一种快速收敛的非参数粒子群优化算法及其收敛性分析

刘兆广[1],纪秀花[1],刘云霞[2]
(1. 山东财经大学计算机科学与技术学院,山东济南 250014; 2. 济南大学信息科学与技术学院,山东济南 250022)

摘 要: 如何调整粒子群算法的参数引起了大量研究人员的关注. 本文提出了一种快速收敛的非参数粒子群优化算法. 为了平衡全局搜索和局部搜索,本文算法融合了基于 exemplar 的学习策略和多交叉操作. 根据进一步的稳定性分析,粒子群收敛于搜索空间中的一个固定位置,同时粒子群的位置方差收敛于零点. 本文收集了常用的 24 个准则函数,与 7 个类似的粒子群算法进行了比较. 实验结果表明,本文搜索算法在大部分准则函数上的搜索性能均优于同类算法. 同时本文算法在收敛速度上要远优于同类算法.

关键词: 粒子群优化算法; 交叉操作; 参数选择; 稳定性分析
中图分类号: TP18 **文献标识码:** A **文章编号:** 0372-2112 (2018)07-1669-06
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2018.07.019

A Non-parameter Particle Swarm Optimization Algorithm with Fast Convergence Speed and Its Stability Analysis

LIU Zhao-guang[1], JI Xiu-hua[1], LIU Yun-xia[2]
(1. *School of Computer Science and Technology, Shandong University of Finance and Economics, Jinan, Shandong* 250014, *China*;
2. *School of Information Science and Engineering, University of Jinan, Jinan, Shandong* 250022, *China*)

Abstract: The adjustment of parameters in particle swarm optimization (PSO) has attracted the attention of many researchers. In the paper, an alternative technology, a non-parameter PSO algorithm with fast convergence speed is proposed. A multi-crossover operation and an exemplar-based learning strategy are combined with the proposed algorithm. According to the first-and second-order stability analyses conducted for the present study, the particle positions are expected to converge at a fixed point in the search space, and the variance of the particle positions converge at zero. In our experiments, we compared the proposed algorithm with 7 other advanced PSO algorithms using 24 widely used benchmark functions. The experimental results indicate that the proposed algorithm yields better solution accuracy than the other PSO algorithms. In particular, the proposed algorithm outperforms the other PSO approaches significantly in terms of the convergence speed.

Key words: particle swarm optimization; crossover operation; parameter selection; stability analysis

1 引言

在粒子群算法中,设 X_i 和 V_i 分别为粒子 $i(i=1,2,\cdots,N)$ 的当前位置和速度,则粒子 i 在第 $t(t=1,2,\cdots,T)$ 代的进化方程为:

$$v_i^d(t+1)=wv_i^d(t)+c_1r_1(p_i^d(t)-x_i^d(t))+c_2r_2(g^d(t)-x_i^d(t)) \quad (1)$$

$$x_i^d(t+1)=x_i^d(t)+v_i^d(t+1) \quad (2)$$

其中,$d(d=1,2,\cdots,D)$ 为第 d 维,D 为总维数,N 为粒子个数,T 为迭代代数,p_i 为第 i 个粒子的个体最佳经历($pbest_i$),g 为全局最佳经历($gbest$). c_1 和 c_2 为加速系数,r_1 和 r_2 服从分布 $U(0,1)$,w 为权重系数.

2 背景知识

2.1 基于 exemplar 的学习策略

多个粒子群算法都采用了基于 exemplar 的学习策略,如 CLPSO[1],CCPSO[2],SLPSO[3]. 如何选择 exemplar 成为这类算法的主要研究内容. 比如在 CLPSO[1] 中首

收稿日期:2017-05-05;修回日期:2017-10-19;责任编辑:李勇锋
基金项目:国家自然科学基金(No. 61772309);山东省自然科学基金(No. 2016GGX101022, No. ZR2017MF067)

先从所有粒子中任意选择两个,然后比较这两个粒子的 *pbest* 目标函数值. 最后选择具有较优结果的粒子作为 *exemplar*. *CCPSO*[2] 利用一个称为“黑板”的共享机制. 在迭代中,所有粒子把搜索到的 *pbest* 都共享到“黑板”上. 粒子在更新速度时,从“黑板”上按一定规则选择粒子作为 exemplar.

2.2 粒子群参数选择

经典粒子群算法含有三个参数:w,c_1,c_2. 如何选择这些参数就成为不少研究人员的关注点,比如彭宇[3]采用方差分析,针对单个准则函数,对参数选择进行了分析. w 为权重因子[5],它用来平衡全局搜索和局部搜索能力. 不少研究人员提出了各种调整 w 的方法,如线性变化[6],随机值[7],自适应变化[8]. 加速系数 c_1 和 c_2 用来调整 *pbest* 和 *gbest* 对粒子飞行速度的影响. 较大的 c_1 值意味着 *pbest* 对粒子的吸引力更大;较大的 c_2 值意味着 *gbest* 对粒子的吸引力更大. 为了能调整 c_1 和 c_2,各种方法也已经被提出,如固定值[6],线性变化[9],随机值[7],自适应变化[8,9].

除了参数调整,另一个替代方案为非参数的粒子群算法(NPPSO). Beheshti[10]尝试提出了一种 NPPSO 算法. 为了更新粒子的速度,NPPSO 采用了粒子邻域的局部最优结果,*gbest* 项则用来指导位置更新. 本文提出了一种新的非参数粒子群优化算法. 该算法不仅舍弃了粒子群的三个参数,同时省略了速度项,进一步简化了粒子群的更新过程. 另外,还给出了算法的稳定性分析.

3 快速收敛的非参数粒子群优化算法(FNPPSO)

3.1 多交叉操作

多交叉操作由 HEPSO[11]引入粒子群算法. 本文进一步简化该操作,删除其中的参数 c,并省略粒子群中的速度项.

$$x_i^d(t+1)=x_i^d(t)+r^d\left(g^d(t)-p_i^d(t)-x_i^d(t)\right) \tag{3}$$

图 1 给出了等式(3)位置更新示意. 由图中所示可以看出,粒子在更新位置时,每次都会从当前的方向往 $gbest\text{-}pbest_i$ 方向进行调整,因此,不会出现经典粒子群算法位置更新中的“振荡”现象[12],从而具有更快的收敛速度.

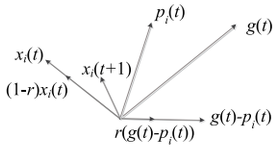


图1 多交叉操作位置更新

3.2 基于 exemplar 的学习策略

本文采用基于 exemplar 的学习策略来更新每个粒子的 *pbest* 位置. 假设 ex_i 为第 i 个粒子的 exemplar,按下式计算中间值 p':

$$p'^d=r_1^d p_i^d(t)+(1-r_1^d)ex_i^d(t)+r_2^d\left(p_i^d(t)-ex_i^d(t)\right) \tag{4}$$

这里 r_1 和 r_2 分别服从分布 $U(0,1)$ 和 $U(-1,1)$.

本文提出的选择 ex_i 和更新 $pbest_i$ 的具体过程见图 2,其中 $F(\cdot)$ 表示目标函数求值.

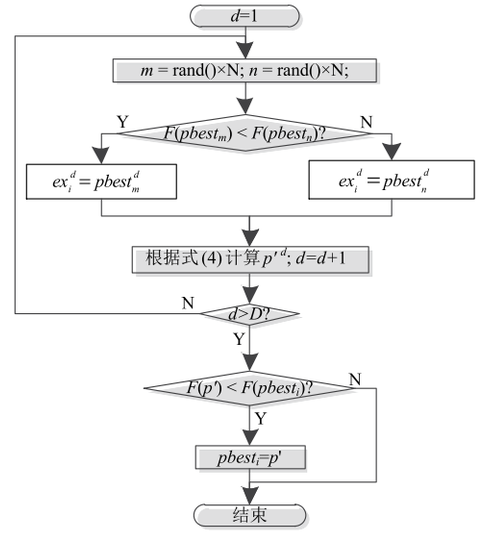


图2 基于exemplar的$pbest_i$更新过程

gbest 项可以引导粒子快速收敛. 但是一旦 *gbest* 陷入局部最优区域,容易引起早熟[1]. 基于这个原因,CLPSO[1]舍弃了 *gbest*,由其他粒子的 *pbest* 代替,从而保证了算法具有很好的全局搜索能力. 但是,由于没有 *gbest* 对粒子飞行的指导,造成了 CLPSO 算法收敛过慢. 本文继承了 CLPSO 中基于 exemplar 的学习策略中的这个优势,从而具有很好的全局搜索能力. *gbest* 项对粒子飞行的指导则由多交叉操作完成.

3.3 FNPPSO 算法步骤

很多研究人员提出了粒子群算法的各种参数控制策略. 如实验方式[6],然而,这种方式不具有推广性. 再如在迭代过程中自适应地调整参数[8]. 这类方法面临的挑战是:在调整过程中,如何有效地判断当前搜索处于哪个阶段(explorative 或 exploitative)[13]. 非参数的粒子群算法则给出了替代方案,即不需要参数调整.

图 3 给出了本文提出的 FNPPSO 算法的流程图. 本文算法实际上可以视为包含两个种群的多种群技术. 在文献[14]中称这两个种群为 current swarm 和 memory swarm. 前者就是所有粒子的群体,即当前粒子群;后者是指所有 pbest 位置组成的种群. 在 FNNPSO 中,当前粒子群采用的是多交叉操作. 由于该操作具有快速收敛性,因此可以完成局部搜索的功能;在 *pbest* 种群中,本

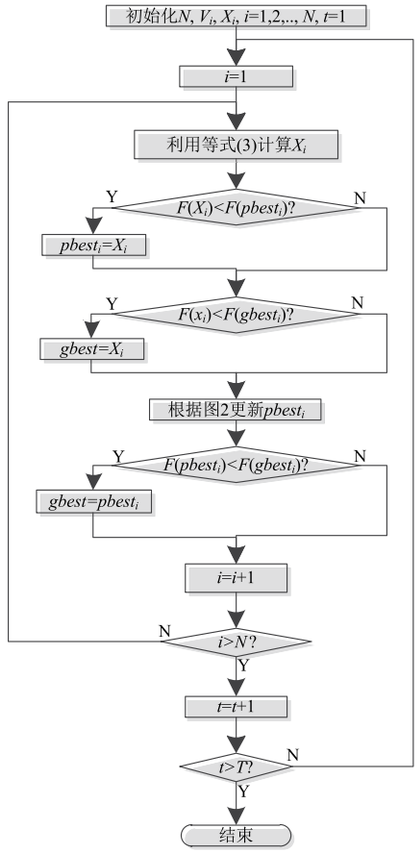


图3　FNPPSO算法流程图

文采用了与 CLPSO 中类似的搜索策略,从而具有很好的全局搜索性能.

3.4　FNPPSO 稳定性分析

在下面的稳定性分析中,假定 *pbest* 位置在一定迭代步骤后停滞不前[15],这样只需要分析多交叉操作影响下粒子的位置变化.

为了便于描述,重写等式(3)如下:

$$y_{t+1}=(1-r)y_t+rA \tag{5}$$

其中,$y_t=x_i(t)$,$A=g(t)-p_i(t)$.

根据文献[16]中的一阶和二阶稳定性的定义,下面分别给出稳定性分析.

(1)一阶稳定性分析

由于 r 服从 $U(0,1)$,所以 $\mu_r=0.5$(变量 r 的均值). 等式(5)两边同时取数学期望,并把迭代等式转换为通项公式.

$$E(y_t)=\frac{1}{2^t}(y_0-A)+A \tag{6}$$

其中,y_0 是第 i 个粒子的初始位置.

定理 1　等式(6)所描述的序列 $E(y_t)$ 是收敛的,且收敛于点 A.

证明　$E(y_t)$ 收敛于 A 意味着任取 $\forall\varepsilon>0$,总存在一个值 $t'(\varepsilon)$,针对所有 $t>t'(\varepsilon)$,总有:

$$|E(y_t)-A|<\varepsilon \tag{7}$$

考虑等式(6),则

$$\frac{1}{2^t}|(y_0-A)|<\varepsilon \tag{8}$$

因此,可以得到下式

$$t>\log_2\left(\frac{|(y_0-A)|}{\varepsilon}\right) \tag{9}$$

所有满足等式(7)的 $t'(\varepsilon)$,等式(9)必然满足. 得证.

(2)二阶稳定性分析

因为

$$V(y_t)=E(y_t^2)-E^2(y_t) \tag{10}$$

根据等式(5),考虑 $\mu_r=0.5$,变量 y_{t+1}^2 的数学期望为

$$E(y_{t+1}^2)=\frac{1}{3}E(y_t^2)+\frac{1}{3}A\cdot E(y_t)+\frac{1}{3}A^2 \tag{11}$$

根据等式(6),$E^2(y_{t+1})$ 可以由下式计算得到.

$$E^2(y_{t+1})=\frac{1}{4}E^2(y_t)+\frac{1}{2}A\cdot E(y_t)+\frac{1}{4}A^2 \tag{12}$$

将等式(11)和(12)带入等式(10),可以得到下式:

$$V(y_{t+1})=\frac{1}{4}V(y_t)+\frac{1}{12}E((y_t-A)^2) \tag{13}$$

把上式可以转换为通项公式.

$$V(y_t)=\frac{1}{4^t}V(y_0)+E((y_0-A)^2)\left(\frac{1}{3^t}-\frac{1}{4^t}\right) \tag{14}$$

定理 2　等式(14)描述的序列 $V(y_t)$ 是收敛的,且收敛到 0.

证明　由于 $V(y_0)=0$,所以当 $t\to\infty$,$V(y_t)$ 的极限为:

$$\begin{aligned}\lim_{t\to\infty}V(y_t)&=\lim_{t\to\infty}\left(E((y_0-A)^2)\left(\frac{1}{3^t}-\frac{1}{4^t}\right)\right)\\&=E((y_0-A)^2)\cdot\lim_{t\to\infty}\left(\frac{1}{3^t}-\frac{1}{4^t}\right))\\&=0\end{aligned} \tag{15}$$

通过上述分析可以看出,HNPPSO 算法同时满足一阶和二阶稳定.

4　实验结果及讨论

4.1　实验环境设置

本文收集了 24 个常用的准则函数[1,17]. 这些准则函数可以被分为三类:6 个单峰函数,6 个多峰函数,12 个包含了旋转和平移变换的复杂准则函数. 前 12 个函数见表 1,函数 F13-F24 为对应函数的旋转平移结果. 如 F13 和 F24 分别为 F1 和 F12 的旋转平移结果.

本文与 7 种不同的粒子群算法进行了比较,这些算法的参数设置均参考各自文献. 本文对所有算法均采用了相同的维数 $D=30$,粒子群规模为 $N=40$,最大迭代次数为(FEs)2×10^5. 为降低统计误差,每个实验均重复 30 次.

表 1 实验中所用的准则函数

单峰函数		多峰函数	
F1	Sphere	F7	Rosenbrock
F2	Schwefel's P1.2	F8	Rastrigin
F3	Elliptic	F9	Ackley
F4	Bent Cigar	F10	Griewank
F5	Discus	F11	Weierstrass
F6	Different Powers	F12	Generalized Penalized 2

4.2 搜索精度的比较

表 2 给出各算法的均值结果和显著性水平为 0.05 的双侧 Wilcoxon 符号秩检验结果[19]. 符号“+”表示 FNNPSO 算法胜出,“-”表示 FNNPSO 算法性能较差,“=”则表示两个算法性能无明显差异. 在表格的底部,行“w/t/l”给出了所有准则函数下的综合结果. 另外,表中还给出了 Friedman 检验结果[19].

表 2 搜索精度比较,Friedman 检验的 $p = 3.55\mathrm{E}-12$

函数	SPSO[5]	CCPSO[2]	HEPSO[11]	ASDPSO[8]	SLPSO[3]	NPPSO[10]	CSPSO[18]	FNPPSO
F1	0.00E+000 =	0.00E+000 =	0.00E+000 =	1.27E-020 +	0.00E+000 =	8.58E-010 +	0.00E+000 =	0.00E+000
F2	3.42E-321 +	2.84E-028 +	0.00E+000 =	7.53E-119 +	6.94E-008 +	3.90E-011 +	0.00E+000 =	0.00E+000
F3	0.00E+000 =	0.00E+000 =	0.00E+000 =	6.05E-019 +	8.00E+002 +	2.44E+004 +	0.00E+000 =	0.00E+000
F4	0.00E+000 =	0.00E+000 =	0.00E+000 =	5.75E-166 +	0.00E+000 =	5.74E-013 +	0.00E+000 =	0.00E+000
F5	0.00E+000 =	0.00E+000 =	0.00E+000 =	3.81E-257 +	0.00E+000 =	2.31E-011 +	0.00E+000 =	0.00E+000
F6	0.00E+000 =	0.00E+000 =	0.00E+000 =	2.01E-024 +	1.65E-021 +	8.00E+000 +	0.00E+000 =	0.00E+000
F7	1.11E+000 +	1.54E-002 +	1.24E-008 +	3.52E-004 +	4.21E+000 +	3.98E-026 =	0.00E+000 -	4.68E-027
F8	7.04E+000 +	0.00E+000 =	0.00E+000 =	1.82E-014 +	1.52E+001 +	0.00E+000 =	0.00E+000 =	0.00E+000
F9	7.26E-015 +	7.40E-015 +	4.44E-016 -	9.30E-011 +	3.99E-015 +	3.30E-006 +	3.99E-015 =	3.85E-015
F10	2.65E-002 +	0.00E+000 =	0.00E+000 =	5.91E-003 +	2.87E-002 +	2.37E-015 +	0.00E+000 =	0.00E+000
F11	0.00E+000 =	0.00E+000 =	0.00E+000 =	0.00E+000 =	0.00E+000 =	0.00E+000 =	0.00E+000 =	0.00E+000
F12	2.43E-032 +	1.34E-032 =	2.03E-032 +	1.55E-020 +	2.21E-030 +	4.39E-004 +	1.34E-032 =	1.34E-032
F13	1.24E+002 +	2.86E+000 +	1.28E+000 +	1.30E-001 +	3.38E+000 +	3.60E+002 +	2.11E-002 +	4.33E-003
F14	3.61E+002 +	1.40E+002 +	3.25E+002 +	9.54E-001 +	2.53E+001 +	1.33E+002 +	7.27E-001 =	6.66E-001
F15	1.18E+006 +	5.29E+005 +	5.82E+005 +	4.81E+003 +	3.35E+005 +	1.01E+006 +	3.40E+005 +	1.86E+003
F16	1.57E+008 +	2.17E+006 +	1.05E+006 +	1.04E+005 +	2.30E+006 +	2.46E+008 +	1.86E+004 +	3.13E+003
F17	4.00E+002 +	7.44E+002 +	1.65E+003 +	2.22E+002 +	5.56E+002 +	2.15E+002 +	9.66E+001 +	3.41E+001
F18	3.94E+001 +	3.04E+000 +	3.98E+000 +	1.55E-001 +	1.94E+000 +	2.37E+002 +	2.08E-001 +	7.99E-002
F19	5.04E+005 +	7.66E+002 +	1.80E+002 +	5.37E+001 +	6.48E+002 +	3.73E+006 +	3.13E+001 +	2.85E+001
F20	6.48E+002 +	4.02E+002 +	1.45E+002 -	9.01E+002 +	1.81E+002 -	6.78E+002 +	3.32E+002 =	3.17E+002
F21	1.87E+001 +	1.84E+001 +	7.12E+000 =	2.01E+001 +	2.01E+001 +	1.92E+001 +	1.24E+001 +	7.62E+000
F22	7.50E-001 +	5.33E-001 +	3.35E-001 +	2.34E-001 +	6.40E-001 +	7.25E-001 +	7.90E-002 =	7.88E-002
F23	3.21E+001 +	3.15E+001 +	3.22E+001 +	3.11E+001 +	2.14E+001 -	2.56E+001 +	2.83E+001 +	2.25E+001
F24	9.69E+006 +	3.61E+006 +	6.52E+005 +	2.91E+003 -	3.10E+003 =	3.42E+005 +	2.93E+003 +	2.89E+003
w/t/l	18/6/0	15/9/0	12/10/2	22/1/1	17/5/2	21/3/0	9/14/1	~
Friedman 排名	5.79	4.69	4.02	4.96	5.21	6.40	2.83	2.10

由表 2 中的 Friedman 检验结果可以看出,FNPPSO 算法的 Friedman 平均排名为 2.10,是所有算法中的最优结果. 通过 Wilcoxon 符号秩检验可以看出,在大部分的准则函数上,FNPPSO 算法的性能都优于其他粒子群算法. 例如相比于 CCPSO 和 NPPSO 算法,FNPPSO 算法在分别在 15 个和 21 个函数上搜索性能胜出,并分别在其余的 9 个和 3 个函数上性能相当;相比于 Friedman 检验中排名第二的算法为 CSPSO,在所有的 24 个准则函数中,FNPPSO 有 9 个函数的性能优于 CSPSO,只有 1 个函数的性能劣于 CSPSO.

另外,FNPPSO 针对旋转和平移等复杂函数(F13-F24)的搜索性能要好于其他算法. 例如只有如下四种情况下,其他算法性能优于 FNPPSO,即:HEPSO 和 SLPSO 在函数 F20 上,SLPSO 在函数 F23 上,ASDPSO 在函数 F24 上. 在其余的所有情况下,FNPPSO 均优于或者与其他粒子群算法相当.

4.3 收敛速度比较

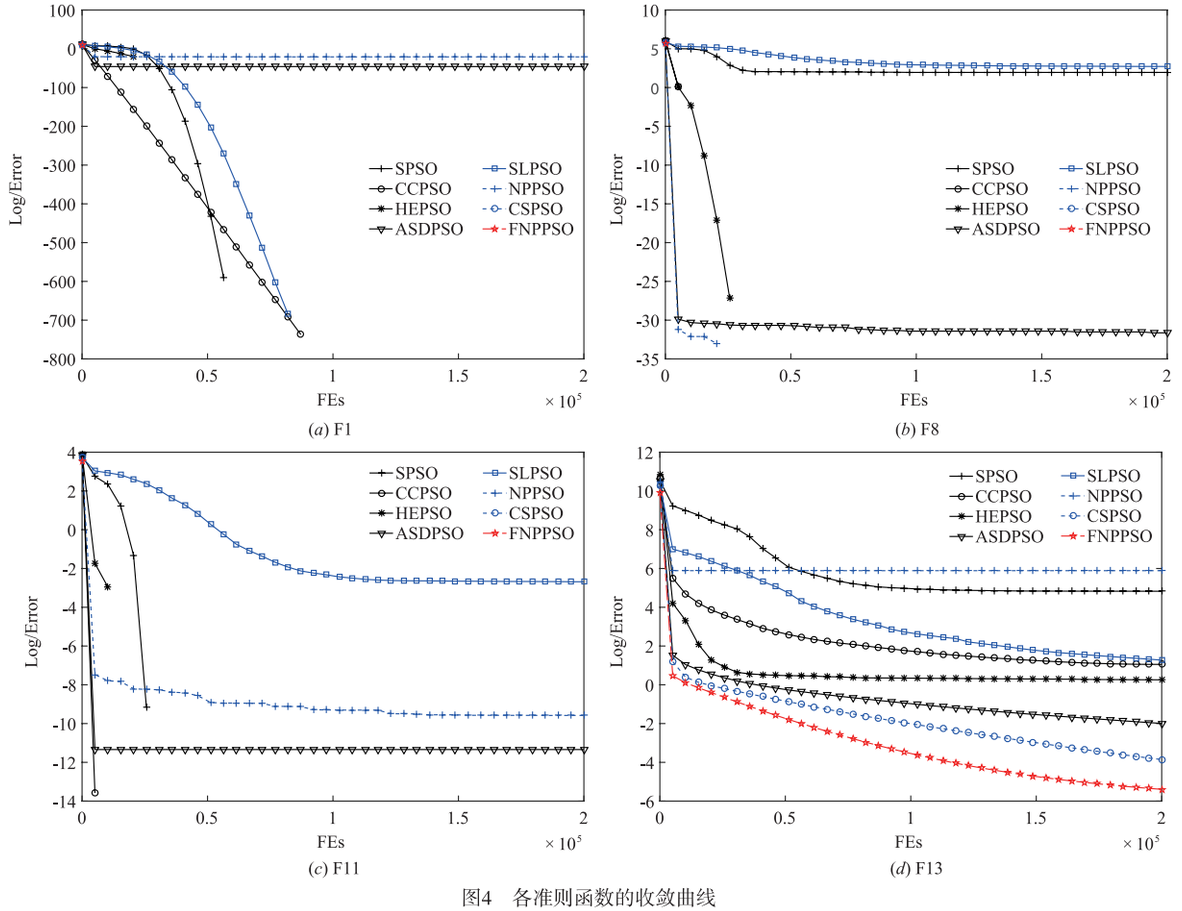
为了比较不同算法的收敛速度,我们选择了不同类型的函数进行了比较,包含单峰函数 F1,多峰函数 F8 和 F11,复杂函数 F13. 实验结果见图 4. 图中横坐标为迭代次数,纵坐标为搜索结果与全局最优结果的差值的对数. 当算法可以搜索到全局最优解时,两者差值的对数为负无穷大. 本文在负无穷时截断了显示的曲线.

由图中结果可知,相比于其他算法,FNPPSO 算法在所有函数上的收敛速度是最快的. 比如针对函数 F1,FNPPSO 算法在第 676 次就搜索到了全局最优解,而 CSPSO 算法则需要消耗 3515 次迭代. 由图 4 还可以看

出,FNPPSO 算法总是能在搜索开始不久,就能快速收敛. 比如针对 F1,F8,F11 等函数,FNPPSO 算法搜索到全局最优解平均分别需要 676 步,56 步,61 步. 针对复杂准则函数,在达到同样的收敛精度前提下,FNPPSO 所用迭代步数要少于其他算法. 比如函数 F13,当各个算法搜索到同样的精度 1.0E +000 时,ASDPSO 和 CSPSO 算法分别消耗 37451 步和 18605 步;FNPPSO 算法则只需要消耗 11982 步.

4.4　时间复杂度分析

本节给出 FNPPSO 算法的时间复杂度分析. 根据搜索过程,完成一次迭代,需要依次完成多交叉操作和中间变量 p' 的计算,共需要 4 次乘法和 7 次加法. 则 FNPPSO 算法完成搜索,总需要乘法次数为:$4 \times D \times N \times T$,加法次数为:$7 \times D \times N \times T$. 经典粒子群完成整个搜索过程,需要乘法次数为:$5 \times D \times N \times T$,加法次数为:$5 \times D \times N \times T$. FNPPSO 算法和经典粒子群算法的乘法和加法的时间运算复杂度均为 $O(D \times N \times T)$. 在一次迭代过程中,FNPPSO 需要进行 2 次目标函数的求值运算,经典粒子群算法只需要 1 次目标函数的求值运算. 因此,完成整个搜索过程,FNPPSO 和经典粒子群算法需要进行的目标函数求值运算次数分别为 $2 \times D \times N \times T$ 和 $D \times N \times T$ 次.



(*a*) F1　(*b*) F8　(*c*) F11　(*d*) F13

图4　各准则函数的收敛曲线

5　结论

很多研究人员提出了粒子群算法中的各种参数控制策略,这些策略都面临着不同的挑战. 本文提出了一种替代的解决方案,即非参数的粒子群算法. 该算法融合了多交叉操作和基于 exemplar 的学习策略,充分利用了前者的快速收敛性和后者的全局搜索能力,从而在全局搜索和局部搜索之间取得平衡. 根据进一步的稳定性分析,本文算法同时满足一阶和二阶稳定. 本文挑选了 24 个包含了多种类型的准则函数进行了实验. 最后的结果证明,本文算法在大部分的准则函数上的搜索精度都优于同类算法. 实验结果还表明,本文算法的收敛速度远超过其他同类粒子群优化算法.

参考文献

[1] Liang J J,Qin A K,Suganthan P N,Baskar S. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions[J]. IEEE Transactions on Evolutionary Computation,2006,10(3):281 – 295.

[2] Li Y H,Zhan Z H,Lin S J,Zhang J,Luo X N. Competitive

and cooperative particle swarm optimization with information sharing mechanism for global optimization problems [J]. Information Sciences, 2015, 293(3): 370 – 382.

[3] 彭宇,彭喜元,刘兆庆. 微粒群算法参数效能的统计分析[J]. 电子学报,2004,32(2):209 – 213.
PENG Yu, PENG Xi-yuan, LIU Zhao-qing. Statistic analysis on parameter efficiency of particle swarm optimization [J]. Acta Electronica Sinica, 2004, 32(2): 209 – 213. (in Chinese)

[4] Ren Z G, Zhang A M, Wen C Y, Feng Z R. A scatter learning particle swarm optimization algorithm for multimodal problems[J]. IEEE Transactions on Cybernetics, 2014, 44(7): 1127 – 1140.

[5] Shi Y H, Eberhart R C. A modified particle swarm optimizer[A]. Proceedings of IEEE International Conference on Computation Intelligence [C]. Anchorage: IEEE Press, 1998. 69 – 73.

[6] Eberhart R C, Shi Y. Comparing inertia weights and constriction factors in particle swarm optimization[A]. Proceedings of IEEE Congress on Evolutionary Computation [C]. California, USA: IEEE Press. 84 – 88.

[7] Khan S U, Yang S Y, Wang L Y, Liu L. A modified particle swarm optimization algorithm for gobal optimizations of inverse problems [J]. IEEE Transactions on Magnetics, 2016, 52(3): 1 – 4.

[8] Ardizzon G, Cavazzini G, Pavesi G. Adaptive acceleration coefficients for a new search diversification strategy in particle swarm optimization algorithms [J]. Information Sciences, 2015, 299: 337 – 378.

[9] Ratnaweera A, Halgamuge S K, Watson H C. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients [J]. IEEE Transactions on Evolutionary Computation, 2004, 8(4): 240 – 255.

[10] Beheshti Z, Shamsuddin S M. Non-parametric particle swarm optimization for global optimization[J]. Applied Soft Computing, 2015, 28: 345 – 359.

[11] Mahmoodabadi M J, Mottaghi Z S, Bagheri B. HEPSO: High exploration particle swarm optimization[J]. Information Sciences, 2014, 273: 101 – 111.

[12] Zhang C, Yi Z. Scale-free fully informed particle swarm optimization algorithm [J]. Information Sciences, 2011, 181(20): 4550 – 4568.

[13] Bonyadi M R, Michalewicz Z. Particle swarm optimization for single objective continuous space problems: a review [J]. Evolutionary Computation, 2017, 25(1): 1 – 54.

[14] Lim W H, Isa N A M. Teaching and peer-learning particle swarm optimization[J]. Applied Soft Computing, 2014, 18(18): 39 – 58.

[15] Bergh F V D, Engelbrecht A P. A study of particle swarm optimization particle trajectories [J]. Information Sciences, 2006, 176(8): 937 – 971.

[16] Bonyadi M R, Michalewicz Z. Stability analysis of the particle swarm optimization without stagnation assumption[J]. IEEE Transactions on Evolutionary Computation, 2016, 20(5): 814 – 819.

[17] Fang W, Sun J, Chen H, Wu X. A decentralized quantum-inspired particle swarm optimization algorithm with cellular structured population[J]. Information Sciences, 2016, 330: 19 – 48.

[18] Meng A, Li Z, Yin H, Chen S, Guo Z. Accelerating particle swarm optimization using crisscross search[J]. Information Sciences, 2016, 329: 52 – 72.

[19] Derrac J, García S, Molina D, Herrera F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms[J]. Swarm and Evolutionary Computation, 2011, 1(1): 3 – 18.

[20] Lim W H, Isa N A M. An adaptive two-layer particle swarm optimization with elitist learning strategy. Information Sciences[J]. 2014, 273(3): 49 – 72.

作者简介

刘兆广 男,1977 年 7 月出生,山东菏泽人. 山东财经大学计算机科学与技术学院副教授、硕士生导师,2008 年毕业于山东大学信息科学与工程学院,获工学博士学位,研究方向包括智能优化算法、数字图像处理、智能视频分析.
E-mail: liuzhg@ sdufe. edu. cn

纪秀花 女,1964 年 10 月生于湖北孝感,1985 年、1988 年在山东大学电子系获理学学士、理学硕士学位,2009 年在山东大学计算机学院获工学博士学位,山东财经大学教授,主要从事计算机图像处理等方面的研究工作.
E-mail: jane@ sdu. edu. cn