simpler, maybe better. *IEEE Trans Evol Comput*, 2004, 8(3): 204

[14] Zhang W J, Xie X F, Bi D C. Handling boundary constraints for numerical optimization by particle swarm flying in periodic search space // *CEC*2004. *Congress on Evolutionary Computation*. IEEE. Portland, 2004: 2307

[15] Helwig S, Branke J, Mostaghim S. Experimental analysis of bound handling techniques in particle swarm optimization. *IEEE Trans Evol Comput*, 2013, 17(2): 259

[16] Zhan Z H, Zhang J, Li Y, et al. Adaptive particle swarm optimization. *IEEE Trans Syst Man Cybern B*, 2009, 39(6): 1362

[17] Riget J, Vesterstrøm J S. *A Diversity-guided Particle Swarm Optimizer: the ARPSO*. Aarhus: Aarhus University, 2002

[18] Wang B, Luo X, Zhang S P. Particle swarm optimization algorithm based on homing. *J Jiangxi Univ Sci Technol*, 2015, 36(3): 95
(王碧, 罗潇, 张水平. 一种返巢模式下的粒子群优化策略. 江西理工大学学报, 2015, 36(3): 95)

[19] Eberhart R C, Shi Y H. Tracking and optimizing dynamic systems with particle swarms // *Proceedings of the* 2001 *Congress on Evolutionary Computation*. IEEE. Seoul, 2001: 94

[20] Shi Y, Eberhart R C. Empirical study of particle swarm optimization // *CEC* 99. *Proceedings of the* 1999 *Congress on Evolutionary Computation*. IEEE. Washington D. C., 1999

[21] Chatterjee A, Siarry P. Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization. *Comput Oper Res*, 2006, 33(3): 859

[22] Li X D, Tang K, Omidvar M N, et al. Benchmark functions for the CEC 2013 special session and competition on large-scale global optimization. *Gene*, 2013, 7(33): 8