文章编号: 1001-0920 (2012) 06-0833-06

引入人工蜂群搜索算子的粒子群算法

高卫峰[1], 刘三阳[1], 焦合华[1,2], 秦传东[1]

(1. 西安电子科技大学 数学科学系, 西安 710071; 2. 长江师范学院 数学与计算机学院, 重庆 408100)

摘 要: 针对标准粒子群算法易出现早熟现象和收敛速度慢等问题, 提出一种引入人工蜂群搜索算子的粒子群算法. 首先利用人工蜂群搜索算子很强的探索能力, 对粒子搜索到的历史最优位置进行搜索以帮助算法快速跳出局部最优点; 然后, 为了提高算法的全局收敛速度, 提出一种基于混沌和反学习的初始化方法. 通过 12 个标准测试函数的仿真实验并与其他算法相比较, 所得结果表明所提出的算法具有较快的收敛速度和很强的跳出局部最优的能力.

关键词: 粒子群优化; 种群初始化; 搜索算子; 人工蜂群算法

中图分类号: TP18 **文献标识码:** A

Particle swarm optimization with search operator of artificial bee colony algorithm

GAO Wei-feng[1], *LIU San-yang*[1], *JIAO He-hua*[1,2], *QIN Chuan-dong*[1]

(1. Department of Applied Mathematics, Xidian University, Xi'an 710071, China; 2. College of Mathematics and Computer Science, Yangtze Normal University, Chongqing 408100, China. Correspondent: GAO Wei-feng, E-mail: gaoweifeng2004@126.com)

Abstract: To the problems of premature convergence frequently appeared in standard particle swarm optimization(PSO) algorithm and its poor convergence, a particle swarm optimization algorithm with the search operator of artificial bee colony is proposed. Firstly, the method makes full use of the exploration of artificial bee colony algorithm search operator to help the algorithm to jump out of the likely local optima. Then to enhance the global convergence, when producing the initial population, both chaotic maps and opposition-based learning based method is proposed. Moreover, simulation experiment on a suite of 12 benchmark functions is given, and the comparisons with other algorithms are provided. The results show that the proposed approach has better convergence rate and great capability of preventing premature convergence.

Key words: particle swarm optimization; population initialization; search operator; artificial bee colony algorithm

1 引 言

Kennedy 和 Eberhart[1]于 1995 年提出的粒子群优化 (PSO) 算法是继遗传算法、蚁群算法之后的又一种新的群体智能优化算法. PSO 算法来源于对聚集生物行为 (如鸟群和鱼群的觅食行为)的研究. 由于结构简单、易于实现、无需梯度信息和参数较少等特点, 算法一经提出便受到众多学者的关注和研究, 并在函数优化、多目标优化、模式识别以及系统辨识等诸多领域得到广泛应用. 与其他智能算法类似, 标准 PSO 算法也存在易于过早陷入局部最优点、进化后期收敛速度慢、对过于复杂的问题可能搜索不到最优解以及计算精度不高等问题. 这些缺点限制了 PSO 在实际中的应用.

针对上述缺点, 人们提出了许多改进策略[2-9]来优化 PSO 的性能. 文献 [2] 提出一种协作 PSO 算法来提高标准 PSO 算法的性能; [3] 在自组织算法的基础上给出一种变异操作随时间变化的自适应层次 PSO 算法; [4] 引入动态惯性权重来提高算法的收敛速度; [5] 通过将混沌系统嵌入 PSO 的机制中来避免早熟收敛; [6] 利用单纯形法的快速收敛性, 提出了混合单纯形粒子群算法; [7] 指出早熟是由于粒子速度降低而失去继续搜索可行解的能力, 进而提出了一种基于种群速度动态改变惯性权重的粒子群算法. 然而, 到目前为止都很难在提高算法收敛速度和避免早熟收

收稿日期: 2010-12-14; **修回日期:** 2011-03-03.

基金项目: 国家自然科学基金项目(60974082); 中央高校基本科研业务费专项基金项目(50510700004).

作者简介: 高卫峰(1985–), 男, 博士生, 从事进化计算、最优化方法的研究; 刘三阳(1959–), 男, 教授, 博士生导师, 从事智能信息处理、最优化方法等研究.

敛两方面取得平衡. 例如, 基于广泛学习的PSO算法(LPSO)[8]侧重于避免早熟收敛, 却以牺牲算法收敛速度为代价; 而自适应的PSO算法(APSO)[9]希望能在这两方面取得平衡, 虽然取得了较好的结果, 但对于某些测试函数仍不能成功地跳出局部最优, 如Griewank函数.

为了克服早熟、提高收敛速度和寻优精度, 本文提出一种新的改进粒子群算法. 首先, 设计一个新的初始化方法——混沌反学习的初始化方法, 以加快算法的全局收敛速度; 然后, 在标准粒子群算法中引入人工蜂群算法搜索算子, 对粒子搜索到的历史最优位置进行引导搜索, 从而使粒子尽快跳出局部最优点. 该方法能提高算法的探索和开发能力, 避免算法因陷入局部最优而导致早熟收敛. 大量的仿真实验表明, 该算法能显著提高优化效率和优化性能.

2 标准PSO算法

标准PSO算法是模拟鸟群飞行觅食的行为, 通过个体之间的协作来寻找最优解的进化计算技术. 设在D维空间中, 有M个粒子组成一个群落, 其中第i个粒子的位置$\boldsymbol{x}_i=[x_{i1},x_{i2},\cdots,x_{iD}]^{\mathrm{T}}$, 速度$\boldsymbol{v}_i=[v_{i1},v_{i2},\cdots,v_{iD}]^{\mathrm{T}}$, $i=1,2,\cdots,M$. 第i个粒子搜索到的历史最优位置为$\boldsymbol{p}_i$, 整个粒子群搜索到的最优位置为$\boldsymbol{p}_g$.

将$\boldsymbol{x}_i$代入目标函数, 计算其适应值. 对于第$k+1$次迭代, 每个粒子根据下式更新自己的速度和位值:

$$v_{i,d}^{k+1}=\omega v_{i,d}^{k}+c_1r_1(p_{i,d}-x_{i,d}^{k})+ c_2r_2(p_{g,d}-x_{i,d}^{k}), \tag{1}$$

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

其中: $d=1,2,\cdots,D$; ω为惯性权重; c_1和c_2分别为认知参数和社会参数; r_1和r_2为[0,1]之间的随机数.

3 引入人工蜂群搜索算子的粒子群算法

3.1 一种新的初始化方法

种群初始化在进化算法中显得尤为重要, 因为它影响着算法的全局收敛速度和解的质量. 在没有任何先验信息可利用的情况下, 一般采用随机初始化的方法产生初始解. 最近, 有学者提出了混沌初始化[10]和基于反学习[11]的初始化方法, 均取得了较好的计算结果. 本文利用这两种初始化方法的优点, 提出一种新的初始化方法——混沌反学习的初始化方法. 具体步骤如下:

算法1 混沌反学习的初始化方法.

1) 设置最大混沌迭代步数$K\geqslant 300$和种群规模M.

{混沌阶段}

2) for $i=1$ to M do

3) for $j=1$ to D do

4) 随机产生$\mathrm{ch}_{0,j}\in(0,1)$

5) for $k=1$ to K do

6) $\mathrm{ch}_{k,j}=\sin(\pi\mathrm{ch}_{k-1,j})$

7) end for

8) $x_{i,j}=x_{\min,j}+\mathrm{ch}_{k,j}(x_{\max,j}-x_{\min,j})$

9) end for

10) end for

{反学习阶段}

11) for $i=1$ to M do

12) for $j=1$ to D do

13) $\mathrm{ox}_{i,j}=x_{\min,j}+x_{\max,j}-x_{i,j}$

14) end for

15) end for

16) 从$\{X(M)\bigcup \mathrm{OX}(M)\}$中选择适应值最好的$M$个粒子作为初始种群.

3.2 搜索算子

在标准粒子群算法中, $\boldsymbol{p}_i$ $(i=1,2,\cdots,M)$引导着粒子的搜索轨迹. 当优化复杂的多峰问题时, 如果$\boldsymbol{p}_i$陷入局部最优跳不出去, 则它将会把粒子i向局部最优区域引导. 当有很多个粒子被引导到局部最优区域时, 极容易导致算法出现早熟现象, 即标准粒子群算法的探索能力较弱, 这是导致算法容易出现早熟的根源所在. 因此, 如何提高标准粒子群算法的探索能力以帮助算法跳出局部最优是提高算法性能的关键.

Karaboga[12]于2006年提出了一种新的智能优化算法——人工蜂群算法(ABC), 模拟了蜜蜂群的智能采蜜行为, 而且表明ABC在优化复杂的多峰问题时具有很好的优化性能. 这主要得力于下式所示的ABC搜索算子具有很强的探索能力:

$$z_{i,j}=x_{i,j}+\phi_{i,j}(x_{i,j}-x_{k,j}). \tag{3}$$

其中: $k\in\{1,2,\cdots,M\}$和$j\in\{1,2,\cdots,D\}$为随机选取的指标, 并且k不等于i; $\phi_{i,j}$为$[-1,1]$之间的随机数. 正是这些参数的随机性保证了算法具有很强的探索能力.

人工蜂群算法的搜索算子侧重于提高探索能力, 却以牺牲算法的开发能力为代价. 为此, Zhu等人[13]受PSO的启发, 给出了一个新的搜索算子, 即

$$z_{i,j}=x_{i,j}+\phi_{i,j}(x_{i,j}-x_{k,j})+\psi_{i,j}(p_{g,j}-x_{i,j}). \tag{4}$$

其中: k, j和$\phi_{i,j}$的选取与式(3)相同; $\psi_{i,j}$为$[0,1.5]$之间的随机数. 可以看出, 新的搜索算子由于有最优位置$\boldsymbol{p}_g$的引导, 在保证探索能力的同时能够提高开发能力.

基于上面所述, 本文利用ABC搜索算子探索能力, 引导 $\boldsymbol{p}_i$ $(i = 1, 2, \cdots, M)$ 快速跳出局部最优, 进而达到避免算法早熟的目的.

3.3 引入人工蜂群搜索算子的粒子群算法

标准粒子群算法在寻优过程中存在收敛速度慢、容易出现早熟的缺点, 严重影响了算法的性能. 对此, 本文提出一种引入人工蜂群搜索算子的粒子群算法. 其基本思想是: 在PSO算法的迭代中引入搜索算子, 对整个粒子群搜索到的历史最优位置 $\boldsymbol{p}_i$ $(i = 1, 2, \cdots, M)$ 进行搜索, 使其快速跳出局部最优位置, 进而达到避免早熟的目的. 其次, 为了提高算法的收敛速度和全局寻优的能力, 本文提出一种新的初始化方法, 即混沌反学习的初始化方法. 这些操作使算法在提高收敛速度和避免早熟两方面取得平衡, 从而其性能得到大幅度的提高. 算法的具体步骤如下:

算法 2　引入人工蜂群搜索算子的粒子群算法.

1) 初始化. 设定影响因子 c_1, c_2, 惯性权重 ω, 最大函数评价次数 MFE.

2) 在定义空间中, 用混沌反学习的初始化方法产生 M 个粒子组成初始种群 $\boldsymbol{x}_i$, 同时产生各粒子的初始速度 $\boldsymbol{v}_i$, $i = 1, 2, \cdots, M$.

3) 计算各粒子的适应度函数值, 更新各粒子的 $\boldsymbol{p}_i$ 和 $\boldsymbol{p}_g$.

4) while 算法的停止条件不满足 do

5)　for $i = 1$ to M do

6)　for $d = 1$ to D do

7)　依据式 (1) 更新粒子速度,

8)　依据式 (2) 更新粒子空间位置.

9)　end for

10)　if $f(\boldsymbol{v}_i) < f(\boldsymbol{p}_i)$ do

11)　$\boldsymbol{p}_i = \boldsymbol{v}_i$.

12)　end if

13)　for $k = 1$ to K do

14)　用式 (3) 或 (4) 在 $\boldsymbol{p}_i$ 周围搜索候选解 $\boldsymbol{z}_i$.

15)　if $f(\boldsymbol{z}_i) < f(\boldsymbol{p}_i)$ do

16)　$\boldsymbol{p}_i = \boldsymbol{z}_i, \boldsymbol{x}_i = \boldsymbol{z}_i$.

17)　end if

18)　end for

19)　if $f(\boldsymbol{p}_i) < f(\boldsymbol{p}_g)$ do

20)　$\boldsymbol{p}_g = \boldsymbol{p}_i$.

21)　end if

22)　end for

23) end while

24) 输出最优解及最优值.

注 1　本文将所提出的引入人工蜂群搜索算子的粒子群算法简记为: PSOWS1 表示采用搜索算子 (3), PSOWS2 表示采用搜索算子 (4).

4 仿真实验

为了测试本文提出的改进粒子群算法的效果, 对文献 [8] 中的 12 个标准测试函数进行仿真实验. 其中: f_1 和 f_2 为单峰函数, $f_3 \sim f_{12}$ 为多峰函数. 表 1 给出了 12 个测试函数的名称、维数、搜索空间范围和最优值. 种群规模为 30, $K = 3$. ω, c_1, c_2 采用 [14] 的参数设置方案, 即 $\omega:\ 0.9 \to 0.4$, $c_1 = c_2 = 2$. 将本文算法与标准的粒子群算法比较, 其参数设置见 [14]. 最大函数评价次数为 150 000. 采用 Matlab7.0 编程工具进行仿真实验. Best, Median, Worst, Mean 和 Std 分别为算法独立实验 20 次的最好值、中间值、最差值、平均值和标准差. Best, Median 和 Worst 反映了解的质量; Mean 显示了在给定函数评价次数下算法所能达到的精度, 反映了算法的收敛速度; Std 反映了算法的稳定性和鲁棒性. 结果如表 2 所示.

表 1　测试函数的维数、搜索空间和最优值

函数	名称	维数	搜索空间	最优值
f_1	Sphere	30	$[-100, 100]$	0
f_2	Rosebrock	30	$[-30, 30]$	0
f_3	Schwefel	30	$[-500, 500]$	$-12\,569.5$
f_4	Rastrigin	30	$[-5.12, 5.12]$	0
f_5	NCRastrigin	30	$[-5.12, 5.12]$	0
f_6	Griewank	30	$[-600, 600]$	0
f_7	Ackley	30	$[-32, 32]$	0
f_8	Penalized	30	$[-50, 50]$	0
f_9	Penalized2	30	$[-50, 50]$	0
f_{10}	Levy	30	$[-10, 10]$	0
f_{11}	Shifted Rastrigin	30	$[-5.12, 5.12]$	0
f_{12}	Shifted Griewank	30	$[-600, 600]$	0

由表 2 数据对比可以看出, 在大部分标准测试函数中, 无论是解的质量, 还是算法的收敛精度和稳定性, PSOWS1 和 PSOWS2 算法都比标准 PSO 算法有了很大的提高. 特别地, 在这 3 种算法中, PSOWS2 算法几乎在所有的测试函数上都取得了最好的优化结果, 除了在 Rosebrock 函数上, PSOWS1 算法求解的结果最好. 对于 PSOWS1 算法而言, 除了在 Sphere 和 Ackley 函数上性能略逊于标准 PSO 算法, 而在其他测试函数上其性能要好于标准 PSO 算法. 这并不意味标准 PSO 算法一无是处, 从表 2 的第 3 列可以看出, 标准 PSO 算法对一些测试函数也可以得到理论最优值, 例如 Griewank 函数. 只是标准 PSO 算法的鲁棒性太差, 容易陷入局部最优而出现早熟现象. 这也正是

表 2 3种算法对12个函数的计算结果比较

Function	Algorithm	Best	Median	Worst	Mean	Std
	PSO	3.55e−031	2.22e−030	2.77e−027	2.24e−028	6.88e−028
f_1	PSOWS1	5.46e−028	1.01e−026	1.10e−025	2.53e−026	3.36e−026
	PSOWS2	2.56e−040	2.67e−040	2.18e−038	2.29e−039	4.64e−039
	PSO	18.904 212	34.823 516	58.702 488	38.206 371	9.239 714
f_2	PSOWS1	0.006 749	0.026 615	0.252 494	0.094 765	0.088 399
	PSOWS2	2.45e−004	3.989 642	4.360 077	0.592 698	1.271 252
	PSO	−10 477.070 1	−9 687.454 4	−9 035.999 8	−9 882.206 1	4.53e+002
f_3	PSOWS1	−12 569.486 6	−12 569.486 6	−12 569.486 6	−12 569.486 6	3.45e−012
	PSOWS2	−12 569.486 6	−12 569.486 6	−12 569.486 6	−12 569.486 6	3.23e−012
	PSO	12.934 467	27.858 833	45.768 056	28.455 806	8.154 605
f_4	PSOWS1	0	0	1.77e−014	8.88e−015	3.87e−016
	PSOWS2	0	0	0	0	0
	PSO	2.538 002	6.061 213	24.358 250	15.800 278	7.286 191
f_5	PSOWS1	0	0	0	0	0
	PSOWS2	0	0	0	0	0
	PSO	0	0.031 971	0.036 857	0.012 468	0.011 263
f_6	PSOWS1	0	0	0	0	0
	PSOWS2	0	0	0	0	0
	PSO	1.50e−014	1.50e−014	7.54e−014	2.50e−014	1.41e−014
f_7	PSOWS1	4.70e−014	7.54e−014	3.34e−013	1.54e−013	8.03e−014
	PSOWS2	7.99e−015	1.15e−014	1.50e−014	1.35e−014	2.85e−015
	PSO	6.47e−029	0.103 669	0.207 316	0.034 554	0.061 812
f_8	PSOWS1	6.32e−029	2.41e−028	7.23e−027	2.84e−027	2.31e−028
	PSOWS2	1.57e−032	1.57e−032	1.57e−032	1.57e−032	2.73e−048
	PSO	2.14e−030	9.93e−027	0.010 987	0.002 197	0.004 394
f_9	PSOWS1	1.81e−028	3.00e−026	3.50e−026	1.27e−026	1.03e−026
	PSOWS2	1.35e−032	1.35e−032	1.47e−032	1.36e−032	3.69e−034
	PSO	2.70e−030	2.88e−026	0.109 873	0.007 324	0.027 407
f_{10}	PSOWS1	1.19e−028	1.95e−028	5.77e−027	2.29e−027	2.06e−027
	PSOWS2	1.35e−031	1.35e−031	1.35e−031	1.35e−031	0
	PSO	24.873 961	52.732 739	54.950 089	38.964 922	10.282 206
f_{11}	PSOWS1	0	0	1.77e−015	1.77e−016	5.32e−016
	PSOWS2	0	0	0	0	0
	PSO	0	9.818 906	15.863 461	4.309 699	5.017 692
f_{12}	PSOWS1	0	0	0	0	0
	PSOWS2	0	0	0	0	0

很多学者研究标准PSO算法的意义所在.

为更直观地反映算法的寻优效果, 将PSOWS1算法, PSOWS2算法和标准PSO算法进行比较. 3种算法对相关测试函数的收敛曲线如图1∼图5所示. 由图可知, 本文提出的粒子群算法由于引入了搜索算子和新的初始化方法, 使算法在处理多峰函数时能很快跳出局部最优, 收敛到全局最优; 在处理单峰函数时有较快的收敛速度. 然而, 由于PSOWS1算法的搜索算子没有最优位置$\boldsymbol{p}_g$的引导而降低了算法的开发能力, 在处理多峰函数时虽能跳出局部最优, 但其收敛速度较PSOWS2算法慢, 如图3∼图5所示. 因此, 可以得出结论: 本文所提出的PSOWS2算法的优化能力整体上强于标准PSO和PSOWS1算法.

为进一步展示PSOWS2算法的先进性, 本文将PSOWS2算法与VPSO [14], FIPS [15], HPSO-TVAC [3], DMS-PSO [16], CLPSO [8]和APSO [9]算法的结果进行比较, 如表3所示. 为了公平起见, 算法的最大函数评价次数与文献[9]相同, 即200 000. 其他算法的计算结果也直接引自文献[9]. 从表3中可以看出, 对于除f_1外的6个函数, PSOWS2算法的性能都有较大幅度的改善, 优化结果和理论值的误差最小.

为了验证本文所提的初始化方法的效果, 对PSOWS1和PSOWS2算法在不同初始化方法下进行实验仿真. 表4给出了这两个算法独立实验20次达到

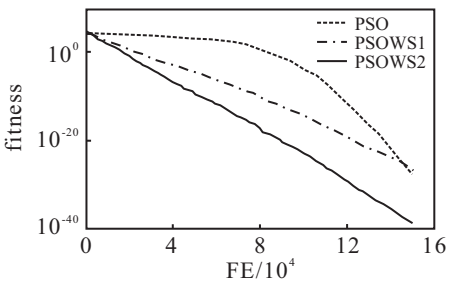

图 1 f_1函数的收敛性能比较

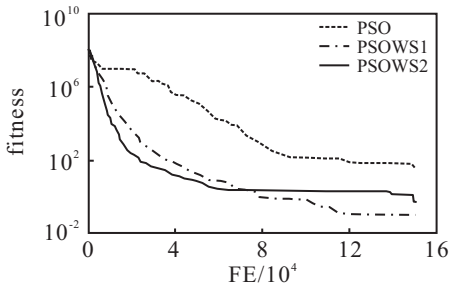

图 2　f_2 函数的收敛性能比较

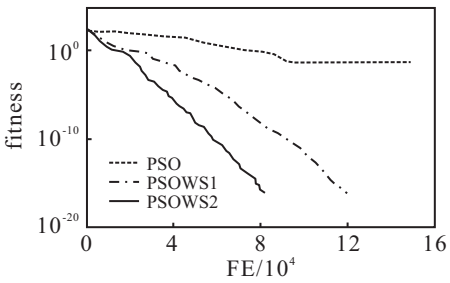

图 4　f_6 函数的收敛性能比较

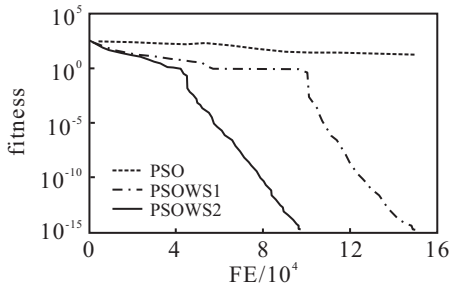

图 3　f_4 函数的收敛性能比较

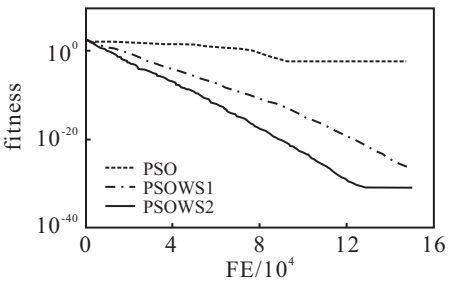

图 5　f_{10} 函数的收敛性能比较

表 3　7 种算法对 7 个函数的计算结果比较

Function	Index	VPSO	FIPS	HPSO-TVAC	DMS-PSO	CLPSO	APSO	PSOWS2
f_1	Mean	5.11e−038	3.21e−030	3.38e−041	2.65e−031	1.89e−019	1.45e−150	7.66e−054
	Std	1.91e−037	3.60e−030	8.50e−041	6.25e−031	1.49e−019	5.73e−150	1.06e−053
f_2	Mean	37.646 9	22.538 7	13	32.3	11	2.84	0.178 124
	Std	24.937 8	0.310 2	16.5	24.1	14.5	3.27	0.175 501
f_3	Mean	−9 845.27	−10 113.8	−10 868.57	−9 593.33	−12 557.65	−12 569.5	−12 569.5
	Std	588.87	889.58	289	441	36.2	5.22e−011	1.40e−012
f_4	Mean	34.09	29.98	2.39	28.1	2.57e−011	5.8e−015	0
	Std	8.07	10.92	3.71	6.42	6.64e−011	1.01e−014	0
f_5	Mean	21.33	35.91	1.83	32.8	0.167	4.14e−016	0
	Std	9.46	9.49	2.65	6.49	0.379	1.45e−015	0
f_6	Mean	1.31e−002	9.04e−004	1.07e−002	1.13e−002	6.45e−013	1.67e−002	0
	Std	1.35e−002	2.78e−003	1.41e−002	1.73e−002	2.07e−012	2.41e−002	0
f_8	Mean	3.46e−003	1.22e−031	7.07e−030	2.05e−032	1.59e−021	3.76e−031	1.57e−032
	Std	1.89e−002	4.85e−032	4.05e−030	8.12e−033	1.93e−021	1.2e−030	2.73e−048

表 4　初始化方法对本文所提两种算法性能的影响

函数	误差限	初始化方法	PSOWS2	PSOWS1
f_1	0.01	随机	21 532	37 308
		本文	21 216	37 031
f_2	100	随机	28 507	40 709
		本文	26 795	38 755
f_4	50	随机	9 667	16 298
		本文	9 062	11 688
f_6	50	随机	3 532	5 703
		本文	3 194	5 250
f_7	0.01	随机	30 183	51 175
		本文	29 426	49 621
f_8	0.01	随机	19 181	36 091
		本文	18 670	30 119

误差限[9]所需的平均函数评价次数. 从表 4 中可以看出, 达到相同的误差限, 本文的初始化方法所需的函数评价次数明显小于随机初始化方法. 同时, 表 4 进一步表明了 PSOWS2 算法的性能优于 PSOWS1 算法.

5　结　论

为了解决标准粒子群算法早熟收敛和收敛速度慢等问题, 本文提出了一种引入人工蜂群搜索算子的粒子群算法. 为了平衡算法的探索和开发能力, 改进的算法引入了搜索算子和新的初始化方法, 从而既克服了早熟现象, 又提高了算法的收敛速度. 对 12 个函数寻优的实验结果表明, 本文算法在优化效率、优化性能和鲁棒性方面均比标准 PSO 及其他改进的粒子群算法有了较大的改善.

新算法也有自身的局限性. 由于在标准的 PSO 算法中引入了探索能力很强的搜索算子, 在一定程度上忽视了算法的开发能力, 从表 2 可以观察到在优化 Sphere 函数时性能要逊于 APSO. 但从算法的整体性能看, 本文提出的算法通过搜索算子与新的初始化方法的相互作用, 使算法的性能得到了改善, 而且取得

了令人满意的结果. 如何在保证算法快速跳出局部最优的同时又具有较高的收敛速度, 将是下一步的研究内容.

参考文献(References)

[1] Kennedy J, Eberhartr C. Particle swarm optimization[C]. Proc of IEEE Int Conf on Neural Networks. Perth: IEEE Piscataway, 1995: 1942-1948.

[2] Bergh F V D, Engelbreeht A P. A cooperative approaeh to partiele swarm optimization[J]. IEEE Trans on Evolutionary Computation, 2004, 8(3): 225-239.

[3] Ratnaweera A, Halgamuge S K, Waton H C. Self-organizing hierarehieal partiele swarm optimizer with time-varying acceleration coeffieients[J]. IEEE Trans on Evolutionary Computation, 2004, 8(3): 240-255.

[4] Jiao B, Lian Z G, Gu X S. A dynamic inertia weight particle swarm optimization algorithm[J]. Chaos Solitons & Fractals, 2008, 37(3): 698-705.

[5] Jiang C W, Etorre B. A hybrid method of chaotic particle swarm optimization and linear interior for reactive power optimization[J]. Mathematics and Computers in Simulation, 2005, 68(1): 57-65.

[6] Fan S, Zahara E. Hybrid simplex search and particle swarm optimization for unconstratined optimization problems[J]. European J of Operational Research, 2007, 181(2): 527-548.

[7] 张顶学, 廖锐全. 一种基于种群速度的自适应粒子群算法[J]. 控制与决策, 2009, 23(7): 756-761.
(Zhang D X, Liao R Q. Adaptive particle swarm optimization algorithm based on population velocity[J]. Control and Decision, 2009, 23(7): 756-761.)

[8] Liang J J, Qin A K, Suganthan P N, et al. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions[J]. IEEE Trans on Evolutionary Computation, 2006, 10(3): 281-295.

[9] Zhan Z H, Zhang J, Li Y, et al. Adaptive particle swarm optimization[J]. IEEE Trans on Systems, Man, and Cybernetics: Part B, 2009, 39(6): 1362-1381.

[10] Liu B, Wang L, Jin Y H, et al. Improved particle swarm optimization combined with chaos[J]. Chaos, Solitons & Fractals, 2005, 25(2): 1261-1271.

[11] Rahnama S, Tizhoosh H R, Salama M M A. Opposition-based differential evolution[J]. IEEE Trans on Evolutionary Computation, 2008, 12(1): 64-79.

[12] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony(ABC) algorithm[J]. J of Global Optimization, 2007, 39(3): 459-171.

[13] Zhu G P, Kwong S. Gbest-guided artificial bee colony algorithm for numerical function optimization[J]. Applied Mathematics and Computation, 2010, 217(7): 3166-3173.

[14] Kennedy J, Mendes R. Population structure and particle swarm performance[C]. Proc of IEEE Congress on Evolutionary Computation. Honolulu, 2002: 1671-1676.

[15] Mendes R, Kennedy J, Neves J. The fully informed particle swarm: Simpler maybe better[J]. IEEE Trans on Evolutionary Computation, 2004, 8(3): 204-210.

[16] Liang J J, Suganthan P N. Dynamic multi-swarm particle swarm optimizer[C]. Proc of IEEE Swarm Intelligence Symposium. Pasadena. 2005: 124-129.

(上接第832页)

[7] Nakashima T, Ariyama T, Yoshida T, et al. Performance evaluation of combined cellular genetic algorithms for function optimization problems[C]. Proc of IEEE Int Computational Intelligence in Robotics and Automation. Kobe, 2003: 295-299.

[8] 张俞, 黎明, 鲁宇明. 元胞遗传算法演化规则的研究[J]. 计算机应用研究, 2009, 26(10): 3635-3638.
(Zhang Y, Li M, Lu Y M. Study on evolution rules of op timization genetic algorithm with cellular automata[J]. Application Research of Computers, 2009, 26(10): 3635-3638.)

[9] Li Xiaodong. A real-coded predator-prey genetic algorithm for multiobjective optimization[C]. Proc of the 2nd Int Conf on Evolutionary Multi-criterion Optimization. Portugal, 2003: 207-221.

[10] Deb K, Bhaskara U. Investigating predator-prey algorithms for multi-objective optimization[R]. Kanpur: Department of Mechanical Engineering, Indian Institute of Technology Kanpur, 2005.

[11] 蒋忠中, 汪定伟. 有时间窗车辆路径问题的捕食搜索算法[J]. 控制与决策, 2007, 22(1): 59-68.
(Jiang Z Z, Wang D W. Predatory search algorithm for vehicle routing problem with time windows[J]. Control and Decision, 2007, 22(1): 59-68.)

[12] Yang Shengxiang. Constructing dynamic test environments for genetic algorithms based on problem difficulty[C]. Proc of the 2004 Congress on Evolutionary Computation. Seattle, 2004: 1262-1269.