文章编号: 1000−8152(2010)10−1404−07

一种混合搜索的粒子群算法

连志刚[1,2], 焦 斌[1]

(1. 上海电机学院 电子信息学院, 上海 200240; 2. 上海交通大学 机械与动力工程学院, 上海 200240)

摘要: 本文通过对粒子群算法个体极值、全局极值和种群极值的结合, 提出一种混合搜索粒子群算法. 用典型的非线性测试函数进行仿真, 其实验数据和收敛曲线验证了该算法的有效性, 具有快速收敛效果和寻优能力.

关键词: 粒子群优化算法; 混合搜索; 优化

中图分类号: TP301 **文献标识码:** A

Particle-swarm optimization algorithm with mixed search

LIAN Zhi-gang[1,2], JIAO Bin[1]

(1. School of Electronic and Information Engineering, Shanghai Dianji University, Shanghai 200240, China;
2. School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China)

Abstract: A mixed search particle-swarm optimization algorithm(MSPSO) is proposed by combining the algorithms for individual optimization, global optimization and generation-population optimization. In the simulations by using benchmark non-linear test functions, experiment data and convergence curves show that this new algorithm is effective, rapidly convergent in optima search.

Key words: PSO algorithm; mixed search; optimization

1 引言(Introduction)

自然界中一些生物的行为呈现群体特征, 一定情况下可以在计算机中用简单的规则来建立个体运动模型. 生物学家Frank Heppner建立了鸟群运动模型, 与其他模型不同的是该鸟群模型中鸟会受到食物地的吸引. Kennedy和Eberhart将食物地类比为所求问题解空间中可能解的位置, 通过个体间的信息传递, 导引整个群体向可能解的方向移动. 他们1995年提出了粒子群优化算法(PSO), 引起了学者和工程技术人员的研究兴趣. 如Shi和Eberhart在基本PSO算法中加上惯性因子和约束因子, 提出了带有惯性权重的PSO算法模型[1]. Clerc证明了采用收敛因子能确保PSO算法的收敛, 并提出了收敛因子模型PSO算法的速度递推方程[2]. Lovbjerg等将进化算法中的交叉操作引入HPSO算法模型[3]. 文献[4,5]分别提出了一种新型粒子群优化算法和动态惯性权重的PSO算法. 作者曾在博士论文中提出了局部与全局相结合的粒子群算法[6], 文献[7]也对其进行了验证分析, 实验表明该算法搜索效率较好. 除了以上改进的PSO算法还有协同PSO算法[8]、混沌PSO算法[9]、量子PSO算法[10]、智能PSO算法[11]等. 在前辈们研究成果的基础上, 本文提出一种全局与局部搜索相结合的混合型粒子群算法, 它在共享个体和全局粒子良好信息外, 还共享了种群中最优粒子的信息. 应用一些典型非线性测试函数进行试验, 其仿真数据和收敛曲线验证了该算法优化中、大规模问题的有效性, 具有快速收敛效果和寻优能力.

2 初始粒子群算法(Original PSO)

初始PSO算法在每一次迭代中, 粒子通过跟踪粒子本身所找到的个体最优$p_{\rm best}$和整个种群目前找到的全局最优$g_{\rm best}$两个“极值”来更新自己的位置. 假设在一个D维目标搜索空间中种群粒子的规模为m, 粒子 $\vec{X}_i = (x_{i1}, x_{i2}, \cdots, x_{iD})(i = 1, 2, \cdots, m)$是$D$维向量. $\vec{V}_i = (v_{i1}, v_{i2}, \cdots, v_{iD})$是粒子$i$飞行速度, $\vec{P}_i = (p_{i1}, p_{i2}, \cdots, p_{iD})$是粒子$i$个体历史最优位置$p_{\rm best}$ $\vec{P}_g = (p_{g1}, p_{g2}, \cdots, p_{gD})$是整个粒子群当前为止搜索到的全局最优位置 $g_{\rm best}$. 初始粒子群算法(original particle-swarm optimization algorithm, OPSO)的递推方程如下[12]:

$$v_{id}(k+1) = wv_{id}(k) + c_1 r_1(p_{id}(k) - x_{id}(k)) +$$

收稿日期: 2009−3−25; 收修改稿日期: 2009−8−17.

基金项目: 教育部人文社会科学研究青年基金资助项目(09yjc630151); 上海市科委基础研究重点资助项目(10JC1405800); 上海高校选拔培养优秀青年教师科研专项基金资助项目(sdju200903); 上海电机学院科研启动基金资助项目(09C403).

$$c_2r_2(p_{gd}(k)-x_{id}(k)), \tag{1}$$

$$\begin{cases} x_{id}(k+1)=x_{id}(k)+v_{id}(k+1), \\ i=1,2,\cdots,m,\ d=1,2,\cdots,D. \end{cases} \tag{2}$$

在式(1)和(2)中, k为迭代代数; 学习因子c_1和c_2为非负常数, 一般取值为2; w是惯性系数, 取值一般小于1大于0; r_1和r_2是均匀分布于[0,1]之间的两个随机数.

OPSO算法主要通过3部分来更新粒子i的新速度: 即粒子i前一次迭代时刻的速度, 粒子i当前位置与自己最好位置之间的方向, 粒子i当前位置与群体最好位置之间的方向. 粒子i通过公式(1)更新位置坐标, 通过递推方程决定下一步的运动位置. 在图1中, 以两维空间为例描述粒子根据递推方程从位置$\vec{X}_i(k)$到$\vec{X}_i(k+1)$的移动原理.

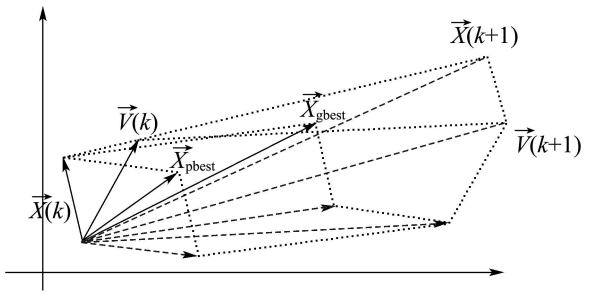


图 1　初始粒子群算法的粒子移动原理

Fig. 1　The particle move theory of OPSO

3　混合搜索粒子群算法(Mixed search PSO)

3.1　MSPSO算法递推方程(Recurrence equation of MSPSO)

采用全局和局部不同极值更新粒子的迭代方程, 称为全局版和局部版的PSO算法. 它们各有利弊, 前者收敛速度快但有时会陷入局部最优, 后者收敛速度慢一点, 但很容易跳出局部最优. 为了克服OPSO算法的缺点, 本文提出全局和局部相结合的混合搜索粒子群算法(mixed search particle-swarm optimization algorithm, MSPSO), 其继承了全局和局部PSO算法的各自优点.

假设在一个D维目标搜索空间中, m个粒子组成一个种群, 粒子

$$\vec{X}_i=(x_{i1},x_{i2},\cdots,x_{iD}),\ i=1,2,\cdots,m$$

表示D维向量.

$$\vec{V}_i=(v_{i1},v_{i2},\cdots,v_{iD})$$

为粒子i的飞行速度,

$$\vec{P}_i=(p_{i1},p_{i2},\cdots,p_{iD})$$

是粒子i迄今为止搜索到的个体历史最优$p_{\rm best}$, $\vec{P}_l=(p_{l1},p_{l2},\cdots,p_{lD})$是每代种群搜索到的最优位置$l_{\rm best}$, $\vec{P}_g=(p_{g1},p_{g2},\cdots,p_{gD})$是整个粒子群迄今为止搜索到的最优位置$g_{\rm best}$. MSPSO算法的递推方程如下:

$$\begin{aligned} &v_{id}(k+1)= \\ &wv_{id}(k)+c_1r_1[(1-\alpha^k)(p_{id}(k)- \\ &x_{id}(k))+\alpha^k(p_{ld}(k)-x_{id}(k))]+ \\ &c_2r_2(p_{gd}(k)-x_{id}(k)), \end{aligned} \tag{3}$$

$$\begin{cases} x_{id}(k+1)=x_{id}(k)+v_{id}(k+1), \\ i=1,2,\cdots,m,\ d=1,2,\cdots,D. \end{cases} \tag{4}$$

在式(3)和(4)中, 迭代代数k惯性系数w, r_1和r_2及学习因子c_1和c_2与OPSO相同; α为在[0,1]之间的一个常数; $v_{id}\in[-v_{\max},v_{\max}]$ $v_{\max}$是用户自己设定的常数; $v_{id}(k)$为第k代迭代粒子i飞行速度矢量的第d维分量; $x_{id}(k)$是第k代迭代粒子i位置矢量的第d维分量; $p_{id}(k)$是粒子i个体最好位置的第d维分量; $p_{ld}(k)$是第k代种群中粒子最好位置的第d维分量; $p_{gd}(k)$为全体粒子最好位置的第d维分量; 迭代终止条件根据具体问题一般选为最大迭代代数或(和)粒子群迄今为止搜索到的最优位置满足预定最小适应阈值.

MSPSO算法主要通过3部分来更新粒子i的新速度: 粒子i前一次迭代速度; 粒子i当前位置与自己最好位置之间的方向; 第k代种群中粒子最好位置与自己位置之间的方向, 再结合粒子i当前位置与群体最好位置之间的方向. 该算法将个体最优粒子和种群最优粒子采用了动态结合方式, 开始主要共享种群最优粒子信息, 随着代数的增加, 更多的共享个体最优粒子的信息. 充分共享全局最优$p_{gd}(k)$、个体最优$p_{id}(k)$和种群最优即局部最优$p_{ld}(k)$的信息, 故具有全局和局部较强的寻优能力. 在图2中, 以两维空间为例描述MSPSO算法的粒子从位置$\vec{X}_i(k)$到$\vec{X}_i(k+1)$的移动原理.

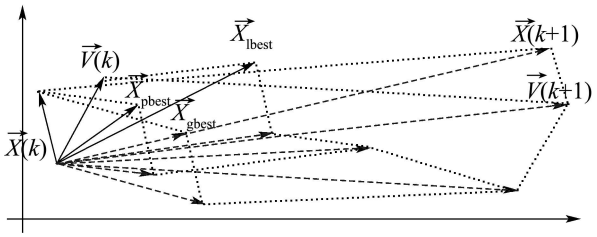


图 2　混合搜索粒子群算法粒子移动原理

Fig. 2　The particle move theory of MSPSO

3.2 MSPSO算法流程图(Flow chart of MSPSO)

MSPSO 算法主要通过 $\vec{x}_{p\mathrm{best}}(k), \vec{x}_{l\mathrm{best}}(k)$ 和 $\vec{x}_{g\mathrm{best}}(k)$ 3部分来传递和共享信息, 更新粒子i的新速度和位置, 其流程如图3所示.

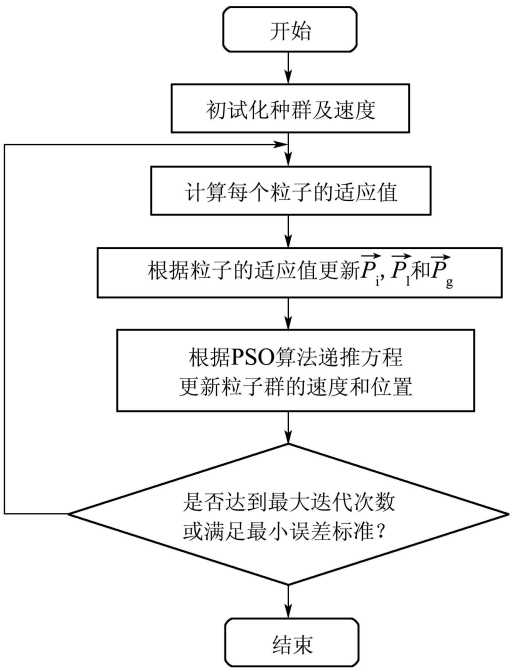


图 3 MSPSO算法流程

Fig. 3 Flow chart of MSPSO

4 仿真实验(Simulation experiment)

4.1 测试函数(Test functions)

为了验证MSPSO算法优化连续函数的效率, 将它和OPSO算法优化经典测试函数在不同维数情况下进行比较, 观察它们优化连续函数的效率差别. 用来实验的经典非线性测试函数有:

1) Sphere model: f_1;

2) Schwefel's Problem 1.2: f_2;

3) Schwefel's Problem 2.21: f_3;

4) Generalized Rosenbrock's function: f_4;

5) Ackley's function: f_5;

6) Generalized Griewank function: f_6;

7) Generalized Penalized Functions: f_1, f_8, 具体相关模型参数等见文献[5].

4.2 试验比较(Experiment comparison)

在MSPSO算法效率测试实验过程中, 为了实验结果科学, 更具说服力, 将它与OPSO算法优化8个经典连续函数的结果进行比较. 对每个例子各运行10次, 对10次运行结果中各自的最优值Min、最差值Max和平均值Average进行比较分析, 如表1所示.

注 1 在表1中, Fun指测试函数, Dim为该函数要测试的维数, Best指该函数的最优值, PS和EG分别为算法的种群规模和停止迭代的代数; MSPSO和OPSO算法在不同的惯性系数下, 运行10次所求到的最优平均值用斜体表示, 最优极值用黑体表示; e~$\pm n$表示$\times 10^{\pm n}$.

表 1 MSPSO与OPSO算法优化测试函数比较

Table 1 The comparison results of MSPSO and OPSO algorithm for test functions

Min/Max/Average			
w	OPSO	a	MSPSO
0.1	1.4516e+9/ 4.7293e−4/ 8.9251e−5	0.99	1.2528e−13/3.0954e−4/3.1294e−5
		0.995	1.539e−20/8.6202e−8/8.7686e−9
		0.999	3.8904e−21/5.4135e−18/9.5162e−19
0.2	8.933e−6/ 2.4332/ 0.2475	0.99	2.9609e−20/0.0011/1.1838e−4
		0.995	6.592e−14/2.9706e−7/3.0146e−8
		0.999	3.4308e−25/9.0369e−23/1.7939e−23
0.3	1.0912e−35/ 1.1266e−32/ 5.2723e−33	**0.99**	**1.2931e−36**/8.7945e−8/9.7903e−9
		0.995	5.637e−35/2.7626e−20/2.8057e−21
		0.999	7.5105e−26/1.3705e−23/4.9872e−24
0.4	6.7803e−25/ 5.1332e−22/ 9.2853e−23	0.99	4.6988e−26/3.6438e−23/5.3027e−24
		0.995	8.9845e−27/4.0722e−24/*8.9337e−25*
		0.999	3.0364e−23/9.7341e−21/2.6667e−21
0.5	5.4532e−15/ 2.5376e−13/ 4.8343e−14	0.99	3.1704e−16/2.5845e−14/8.1655e−15
		0.995	2.8225e−18/1.2911e−15/3.8379e−16
		0.999	1.0594e−16/8.4929e−14/1.4377e−14
Fun= f_1, Dim=30, Best=0, PS=100, EG=1000			
w	OPSO	a	MSPSO
0.1	20.499/ 108.9765/ 53.1929	0.99	22.8468/202.6571/77.2275
		0.995	8.106/76.5457/27.6106
		0.999	1.4009e−4/0.0177/0.0073
0.2	0.1395/ 3.8792/ 1.3891	0.99	0.0263/1.4385/0.5168
		0.995	0.0142/0.8972/0.1533
		0.999	5.9124e−8/2.261e−6/7.7363e−7
0.3	0.0515/ 0.3006/ 0.1588	0.99	0.0063/0.1807/0.0802
		0.995	5.3284e−4/0.0429/0.0122
		0.999	**8.7859e−10**/3.6843e−7/*1.0195e−7*
0.4	0.344/ 3.1586/ 0.9316	0.99	0.019/0.3608/0.1579
		0.995	0.0043/0.157/0.0489
		0.999	1.9729e−8/1.4355e−6/4.538e−7
0.5	2.5381/ 23.079/ 10.4773	0.99	1.409/32.9093/7.1717
		0.995	0.1031/4.5568/1.8675
		0.999	8.7562e−5/0.0045/0.0014
Fun=f_2, Dim=30, Best=0, PS=150, EG=2000			

(转下页)

(接上页)

w	OPSO	a	MSPSO
0.1	1.3206/ 7.328/ 3.6149	0.99	0.5084/1.8239/1.1503
		0.995	0.3215/1.8537/0.7121
		0.999	3.9648e−4/0.0055/0.0022
0.2	0.05/ 0.2297/ 0.1118	0.99	0.0034/0.1028/0.027
		0.995	0.0019/0.0278/0.0129
		0.999	2.0251e−7/2.4716e−5/4.8998e−6
0.3	2.4735e−4/ 0.053/ 0.012	0.99	2.5251e−4/0.0077/0.0031
		0.995	9.138e−5/0.0018/0.0011
		0.999	**4.6562e−9**/1.9145e−7/3.9337e−8
0.4	0.0011/ 0.0213/ 0.0086	0.99	3.2694e−004/0.0116/0.0027
		0.995	3.6434e−5/0.0013/4.1507e−4
		0.999	8.8118e−9/4.9825e−8/*2.2776e−8*
0.5	0.0485/ 0.6536/ 0.2644	0.99	0.0113/0.4077/0.0662
		0.995	0.0035/0.1034/0.0278
		0.999	4.1982e−7/1.6134e−5/4.5272e−6

Fun=f_3, Dim=20, Best=0, PS=150, EG=2500

w	OPSO	a	MSPSO
0.1	3.4031/ 1.1515e+3/ 141.7273	0.99	0.0034/452.9054/137.2951
		0.995	0.5257/3.2293e+3/507.6061
		0.999	0.321/481.4141/120.124
0.2	2.5243/ 249.0349/ 33.338	0.99	0.0502/297.7689/36.8927
		0.995	0.0024/189.5679/22.0506
		0.999	1.2301e−4/3.0825e+3/313.3669
0.3	4.5776/ 85.5203/ 17.6426	0.99	0.3937/629.8227/122.4108
		0.995	0.0773/465.6484/54.5788
		0.999	**5.739e−5**/272.3007/30.6536
0.4	0.0209/ 481.1329/ 56.1045	0.99	1.8046/88.6499/17.264
		0.995	0.0058/55.1685/13.0651
		0.999	0.0054/19.5187/*5.6797*
0.5	4.6799/ 203.955/ 28.2054	0.99	0.7523/245.655/37.9698
		0.995	0.0191/945.7564/119.2539
		0.999	0.0681/18.268/6.6254

Fun=f_4, Dim=20, Best=0, PS=150, EG=2500

w	OPSO	a	MSPSO
0.1	17.812/ 20.1651/ 19.179	0.99	1.0557/20.9481/18.7725
		0.995	0.7624/20.8443/14.8518
		0.999	0.9066/20.8983/16.92
0.2	18.9906/ 20.0748/ 19.2255	0.99	1.772/19.5426/12.6075
		0.995	2.3525/19.1579/15.8684
		0.999	1.9499/20.952/17.6407
0.3	5.3919/ 19.4215/ 15.6091	**0.99**	**2.2097e−5**/19.3164/4.2292
		0.995	0.0022/19.157/6.1123
		0.999	1.8767/19.205/7.0285
0.4	2.5561e-4/ 19.2963/ 13.4944	0.99	7.7184e−5/7.4804/0.7634
		0.995	5.8996e−5/0.085/*13.0651*
		0.999	0.811/18.8865/3.6245
0.5	16.6545/ 19.4214/ 18.4312	0.99	0.0273/18.837/12.642
		0.995	0.0461/19.1409/12.4507
		0.999	1.7452/20.8859/12.6183

Fun=f_5, Dim=100, Best=0, PS=150, EG=2500

w	OPSO	a	MSPSO
0.1	5.1566e+3/ 5.5765e+3/ 5.3574e+3	0.99	0.5766/4.6271e+3/464.3464
		0.995	0.0514/0.6502/0.264
		0.999	1.0788/47.9959/9.8066
0.2	4.6073e+3/ 5.5016e+3/ 5.1537e+3	**0.99**	**0.0477**/92.2861/28.8496
		0.995	3.5112/394.7064/241.9447
		0.999	9.0387/136.0747/63.2605
0.3	5.0957e+003/ 5.4118e+003/ 5.2722e+3	0.99	0.259/137.3871/15.4753
		0.995	0.8680/302.4841/158.9036
		0.999	5.784/103.7519/52.2795
0.4	4.8707e+3/ 5.5464e+3/ 5.28e+3	*0.99*	0.0972/0.4535/*0.2067*
		0.995	0.3204/25.4048/3.0797
		0.999	0.7405/27.1906/6.2224
0.5	5.0722e+3/ 5.4679e+3/ 5.3236e+3	0.99	3.4191/3.1955e+3/334.5516
		0.995	1.711/2.8684/2.223
		0.999	1.1121/5.1068/2.5369

Fun=f_6, Dim=200, Best=0, PS=150, EG=2500

w	OPSO	a	MSPSO
0.1	1.7774e+9/ 2.7663e+9/ 2.3549e+9	0.99	18.1293/1.2967e+9/1.8199e+8
		0.995	6.5832/358.3108/53.6927
		0.999	0.5039/3.2196/1.8706
0.2	1.9814e+9/ 2.7999e+9/ 2.3182e+9	0.99	4.9507/6.2884e+8/6.2884e+7
		0.995	1.8364/16.3723/7.4575
		0.999	0.187/2.3223/1.3681
0.3	1.4125e+9/ 2.8113e+9/ 2.3795e+9	0.99	0.8148/5.5739/3.1983
		0.995	0.2308/6.1238/3.03
		0.999	0.0015/1.9817/0.536
0.4	1.835e+9/ 2.633e+9/ 2.4368e+9	0.99	1.167/7.9968/3.2991
		0.995	3.1727e−4/2.8202/1.3172
		0.999	**6.6459e−10**/0.2497/**0.0717**
0.5	1.9809e+9/ 2.5539e+9/ 2.2965e+9	0.99	18.3986/2.1726e+9/1.1857e+9
		0.995	2.2136/1.2806e+9/2.0675e+8
		0.999	2.6637e−5/0.5308/0.1605

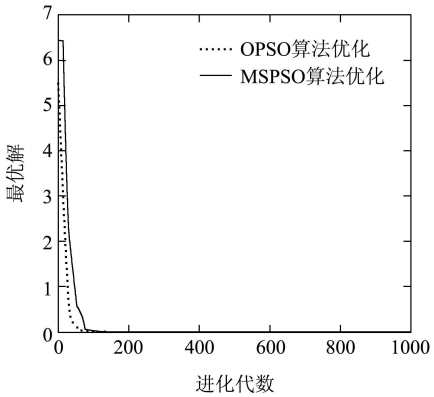
Fun=f_7, Dim=100, Best=0, PS=150, EG=2500

(转下页)

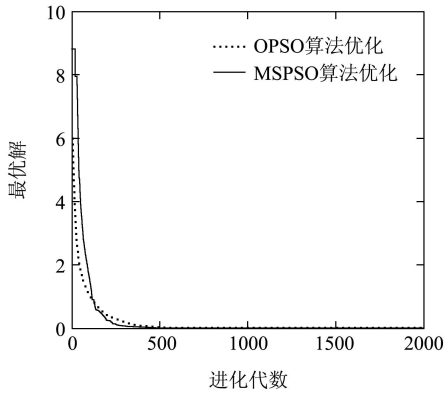
(接上页)

w	OPSO	a	MSPSO
0.1	2.0169e+9/ 2.9576e+9/ 2.5022e+9	0.99	80.3299/1.6411e+9/4.6903e+8
		0.995	23.51/2.2528e+9/2.2528e+8
		0.999	1.0054/7.5395/2.9729
0.2	2.1738e+9/ 2.8269e+9/ 2.5121e+9	0.99	11.2527/9.0641e+8/9.0641e+7
		0.995	11.3236/38.8072/18.7422
		0.999	1.6533/8.819/4.8169
0.3	2.1738e+9/ 2.6814e+9/ 2.4292e+9	0.99	1.8924/27.1742/8.4305
		0.995	**6.8786e-4**/15.7887/6.8938
		0.999	0.0971/4.5465/1.9147
0.4	1.9752e+9/ 2.787e+9/ 2.2897e+9	0.99	0.7673/11.9368/7.6444
		0.995	0.0011/11.0721/3.2545
		0.999	2.3104/10.4719/5.9088
0.5	1.8155e+9/ 2.4058e+9/ 2.1745e+9	0.99	23.9771/2.4618e+9/1.3709e+9
		0.995	18.4184/4.6873e+8/4.6903e+7
		0.999	**1.0485e-4**/0.4217/*0.0887*
Fun=f_8, Dim=100, Best=0, PS=150, EG=2500			

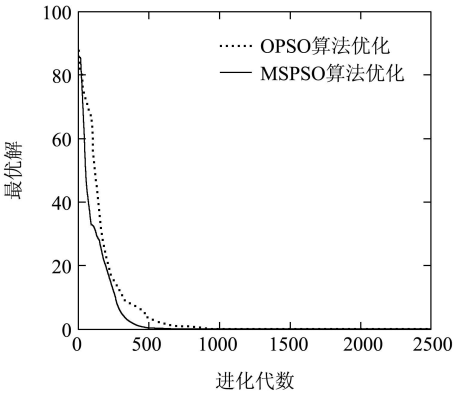
从表1的优化结果比较来看, MSPSO算法优化得到的10次最小极值Min、最差值Max和平均值都小于OPSO所搜索到的, 特别是MSPSO算法优化中、大规模问题相对更为有效. 另外从表1可以发现, 惯性系数$w \in [0.15, 0.45]$时MSPSO算法有更高的效率, 此时它优化连续函数获得的极值都比采用其他惯性系数更优. 通过单峰、多峰经典函数在不同维数情况下的仿真, 发现惯性系数$w \approx 0.3$时, MSPSO算法效率最高. MSPSO算法优化某些低维连续函数效果不明显, 甚至没有OPSO效率高, 但对相同的问题, 当为中、大规模时它的优势就明显的表现出来了. 对于不同的优化问题, 应选取不同的参数α, 其优化效率会更好.



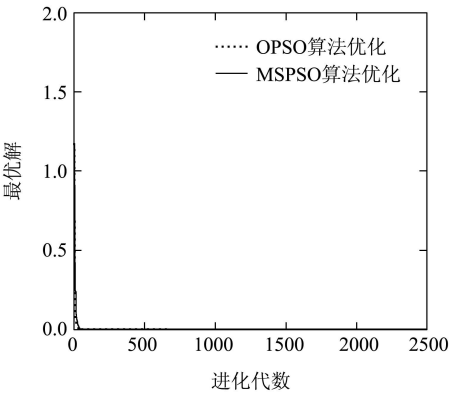
(a) PSO和OPSO算法优化函数f_1收敛对比图



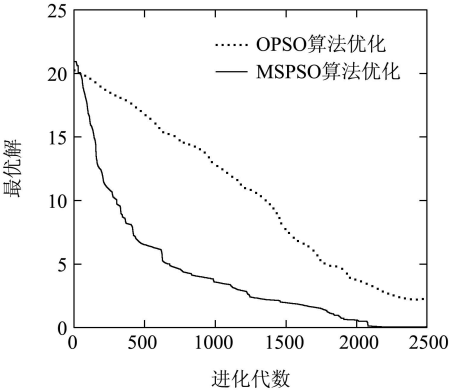
(b) MPSO和OPSO算法优化函数f_2收敛对比图



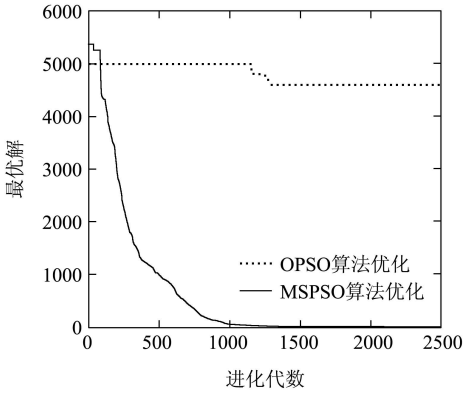
(c) MPSO和OPSO算法优化函数f_3收敛对比图



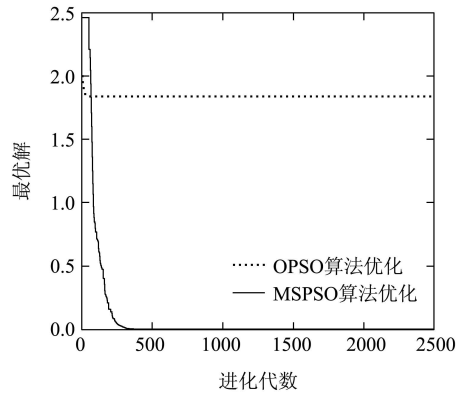
(d) MPSO和OPSO算法优化函数f_4收敛对比图



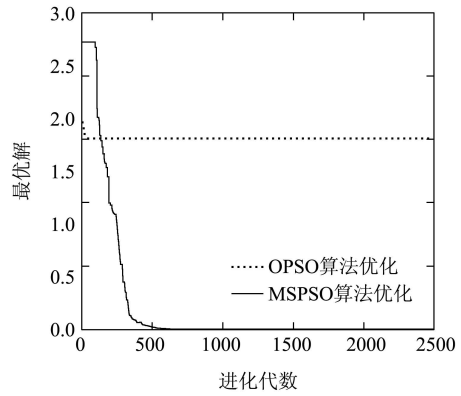
(e) MPSO和OPSO算法优化函数f_5收敛对比图



(f) MPSO和OPSO算法优化函数f_6收敛对比图



(g) MPSO和OPSO算法优化函数f_7收敛对比图



(h) MPSO和OPSO算法优化函数f_8收敛对比图

图 4　MSPSO与OPSO算法优化测试函数的收敛比较

Fig. 4　The convergence comparison of MSPSO and OPSO algorithm for test functions

文中给出MSPSO和OPSO算法使用不同惯性系数优化8个典型非线性连续函数时, 搜索最优解过程的收敛曲线图(见图4所示).

从图4来看, MSPSO比OPSO算法优化连续函数的收敛速度明显更快. OPSO算法优化中、大维数的连续函数时容易陷入局部最优, 如优化维数为150的函数f_1,f_5, 维数为200的函数f_6, 维数为100的函数f_7, f_8, 很早地陷入了局部最优值, 但采用MSPSO算法优化同样维数的相同问题时, 更容易趋近全局最优值. 从实验结果来分析, 优化高维连续函数时MSPSO比OPSO算法明显更为有效.

5　结论(Solution)

粒子群优化算法是一种优良的优化工具, 由于其容易理解、易于实现, 其发展速度很快, 在众多领域得到了深入广泛的应用. 本文实验结果表明初始PSO算法对优化小规模问题具有快速收敛的能力, 但对中、大规模的问题容易陷入局部最优, 几乎很难收敛到全局最优. 基于初始PSO算法的原理, 本文提出全局和局部相结合的混合搜索粒子群算法, 这种新模型在一定程度上继承了全局和局部PSO算法的各自优点, 通过实验测试, 发现该算法优化中、大规模问题效果明显比初始PSO算法优良.

本文对参数α在小范围内进行了实验, 发现其优化不同问题时取值不同算法的效率较好, 将来可以继续研究它对算法性能的影响. 今后一方面的工作可以探讨该算法的收敛性能, 及优化更大规模问题的效率. 另一方面将MSPSO算法进行移植, 用于优化其他类型的优化问题, 如TSP、分配问题、最短路问题、调度问题等.

参考文献(References):

[1] SHI Y, EBERHART R C. Modified particle-swarm optimizer[C] //*IEEE International Conference of Evolutionary Computation*. Anchorage, Alaska, [s.n.], 1998: 69 – 73.

[2] CLERC M. The swarm and queen towards a deterministic and adaptive particle-swarm optimization[C] //*Proceedings of Congress on Evolutionary Computation*. Washinton DC, USA: [s.n.], 1999: 782 – 786.

[3] LOVBJERG M, RASMUSSEN TK, KRINK T. Hybrid particle-swarm optimization with breeding and subpopulations[C] //*IEEE International Conference on Evolutionary Computation*. San, Diego: IEEE, 2000: 1217 – 1222.

[4] 巩敦卫, 张勇, 张建化, 等.一种新型粒子群优化算法[J]. 控制理论与应用, 2008, 25(1): 111 – 114, 119.
(GONG Dunwei, ZHANG Yong, ZHANG Jianhua, et al. Novel particle-swarm optimization algorithm[J]. *Control Theory & Applications*, 2008, 25(1): 111 – 114, 119.)

[5] JIAO B, LIAN Z G, GU X S. A dynamic inertia weight particle-swarm optimization algorithm[J]. *Chaos, Solitons and Fractals*, 2008, 37: 698 – 705.

[6] 连志刚. 粒子群优化算法及其在生产调度中的应用[D]. 上海: 华东理工大学, 2006, 12.

[7] 胡乃平, 宋世芳. 一种局部与全局相结合的微粒群优化算法[J]. 计算机工程, 2008, 34(17): 205 – 207.

[8] 周苗, 陈义保, 刘加光. 一种新的协同多目标粒子群算法[J]. 山东理工大学学报(自然科学版), 2008, 22(5): 6 – 10.
(ZHOU Miao, CHEN Yibao, LIU Jiaguang. A new kind of cooperative multi2objective particle-swarm optimization algorithm[J]. *Journal of Shandong University of Technology (Natural Science Edition)*, 2008, 22(5): 6 – 10.)

[9] 高鹰, 谢胜利. 混沌粒子群优化算法[J]. 计算机科学, 2004, 31(8): 13 – 15.
(GAO Ying, XIE Shengli. Chaos particle-swarm optimization algorithm[J]. *Computer Science*, 2004, 31(8): 13 – 15.)

[10] 方伟, 孙俊, 须文波. 基于微分进化算子的量子粒子群优化算法及应用[J]. 系统仿真学报, 2008, 20(24): 6740 – 6744.
(FANG Wei, SUN Jun, XU Wenbo. Improved Quantum-behaved particle swarm optimization algorithm based on differential evolution operator and Iis application[J]. *Journal of System Simulation*, 2008, 20(24): 6740 – 6744.)

[11] CIUPRINA G, IOAN D, MUNTEANU I. Use of intelligent-particle-swarm optimization in electromagnetics[J]. *IEEE Transactions on Magnetics*, 2002, 38(2): 1037 – 1040.

[12] EBERHART R, KENNEDY J. A new optimizer using particle-swarm theory[C] //*Proceedings of the 6th International Symposium on Micro Machine and Human Science*. Nagoya, Japan: [s.n.], 1995: 39 – 43.

作者简介:

连志刚 (1975—), 男, 博士, 研究方向为优化算法、MES、生产计划与调度、港口调度、物流工业工程, E-mail: lllzg@163.com;

焦 斌 (1958—), 男, 博士, 教授, 目前研究方向为自动控制、智能优化算法、生产计划与调度, E-mail:binjiaocn@163.com.

下 期 要 目