控制与决策
Control and Decision

一种自适应多策略行为粒子群优化算法

张强， 李盼池

引用本文:
张强, 李盼池. 一种自适应多策略行为粒子群优化算法[J]. 控制与决策, 2020, 35(1): 115−122.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2018.0240

文章编号: 1001-0920(2020)01-0115-08 DOI: 10.13195/j.kzyjc.2018.0240

一种自适应多策略行为粒子群优化算法

张 强†, 李盼池

(东北石油大学 计算机与信息技术学院，黑龙江 大庆 163318)

摘 要: 针对粒子群优化算法收敛速度慢、局部搜索能力差等缺点,提出一种自适应多策略行为粒子群优化算法. 算法中每个粒子拥有4种行为进化策略,在迭代过程中通过计算每种进化策略的立即价值、未来价值和综合奖励来决定粒子的进化行为,并通过策略行为概率变异算法提升个体寻优速度或避免陷入局部最优解. 在经典的基准测试函数上,对新算法与其他7个群智能进化算法的测试结果进行比较分析,结果表明所提出算法具有很好的求解精度和收敛速度,尤其适合应用于一些高维优化问题.

关键词: 粒子群算法；多策略；差分变异；上限置信区间；优化；极限学习机

中图分类号: TP273 **文献标志码:** A

An adaptive multi-strategy behavior particle swarm optimization algorithm

ZHANG Qiang†, LI Pan-chi

(School of Computer and Information Technology，Northeast Petroleum University，Daqing 163318，China)

Abstract: Aiming at the shortcomings of slow convergence rate and poor local search ability of particle swarm optimization algorithms, an adaptive multi-strategy particle swarm optimization algorithm is proposed. Each particle has four behavioral evolution strategies in the algorithm. In the iteration process, the evolutionary behavior of the particles is determined by calculating the immediate value, the future value and the comprehensive reward of each evolutionary strategy, and the strategy behavioral mutation algorithm is proposed to improve the individual search speed or to avoid falling into the local optimal solution. Comparison of the results of the proposed algorithm with the other 7 swarm intelligence evolutionary algorithms for the classical benchmark function show that the algorithm has better accuracy and convergence speed, especially suitable for some high-dimensional optimization problems.

Keywords: particle swarm optimization；multi-strategy；differential variation；upper confidence bound；optimization；extreme learning machine

0 引 言

自1995年Kennedy等[1]提出粒子群优化算法(Particle swarm optimization, PSO)以来,引起了学者们的极大关注. PSO通过粒子间的竞争和协作实现在复杂搜索空间中寻找全局最优点,具有易理解、易实现、全局搜索能力强等特点,已经成为发展最快的智能优化算法之一. 目前,粒子群算法在参数优化[2]、神经网络训练[3]、聚类分析[4]、组合优化[5]、模式识别与图像处理[6-7]等多个领域得到了广泛应用. 为提高经典粒子群算法的寻优性能,许多学者提出了PSO的改进算法,这些改进算法大体上可以分为4类: 群体拓扑结构和保持种群多样性[8-9]、参数控制[10-12]、新的进化策略方面[13-15]、与其他智能算法融合方面[16-18]. 研究表明,惯性权重在PSO中是非常重要的参数. 文献[19]提出一种自适应惯性权重调整策略,将权重的变化与粒子适应值的变化联系起来,但这也仅是降低算法发生早熟收敛的可能性;文献[20]在速度更新方式中引入局部最优位置,通过动态调整个体最优、局部最优和全局最优在速度更新公式中的权重来改进PSO的性能,但在高维函数优化中效果并不理想且参数设置较多. 对影响算法寻优性能进行分析,除了参数设置对算法的影响外,粒子进化策略的不变性也是造成算法寻优效果不高的重要原因,比如在整个进化过程中粒子均要受到全局最优粒子的影响,对于多峰函数优化问题,如果全局最优粒子陷入局部最优解,则整个种群陷入局部最优解的概率就

收稿日期: 2018-03-02；**修回日期:** 2018-07-10.

基金项目: 国家自然科学基金项目(61702093)；黑龙江省自然科学基金项目(F2018003).

责任编委: 孙秋野.

†通讯作者. E-mail: dqpi_zq@163.com.

会增大,虽然现在一些改进算法通过混沌变异的方式来增加跳出局部最优解的几率,但从本质上而言也是个体进化行为策略的改变.

本文提出一种自适应多策略行为粒子群优化算法,通过多策略行为选择概率算法使粒子进化具有一定的自主性,使个体可以根据历史经验和环境感知来选择进化策略,力求个体在每次迭代过程中能获得较大价值.

1 粒子群优化算法原理

设粒子群种群规模为Num,其中每个粒子在D维空间中的坐标位置向量表示为$\boldsymbol{X}_i=(x_{i1},x_{i2},\cdots,x_{id},\cdots,x_{iD})$,速度向量表示为$\boldsymbol{V}_i=(v_{i1},v_{i2},\cdots,v_{id},\cdots,v_{iD})$,粒子个体最优位置表示为$\boldsymbol{P}_i=(p_{i1},p_{i2},\cdots,p_{id},\cdots,p_{iD})$,群体最优位置表示为$\boldsymbol{P}_g=(p_{g1},p_{g2},\cdots,p_{gd},\cdots,p_{gD})$.不失一般性,以最小化问题为例,粒子群优化算法的速度和位置按照下式进行更新:

$$v_{i,t+1}^d=\omega v_{i,t}^d+c_1\mathrm{rand}(p_{i,t}^d-x_{i,t}^d)+c_2\mathrm{rand}(p_{g,t}^d-x_{i,t}^d),\tag{1}$$

$$x_{i,t+1}^d=x_{i,t}^d+v_{i,t+1}^d.\tag{2}$$

其中:ω为惯性权重;c_1,c_2为学习因子;rand( )为(0,1)之间的随机数;$p_{g,t}^d$为群体最优位置;$p_{i,t}^d$为个体最优位置.

个体最优位置通过下式进行迭代计算:

$$p_{i,t+1}^d=\begin{cases}x_{i,t+1}^d f(X_{i,t+1})<f(P_{i,t}),\\ p_{i,t}^d.\end{cases}\tag{3}$$

2 自适应多策略行为粒子群优化算法原理

整个群体的全局最优解(Gbest)仅是算法迭代过程中某个粒子自身所能找到的最优位置,只能代表所有粒子的目前最好水平,而不能代表目前的整体进化水平. 特别是在求解高维复杂的优化问题时,Gbest很可能是一个局部极值,仅利用这个局部极值来指导整个群体的学习,容易陷入局部最优,较难达到需要的求解效果. 进化论研究表明:生物对环境有巨大的适应能力,环境的变化会引起生物的变化,生物会由此改进其行为,环境的多样化是生物多化的根本原因,而粒子群算法个体所要面临的环境就是其他个体传递给它的知识,粒子个体参照的知识不同则进化行为也应有所不同,即个体的行为在整个进化过程中不应该是不变的. 粒子群体中的个体在整个进化过程中,能够获得知识的来源主要分为4部分:1)个体自身进化过程中的知识(个体最优解);2)全局最优个体的知识(全局最优解);3)其他个体进化过程中的知识;4)进化过程中的群体平均知识.本文针对这4类知识提出4种策略行为以完成个体进化.

2.1 多策略行为进化方式

1) 基于个体最优和全局最优个体学习的进化方式.

这种方式保留了粒子群原有的进化方式,但是式(2)具有3个可变参数,对于c_1和c_2的选择具有一些固定的优选值,针对不同的优化问题,这两个参数的设置对求解问题的结果具有一定的影响[21]. 本文不考虑c_1和c_2的影响,赋予粒子较大的自主性,提出改进的自适应权重进化方式

$$\begin{cases}v_{i,t+1}^d=\omega v_{i,t}^d+\mathrm{rand}(p_{i,t}^d-x_{i,t}^d)+\\ \qquad\mathrm{rand}(p_{g,t}^d-x_{i,t}^d),\\ \omega=\omega_{\max}-\dfrac{t}{T}(\omega_{\max}-\omega_{\min}).\end{cases}$$

其中:t为当前迭代次数,T为总的迭代次数,$\omega_{\max}$、$\omega_{\min}$为惯性权重的最大值和最小值.

2) 基于群体平均知识学习的进化方式.

群体平均知识可以用“群体位置质心”来代替,为方便计算采用各粒子位置的均值作为质心,即

$$v_{i,t+1}^d=\omega v_{i,t}^d+\mathrm{rand}\Big(\frac{\sum_{i=1}^{\mathrm{Num}}x_{i,t}^d}{\mathrm{Num}}-x_{i,t}^d\Big),\tag{4}$$

其中Num为粒子群种群规模.

3) 基于全局最优个体和其他个体知识学习的进化方式.

采用差分算法[22]的DE/best/1进化方式进行表示,有

$$v_{i,t+1}^d=p_{g,t}^d+\mathrm{rand}(x_{r1,t}^d-x_{r2,t}^d),\tag{5}$$

其中r_1,r_2为[1,Num]之间的随机整数,且$r_1\neq r_2$.

4) 基于其他个体知识学习的进化方式.

采用差分算法的DE/rand/1进化方式来表示,这种随机选取个体进行进化的方式等价于个体变异,并且这种变异方式与一些基于混沌理论[23]进行变异方式相比较,更具有目的性和方向性,有

$$v_{i,t+1}^d=x_{r1,t}^d+\mathrm{rand}(x_{r2,t}^d-x_{r3,t}^d),\tag{6}$$

其中r_1,r_2,r_3为[1,Num]之间的随机整数,且$r_1\neq r_2\neq r_3$.

2.2 计算多策略行为立即价值

所谓立即价值是指采用某种策略行为后,所产生的适应度$f(x_i,t)$较上一次迭代计算适应度的提高程度,有

$$\mathrm{value}_i(t)=\frac{f(x_i,t-1)-\min(f(x_i,t-1),f(x_i,t))}{f(x_i,t-1)}. \quad (7)$$

若计算的立即价值为零,则只能说明该策略目前没有取得很好的寻优效果,不能代表此种策略不好,还需计算其未来价值.

2.3 计算多策略行为未来价值

粒子个体在没有任何先验知识的前提下,在每次确定所要采取的策略行为时,可通过对已获得知识的利用和对新知识的探索进行协调来确定,以便更快捷地做出最优决策. 信心上界(Upper confidence bound, UCB)算法[24-25]的上限置信区间由两部分之和构成,其中一部分是当前已有回报值,另一部分与回报值在单侧置信区间的大小有关,以保证所期望的回报能以极大的可能性落在回报范围之内.该算法已被应用到计算机围棋程序中并取得了很好的效果,利用UCB算法根据棋盘当前局面的立即价值和可选落子点的未来价值计算上限置信区间最大的落子点作为当前局面的落子点. 粒子个体的策略行为选择可以类比于围棋的落子点决策.

令$N_i{}^p(t)$为粒子采用第i种策略在第t次迭代前的成功次数,$M_i{}^p(t)$为粒子采用第i种策略在第t次迭代前总的执行次数,$N_i{}^g(t)$为所有粒子采用第i种策略在第t次迭代前的成功个数,$M_i{}^g(t)$为所有粒子采用第i种策略在第t次迭代前总的执行次数. 分两种情况按照UCB公式计算未来价值,一种是单个粒子执行某一策略的成功率,另一种是该策略在整个种群的成功率. Num为种群规模,有

$$\mathrm{success}_i(t)=\alpha\times\Big(\frac{N_i{}^p(t)}{M_i{}^p(t)}+\sqrt{\frac{C_0\times\log(t)}{N_i{}^p(t)}}\Big)+(1-\alpha)\Big(\frac{N_i{}^g(t)}{M_i{}^g(t)}+\sqrt{\frac{C_0\times\log(\mathrm{Num}\times t)}{\sum_{i=1}^{\mathrm{Num}}N_i{}^p}}\Big). \quad (8)$$

其中:t为当前迭代的次数;α和C_0为一个常数,α的作用是平衡个体未来价值与全局未来价值在策略行为预测的重要性,C_0的作用是平衡个体所获得知识的利用和探索需求,本文按照UCB算法的取值令$C_0=2$.

2.4 计算多策略行为综合奖励

在考察一种策略是否有效时,需考虑实施该策略的立即价值、估算的未来价值,同时还要考虑之前的历史经验$P_i(t)$,即上一次迭代采用该策略的概率,故采用下式计算一个粒子采用某种策略所能获得的综合奖励:

$$\mathrm{score}_i(t)=\mathrm{value}_i(t)+\mathrm{success}_i(t)+P_i(t),\quad 1\leqslant i\leqslant M. \quad (9)$$

2.5 更新多策略行为选择概率

每个粒子在分别计算完采用M种策略行为的综合奖励后,更新在第$t+1$次进化过程中采用某种策略行为的概率,有

$$P_i(t+1)=P_{\min}+\frac{\mathrm{score}_i(t)}{\sum_{i=1}^{N}\mathrm{score}_i(t)}(1-M\times P_{\min}). \quad (10)$$

为保证每种策略的选择概率不为0,需设置一个最小概率$P_{\min}<1/M$.

2.6 策略行为概率变异算法

在算法的进化过程中会存在个体一直采取某种策略的行为,虽然每次进化都会对个体适应值有所提升,但提升幅度不大,这会造成个体寻优速度缓慢或陷入局部最优的情况,需要对这种情况进行判断以降低当前所采取策略行为的选择概率.适应度方差可以反映粒子的收敛程度,方差越小表明或已陷入局部最优,或已发现全局最优解. 算法利用个体适应值的方差进行判断,有

$$\sigma^2=\frac{\sum_{i=t-\mathrm{count}}^{t}(f_i-\bar{f})^2}{\mathrm{count}}. \quad (11)$$

其中:t为当前的迭代次数,count为第t次迭代前的适应值个数(本文取10),f_i为每次迭代的适应值,$\bar{f}$为count个适应值的均值. 如果σ^2小于某一阈值则将该策略行为的概率置为$P_{\min}$.

3 自适应多策略行为粒子群优化算法流程

Step 1: 设定种群规模Num,总迭代次数T,优化维数,结束条件,并对各个粒子的位置和速度进行初始化.

Step 2: 按照第2.2节～第2.5节计算每个粒子使用该策略行为的立即价值、未来价值、综合奖励,并更新每种行为策略的选择概率.

Step 3: 在第2.1节4种策略行为中,每个粒子选择概率最大的策略行为更新个体速度和位置.

Step 4: 计算每个粒子的适应值并更新个体最优位置和全局最优位置.

Step 5: 计算最近count迭代次数适应值的方差,如果小于指定阈值,则将该个体最后采用的策略行为

概率重新初始化.

Step 6: 如果满足指定迭代次数或结束条件则算法结束,否则转至Step 2.

4 数值实验及分析

4.1 函数极值优化

为验证本文所提出算法AMBPSO的特点,将其与PSO[1]、PSOGSA[16]、DGLCPSO[20]、两种变异方式(DE/best/1、DE/rand/1)的差分进化算法[26]、线性递减权重PSO[27]和自适应权重PSO[28]进行寻优性能对比. 实验环境为: Windows7操作系统, Intel酷睿i5处理器,主频2.5 Hz, 4 G内存,开发工具为Matlab R2014a.

针对9个函数极值求解问题,各算法的参数设置如下:种群个数均设置为各自迭代1 000次,每个算法独立运行10次; AMBPSO的$\alpha = 0.5, \omega \in [0.2, 0.9]$; PSO的$\omega = 0.729\,8, c_1 = 1.496\,2, c_2 = 1.496\,2$; LinwPSO的$\omega \in [0.2, 0.9], c_1 = 1.496\,2, c_2 = 1.496\,2$; SAPSO的$\omega \in [0.2, 0.9], c_1 = 1.496\,2, c_2 = 1.496\,2$. 参照文献[20]设置DGLCPSO, $\omega = 0.3, a = 0.6, b = 1.005, c = 2$. 参照文献[16]设置PSOGSA, $G_0 = 1, c_1 = 0.5, c_2 = 1.5$. 测试函数如下:

$$f_1 = \sum_{i=1}^{n} x_i^2, \ -100 \leqslant x_i \leqslant 100. \tag{12}$$

$$f_2 = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i|, \ -10 \leqslant x_i \leqslant 10. \tag{13}$$

$$f_3 = \sum_{i=1}^{n} \Big(\sum_{j=1}^{i} x_j \Big), \ -10 \leqslant x_i \leqslant 10. \tag{14}$$

$$f_4 = \max(abs(x_i)), \ -10 \leqslant x_i \leqslant 10. \tag{15}$$

$$f_5 = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2], \quad -30 \leqslant x_i \leqslant 30. \tag{16}$$

$$f_6 = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)] + 10n, \quad -5.12 \leqslant x_i \leqslant 5.12. \tag{17}$$

$$f_7 = 20 + \exp(1) - 20\exp\Big[-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\Big] - \exp\Big[\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\Big], \ -32 \leqslant x_i \leqslant 32. \tag{18}$$

$$f_8 = \frac{1}{4\,000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\Big[\frac{x_i}{\sqrt{i}}\Big] + 1, \quad -600 \leqslant x_i \leqslant 600. \tag{19}$$

$$f_9 = \frac{\pi}{n}\Big\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\Big\} + \sum_{i=1}^{n} u(x_i, 10, 1\,000, 4),$$

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a; \\ 0, & -a \leqslant x_i \leqslant a; \\ k(-x_i - a)^m, & x_i < -a; \end{cases}$$

$$y_i = 1 + \frac{x_i + 1}{4}. \tag{20}$$

分别对每种算法的不同维数$(10, 30, 60, 100)$进行寻优,为对比本文算法在高维函数寻优的性能,选取100维函数测试结果数据如表1～表9所示,测试平均最优适应值迭代曲线如图1～图9所示.

表 1 函数f_1的运行仿真结果对比

算法	最优结果	最大值	平均值	方差
AMBPSO	7.96e-113	2.31e-99	4.63e-100	1.04e-99
PSO	3.58e+04	5.26e+04	4.26e+04	7.12e+03
DE/best/1	2.33e+04	4.79e+04	3.55e+04	9.62e+03
DE/rand/1	7.89e+02	3.22e+03	1.88e+03	9.76e+02
LinwPSO	2.71e+04	3.93e+04	3.28e+04	4.48e+03
SAPSO	2.97e+04	5.07e+04	3.66e+04	8.75e+03
DGLCPSO	6.20e+03	1.04e+04	8.26e+03	1.76e+03
PSOGSA	3.29e+04	7.67e+04	6.04e+04	1.92e+04

表 2 函数f_2的运行仿真结果对比

算法	最优结果	最大值	平均值	方差
AMBPSO	2.72e-66	1.18e-44	2.36e-45	5.29e-45
PSO	2.65e+02	3.49e+02	3.09e+02	3.21e+01
DE/best/1	8.00e+01	1.71e+02	1.33e+02	3.69e+01
DE/rand/1	7.18e-01	1.47e+01	4.27e+00	5.91e+00
LinwPSO	1.51e+02	2.18e+02	1.75e+02	2.56e+01
SAPSO	1.64e+02	2.08e+02	1.84e+02	1.90e+01
DGLCPSO	1.21e+01	2.09e+01	1.72e+01	3.64e+00
PSOGSA	3.53e+02	5.61e+10	1.12e+10	2.51e+10

表 3 函数f_3的运行仿真结果对比

算法	最优结果	最大值	平均值	方差
AMBPSO	1.44e-107	9.41e-86	1.88e-86	4.21e-86
PSO	1.22e+06	4.35e+06	2.11e+06	1.28e+06
DE/best/1	2.22e+05	3.16e+05	2.65e+05	4.30e+04
DE/rand/1	4.87e+05	7.20e+05	5.59e+05	9.75e+04
LinwPSO	9.31e+04	2.73e+05	1.70e+05	7.31e+04
SAPSO	2.15e+05	5.59e+05	3.21e+05	1.39e+05
DGLCPSO	2.52e+04	1.93e+05	8.79e+04	7.05e+04
PSOGSA	1.75e+05	2.54e+05	2.04e+05	3.57e+04

表 4　函数 f_4 的运行仿真结果对比

算法	最优结果	最大值	平均值	方差
AMBPSO	7.23e-64	9.46e-51	1.89e-51	4.23e-51
PSO	9.54e+01	1.00e+02	9.84e+01	1.77e+00
DE/best/1	6.17e+01	7.41e+01	6.86e+01	4.60e+00
DE/rand/1	5.21e+01	6.93e+01	6.08e+01	6.81e+00
LinwPSO	4.10e+01	5.30e+01	4.53e+01	4.97e+00
SAPSO	3.90e+01	5.00e+01	4.51e+01	4.45e+00
DGLCPSO	2.86e+01	3.39e+01	3.14e+01	2.28e+00
PSOGSA	7.44e+01	9.79e+01	9.22e+01	1.00e+01

表 5　函数 f_5 的运行仿真结果对比

算法	最优结果	最大值	平均值	方差
AMBPSO	9.90e+01	9.90e+01	9.90e+01	9.27e-03
PSO	1.20e+07	1.38e+08	1.28e+08	7.03e+07
DE/best/1	2.04e+07	4.91e+07	3.93e+07	1.18e+07
DE/rand/1	8.90e+05	4.49e+06	2.65e+06	1.63e+06
LinwPSO	2.41e+07	3.44e+07	2.86e+07	3.79e+06
SAPSO	2.44e+07	6.13e+07	4.02e+07	1.46e+07
DGLCPSO	4.82e+02	9.07e+02	6.42e+02	1.72e+02
PSOGSA	1.89e+06	1.62e+08	8.16e+07	7.98e+07

表 6　函数 f_6 的运行仿真结果对比

算法	最优结果	最大值	平均值	方差
AMBPSO	0.00e+00	0.00e+00	0.00e+00	0.00e+00
PSO	7.92e+02	9.91e+02	8.93e+02	7.90e+01
DE/best/1	3.45e+02	4.68e+02	4.13e+02	5.68e+01
DE/rand/1	6.49e+02	6.83e+02	6.68e+02	1.37e+01
LinwPSO	8.13e+02	9.65e+02	9.11e+02	6.30e+01
SAPSO	9.29e+02	1.08e+03	9.92e+02	6.89e+01
DGLCPSO	1.89e+02	2.50e+02	2.12e+02	2.32e+01
PSOGSA	4.49e+02	7.63e+02	6.34e+02	1.16e+02

表 7　函数 f_7 的运行仿真结果对比

算法	最优结果	最大值	平均值	方差
AMBPSO	8.88e-16	8.88e-16	8.88e-16	0.00e+00
PSO	2.11e+01	2.13e+01	2.12e+01	6.91e-02
DE/best/1	1.50e+01	1.60e+01	1.53e+01	4.10e-01
DE/rand/1	4.90e+00	7.96e+00	6.43e+00	1.12e+00
LinwPSO	1.45e+01	1.59e+01	1.55e+01	6.15e-01
SAPSO	1.49e+01	1.73e+01	1.60e+01	8.45e-01
DGLCPSO	1.29e+01	1.36e+01	1.32e+01	2.93e-01
PSOGSA	1.91e+01	2.00e+01	1.95e+01	4.14e-01

表 8　函数 f_8 的运行仿真结果对比

算法	最优结果	最大值	平均值	方差
AMBPSO	0.00e+00	3.77e-02	1.15e-02	1.70e-02
PSO	2.61e+03	3.22e+03	2.93e+03	2.70e+02
DE/best/1	1.72e+02	2.87e+02	2.24e+02	4.66e+01
DE/rand/1	8.73e+00	3.73e+01	2.12e+01	1.24e+01
LinwPSO	2.19e+02	3.81e+02	2.95e+02	6.13e+01
SAPSO	2.92e+02	3.63e+02	3.14e+02	2.89e+01
DGLCPSO	6.24e+02	7.35e+02	6.97e+02	5.00e+01
PSOGSA	4.83e+02	7.54e+02	6.64e+02	1.12e+02

表 9　函数 f_9 的运行仿真结果对比

算法	最优结果	最大值	平均值	方差
AMBPSO	9.39e-01	1.24e+00	1.11e+00	1.31e-01
PSO	2.59e+07	3.24e+08	2.89e+08	2.44e+08
DE/best/1	1.56e+07	1.15e+08	4.43e+07	4.04e+07
DE/rand/1	3.05e+06	1.55e+07	7.57e+06	5.77e+06
LinwPSO	4.30e+06	2.76e+07	1.23e+07	9.14e+06
SAPSO	2.23e+06	5.18e+07	1.65e+07	2.04e+07
DGLCPSO	1.32e+01	1.54e+01	1.41e+01	8.57e-01
PSOGSA	3.95e+06	1.02e+09	4.12e+08	4.27e+08

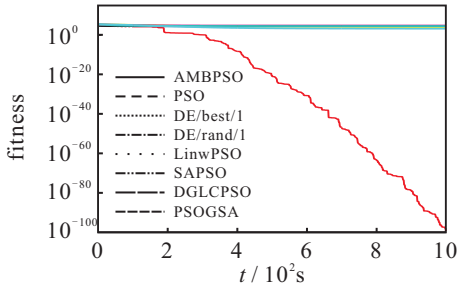

图 1　f_1 最优适应值迭代曲线

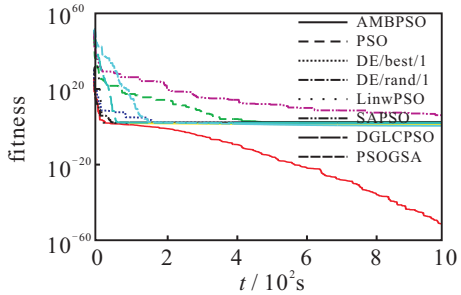

图 2　f_2 最优适应值迭代曲线

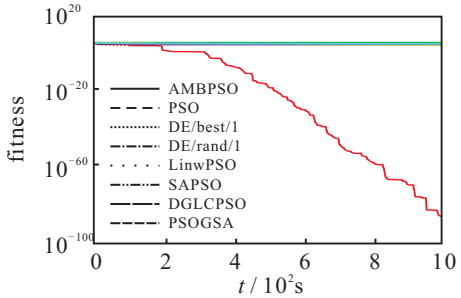

图 3　f_3 最优适应值迭代曲线

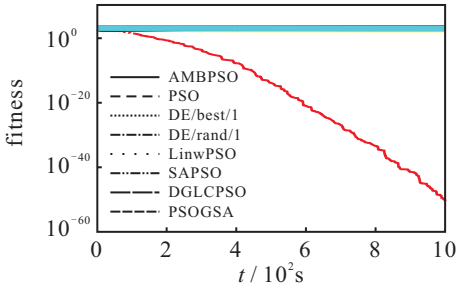

图 4　f_4 最优适应值迭代曲线

由仿真结果可以看出,本文所提出的自适应多策略行为粒子群优化算法具有很好的求解精度和求解速度. 从以下3个方面进行分析:

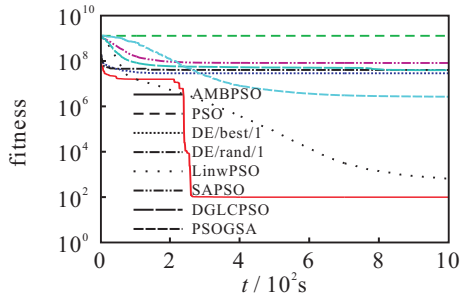

图 5 f_5 最优适应值迭代曲线

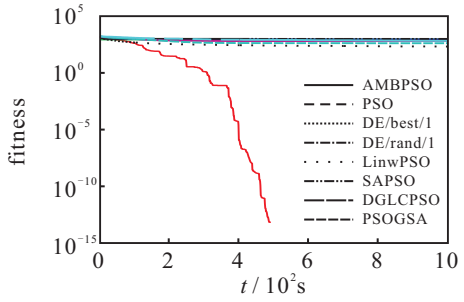

图 6 f_6 最优适应值迭代曲线

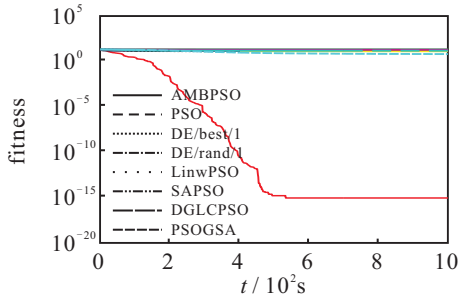

图 7 f_7 最优适应值迭代曲线

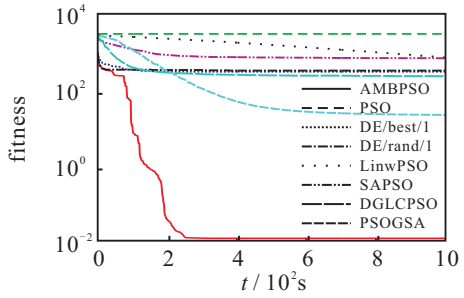

图 8 f_8 最优适应值迭代曲线

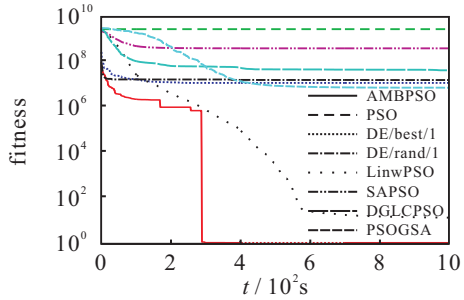

图 9 f_9 最优适应值迭代曲线

1) 从最优结果、最差结果、平均结果和方差可以看出, AMBPSO 寻优结果最好. 观察表 1 ~ 表 9, 由不同维数的实验结果可知本文算法具有很好的稳定性, 即在增加维数的情况下寻优效果变化不大, 都可以获得较为理想的解. 虽然从函数 f_9 的 10 维测试结果中可知 DE/best/1、DE/rand/1 和 PSOGSA 的最优结果优于 AMBPSO, 但从平均结果的比较可见其性能相差不多. 但随着维数的增加, 这些算法的性能下降很快, 表明这 3 种算法求解高维问题具有不稳定性. 本文算法的性能随维数增加没有明显改变. 主要是因为 AMBPSO 每个寻优个体都会在迭代过程不断吸取知识, 并将其转换为指导自己采用不同行为的进化策略, 算法可以充分利用每步迭代获得的立即价值、估算的未来价值和历史经验来判断如何从其他个体汲取有用的知识, 可以让算法在自身感知的情况下选用较优的进化行为, 进化策略 3 和策略 4 引入的差分变异算法可以避免陷入局部最优, 有利于获得全局最优解, 进而解决算法在求解某些复杂函数时收敛速度慢、容易陷入局部最优的缺点.

2) 由迭代曲线可以看出, 虽然文中实例的迭代次数都是 1 000 次, 但是 AMBPSO 的迭代次数和方差均优于对比算法, 其原因是: AMBPSO 策略行为的综合奖励可以较好地指引粒子选择较优的进化行为更新个体位置. 由 $f_1 \sim f_4$ 的迭代曲线可以看出, 其他算法由于采用固定的进化策略和单一的知识来源, 使得寻优能力在达到一定精度后再难跳出局部最优, 而从 AMBPSO 的 $f_1 \sim f_4$ 迭代曲线可以看出, 即使维数达到 100, 算法仍能以较快的迭代方式不断更新最优解, 而 $f_5 \sim f_9$ 的迭代曲线显示, AMBPSO 以较小的迭代次数获取较优结果. 可知, 算法在迭代后期寻优性能的效果不是很明显, 这也从侧面说明了“没有免费午餐定理”. 从实验结果来看, 文中算法相对于 7 个比较算法而言还是具有很好的优化性能, 尤其在处理高维函数过程中具有较好的适应性.

3) 分析 AMBPSO 的算法原理可知, 虽然粒子有 4 种进化策略行为, 但对比式 (1) 与 (4) ~ (7) 可见, 实际上是减少了原有粒子速度更新的计算量. 由于每次迭代个体都是选取一种获得价值最高的策略行为来更新速度, AMBPSO 的速度计算量要小于 PSO, 但每次迭代都会计算策略行为的综合奖励, 如果 AMBPSO 加上综合奖励的计算量, 则相对于 PSO 计算量要多, 但根据综合奖励计算公式和依据算法复杂性渐近性理论可知, 改进后的 AMBPSO 没有增加经典 PSO 算法的时间复杂度. 从迭代曲线可知, 算法在迭代次数 500 次内就已获得较好的寻优结果, 这表明综合奖励的计算能够有效地增强粒子个体寻优性能, 避免了一些局部震荡计算, 这种寻优特性尤其适合适应度计算量复杂的优化问题, 即以更少的迭代次

数获取更优的结果,换言之就是当计算适应度函数值的时间复杂度越大于AMBPSO计算综合奖励的复杂度时,越能体现AMBPSO算法计算能力的优越性.

4.2 极限学习机网络权值优化

极限学习机[29](Extreme learning machine, ELM)具有学习速度快且泛化性能好的优点,但是连接权值、偏置阈值、隐含层节点个数,对极限学习机的性能存在很大影响,实际应用中需要大量的隐含层节点才能获得预期的效果. 传统ELM的权值及隐含层偏差都是随机的,无法保证这些参数都是最优的,并且广义逆矩阵的计算量会随着矩阵规模的变大而增加,其计算复杂度要多于本文的综合奖励计算量. 为此,采用所提出算法对权值及偏差进行优化建立分类模型,并与4种粒子群改进算法进行对比. 算法参数设置参照第4.1节,连接权值$[-1, 1]$,EML的隐层节点数设置为20. 从UCI标准分类数据集中选择5种数据集进行实验,每次取数据集的70%作为训练数据集,剩下的30%作为测试数据集,分别运行10次,取10次训练和预测的平均值,训练数据集识别率大于80%即为收敛.每个数据集分类性能对比如表10所示.

表 10　各类方法分类性能对比

UCI数据	算法	训练识别率/%	测试识别率/%	收敛次数
Heart	AMBPSO	88.65	85.42	10
	LinwPSO	78.62	70.62	0
	SAPSO	79.43	77.98	1
	DGLCPSO	81.53	78.01	4
	PSOGSA	84.61	80.92	8
Kr_V_kp	AMBPSO	93.43	92.71	10
	LinwPSO	77.1	69.19	0
	SAPSO	79.82	73.17	2
	DGLCPSO	86.02	78.43	4
	PSOGSA	89.45	81.58	6
Pendigits	AMBPSO	88.01	82.16	10
	LinwPSO	76.36	65.83	0
	SAPSO	78.71	73.35	0
	DGLCPSO	79.85	74.62	0
	PSOGSA	82.09	78.47	3
Tic_tac_toc	AMBPSO	87.85	84.95	9
	LinwPSO	78.31	70.31	0
	SAPSO	78.13	68.57	0
	DGLCPSO	80.71	73.19	2
	PSOGSA	82.42	78.25	5
Balance	AMBPSO	95.66	93.48	10
	LinwPSO	91.02	86.36	10
	SAPSO	91.51	87.37	10
	DGLCPSO	91. 81	89.86	10
	PSOGSA	92.43	90.61	10

由表10分类性能对比结果可知,AMBPSO可以使ELM获得很好的数据集识别率,尤其对数据样本数和特征维数较多的数据集Kr_V_kp、Pendigits、Tic_tac_toc. 根据ELM建模原理,对于这3类UCI数据集,需要优化的输入权值及隐含层偏差相对较多,这也表明本文算法对于高维数据寻优具有较好的效果.

5 结　论

本文针对粒子群优化算法收敛速度慢、局部搜索能力差等缺点,提出一种自适应多策略行为粒子群优化算法. 算法中每个粒子拥有4种行为进化策略,在迭代过程中通过计算每种进化策略的立即价值、未来价值和综合奖励来决定粒子的进化行为,并通过策略行为概率变异算法提升个体寻优速度或避免陷入局部最优解. 在经典的基准测试函数上,对新算法与其他7个群智能进化算法的测试结果进行比较分析,结果表明所提出算法具有很好的求解精度和收敛速度,尤其适合应用于一些高维优化问题.

参考文献(References)

[1] Kennedy J, Eberhart R C. Particle swarm optimization[C]. Proc of IEEE Int Conf on Neural Networks. New York: IEEE, 1995: 1942-1948.

[2] 游浩, 申永军, 杨绍普. 多目标粒子群优化PCNN参数的图像融合算法[J]. 振动与冲击, 2017, 36(16): 224-254.
(You H, Shen Y J, Yang S P. Parameters design for passive fractional-order vehicle suspension based on particle swarm optimization[J]. J of Vibation and Shock, 2017, 36(16): 224-254.)

[3] 于立君, 陈佳, 刘繁明, 等. 改进粒子群算法的PID神经网络解耦控制[J]. 智能系统学报, 2015, 10(5): 699-704.
(Yu L J, Chen J, Liu F M, et al.An improved particle swarm optimization for PID neural network decoupling control[J]. CAAI Trans on Intelligent Systems, 2015, 10(5): 699-704.)

[4] 刘衍民, 隋常玲, 赵庆祯. 基于K-均值聚类的动态多种群粒子群算法及其应用[J]. 控制与决策, 2011, 26(7): 1019-1025.
(Liu Y M, Sui C L, Zhao Q Z. Dynamic multi-swarm particle swarm optimizer based on K-means clustering andits application[J]. Control and Decision, 2011, 26(7): 1019-1025.)

[5] Sha D Y. A new particle swarm optimization for the open shop scheduling problem[J]. Computers& Operations Research, 2008, 35(10): 3243-3261.

[6] 王佺, 聂仁灿, 周冬明, 等. 多目标粒子群优化PCNN参数的图像融合算法[J]. 中国图象图形学报, 2016, 21(10): 1298-1306.
(Wang Q, Nie R C, Zhou D M, et al. Image fusion algorithm using PCNN model parameters of multi-objective particle swarm optimization[J]. J of Image and Graphics, 2016, 21(10): 1298-1306.)

[7] Li W T, Shi X W, Hei Y Q. A hybrid optimization algorithm and its application for conformal array pattern

synthesis[J]. IEEE Trans on Antennas Propagation, 2010, 58(10): 3401-3406.

[8] 吴建辉, 章兢, 李仁发. 多子种群微粒群免疫算法及其在函数优化中应用[J]. 计算机研究与发展, 2012, 49(9): 1883-1898.
(Wu J H, Zhang J, Li R F. A multi-subpopulation pso immune algorithm and its application on function optimization[J]. J of Computer Research and Development, 2012, 49(9): 1883-1898.)

[9] 周凌云, 丁立新, 彭虎. 一种邻域重心反向学习的粒子群优化算法[J]. 电子学报, 2017, 45(11): 2815-2824.
(Zhou L Y, Ding L X, Peng H. Neighborhood centroid opposition-based particle swarm optimization[J]. Acta Electronica Sinica, 2017, 45(11): 2815-2824.)

[10] Juang Y T, Tung S L, Chiu H C. Adaptive fuzzy particle swarm optimization for global optimization of multimodal functions[J]. Information Sciences, 2011, 181(20): 4539-4549.

[11] Meng H, Teresa W, Weir J D. An adaptive particle swarm optimization with multiple adaptive methods[J]. IEEE Trans on Evolutionary Computation, 2013, 17(5): 705-720.

[12] 董文永, 康岚兰, 刘宇航, 等. 带自适应精英扰动及惯性权重的反向粒子群优化算法[J]. 通信学报, 2016, 37(12): 1-10.
(Dong W Y, Kang L L, Liu Y H, et al. An opposition-based particle swarm optimization with adaptive elite mutation and nonlinear inertia weight[J]. J on Communications, 2016, 37(12): 1-10.)

[13] 周新宇, 吴志健, 王晖, 等. 一种精英反向学习的粒子群优化算法[J]. 电子学报, 2013, 41(8): 1647-1652.
(Zhou X Y, Wu Z J, Wang H, et al. Elite opposition-based particle swarm optimization[J]. Acta Electronica Sinica, 2013, 41(8): 1647-1652.)

[14] Li C H, Yang S X, Nguyen T T. A self-learning particle swarm optimizer for global optimization problems[J], IEEE Trans on System, Man, and Cybernetics, 2012, 42(3): 627-646.

[15] Ren Z H, Zhang A M, Wen A M. A scatter learning particle swarm optimization algorithm for multimodal problems[J]. IEEE Trans on System, Man, and Cybernetics, 2014, 44(7): 1127-1140.

[16] Mirjalili S, Hashim S Z M. A new hybrid PSOGSA algorithm for function optimization[C]. Int Conf on Computer and Information Application. Tianjin: IEEE: 2010: 374-377.

[17] Li S, Tan M, Tsang I W. A hybrid PSO-BFGS strategy for global optimization of multimodal functions[J]. IEEE Trans on System, Man, and Cyberntics, 2011, 41(4): 1003-1014.

[18] 刘朝华, 李小花, 章兢. 精英免疫克隆选择的协同进化粒子群算法[J]. 电子学报, 2013, 14(11): 2167-2173.
(Liu Z H, Li X H, Zhang J. Co-evolutionary particle swarm optimization algorithm based on elite immune clonal selection[J]. Acta Electronica Sinica, 2013, 14(11): 2167-2173.)

[19] 周燕, 刘培玉, 赵静. 基于自适应惯性权重的混沌粒子群算法[J]. 山东大学学报: 理学版, 2012, 47(3): 27-32.
(Zhou Y, Liu P Y, Zhao J. Chaos particle swarm optimization based on the adaptive inertia weight[J]. J of Shandong University: Natual Science, 2012, 47(3): 27-32.)

[20] Jiao B, Lian Z, Chen Q. A dynamic global and local combined particle swarm optimization algorithm[J]. Chaos, Solitons & Fractals, 2009, 42(5): 2688-2695.

[21] 李丹. 粒子群优化算法及其应用研究[D]. 沈阳: 东北大学信息科学与工程学院, 2007.
(Li D. Particle swarm optimization algorithm and aPPlieationresearch[D]. Shenyang: College of Information Science and Engineering, Northeastern University, 2007.)

[22] Das S, Mandal A, Mukherjee R. An adaptive differential evolution algorithm for global optimization in dynamic environments[J]. IEEE Trans on Cybernetics, 2014, 44(6): 966-978.

[23] Sheng L S, Shu J L, Li K, et al. Improved particle swarm cooperative optimization algorithm based on chaos & simplex method[C]. Proc of the IEEE Int Conf on Education Technology and Computer Science. Wuhan: IEEE, 2010: 45-48.

[24] Auer P, Cesa-Bianchi N, Fischer P. Finite-time analysis of the multiarmed bandit problem[J]. Machine Learning, 2002, 47(2): 235-256.

[25] Sturtevant N R. An analysis of UCT in multi-player games[C]. Computers and Games. Berlin: Springer-Heidelberg, 2008: 37-49.

[26] 杨启文, 蔡亮, 薛云灿. 差分进化算法综述[J]. 模式识别与人工智能, 2008, 21(4): 506-511.
(Yang Q W, Cai L, Xue Y C. A survey of differential evolution algorithms[J]. Pattern Recognition and Artificial Intelligence, 2008, 21(4): 506-511.)

[27] Shi Y, Eberhart R. Empirical study of particle swarm optimization[C]. Proc of IEEE Int Conf on Evolutionary Computation. Washington: IEEE, 1999: 1945-1950.

[28] Shi Y, Eberhart R. A modified partical swarm optimizer[C]. Proc of IEEE Int Conf on Evolutionary Computation. Anchorage: IEEE, 1998: 69-73.

[29] Huang G B, Zhu Q Y, Siew C K. Extreme learning machine: A new learning scheme of feedforward neural networks[C]. IEEE Int Joint Conf on Neural Networks. Budapest: IEEE, 2004: 985-990.

作者简介

张强(1982−), 男, 副教授, 博士, 从事智能算法、神经网络等研究, E-mail: dqpi_zq@163.com;

李盼池(1969−), 男, 教授, 博士, 从事量子智能优化、过程神经网络等研究, E-mail: lipanchi@vip.sina.com.

(责任编辑：郑晓蕾)