文章编号: 1001-0920 (2012) 03-0343-06

融合 Powell 搜索法的粒子群优化算法

吴建辉[a,b], 章 兢[a], 陈红安[b]

(湖南大学 a. 电气与信息工程学院，b. 信息科学与工程学院，长沙 410082)

摘 要: 为了进一步提高多模态函数寻优的效率, 提出一种融合 Powell 搜索法的粒子群优化算法. 将 PSO 算法的全局搜索能力与 Powell 法的强局部寻优能力有机地结合起来, 在保证求解速度, 尽可能找到全部极值点的同时提高了解的精确性. 由于该算法只利用了函数值信息而不需要计算导数, 是求解可微和不可微多模态函数优化问题的通用方法. 仿真实验表明了新混合算法的有效性.

关键词: 多模态函数；Powell 搜索法；粒子群优化

中图分类号: TP18 **文献标识码:** A

Particle swarm optimization algorithm combination with Powell search method

WU Jian-hui[a,b], *ZHANG Jing*[a], *CHEN Hong-an*[b]

(a. College of Electrical and Information Engineering，b. College of Information Science and Engineering，Hu'nan University，Changsha 410082，China．Correspondent：WU Jian-hui，E-mail：wujianhui123@tom.com)

Abstract: In order to further improve the efficiency of multi-modal function optimization, a novel hybrid algorithm which combines improved particle swarm optimization algorithm and Powell search method is proposed. The Powell-IPSO hybrid algorithm organically integrates PSO algorithm which has powerful global search capability with Powell search method which has strong local search ability. The hybrid algorithm ensures quick convergent speed and find all extreme points as much as possible, and solution's precision is improved. The hybrid algorithm only uses function values information and doesn't need to calculate the derivative of function. So it is a generalized algorithm for multi-modal function of differentiable and non-differentiable. The simulation experionents show the effectiveness of the hybrid algorithm.

Key words: multi-modal function；Powell search method；particle swarm optimization

1 引 言

现实中很多实际优化问题 (如模糊系统结构和参数优化、神经元的结构及权重优化等) 常存在多个最优解或者一个最优解和多个局部最优解, 它们都可以归类为多模态函数优化问题 (MFO) 或多峰值函数优化问题. 如何构造一种快速、有效的优化算法, 使之能求出全部全局最优解和尽可能多的局部最优解, 已成为进化计算的一个重要研究方向. 目前已提出了多种多峰值寻优算法, 如: 协同多群体遗传算法[1]、多种群进化策略[2]、融合粒子群优化的免疫网络算法[3]、与经典优化算法结合的混沌微粒群算法[4]等. 这些算法取得了一定的效果, 但也存在着求解精度不高等问题.

粒子群优化算法 (PSO) 是一种基于种群搜索策略的随机优化算法, 由 Eberhart 等人[5]首次提出. 由于 PSO 算法具有概念简单、容易实现、不依赖于函数性态、适应范围广和鲁棒性强等优点, 在科学和工程领域得到了广泛应用[6], 但对于复杂优化问题, PSO 算法也存在早熟收敛、求解精度较差、对多模态函数搜索效果不佳等缺陷.

Powell 搜索法是由 Powell 提出的一种解无约束最优化问题的常规的直接局部搜索法[7]. 它具有计算简单、收敛速度快、不需要计算导数、精度高、可靠性好等优点, 但为了保证收敛到全局最优值, 必须认真选择初始点[8]. Powell 法具有较强的局部搜索能力, 可以找出每个个体 (粒子) 在目前环境下所对应的局部最优解.

为了尽可能寻找多模态函数的全部极值点及提高多峰寻优求解精度, 融合 PSO 和 Powell 搜索法各自

收稿日期: 2010-10-08；**修回日期:** 2010-12-18.

基金项目: 国家自然科学基金重点项目(60634020)；国家自然科学基金项目(60874096).

作者简介: 吴建辉(1970−), 男, 讲师, 博士生, 从事自然计算的研究；章兢(1957−), 男, 教授, 博士生导师, 从事智能控制、智能计算等研究.

的优点, 提出了 Powell-IPSO 算法. 将 PSO 算法的全局搜索能力与 Powell 法的强局部寻优能力有机地结合起来, 在保证求解速度的同时提高了解的精确性. 由于该算法只利用了函数值信息而不需要计算导数, 是求解可微和不可微多模态函数优化问题的通用方法. 仿真实验表明了新混合算法的有效性.

2 Powell-IPSO 算法

2.1 多模态函数优化问题及适应度函数

2.1.1 多模态函数优化问题

多模态函数优化问题为

$$\min(f(x)) = f(x_1, x_2, \cdots, x_n),$$
$$\text{s.t. } X = (x_1, x_2, \cdots, x_n) \in S. \tag{1}$$

其中: 满足所有约束条件的 S 称为可行域, 可行域中的解称为可行解, 在可行域中使目标函数最小的解为最优解. 对于 MFO 问题, 存在多个全局最优解和局部最优解.

2.1.2 适应度函数

对于求目标函数最小值的 MFO 问题, 适应度函数定义为

$$\text{fit}(X) = f(X). \tag{2}$$

其中: $f(X)$ 为 X 的函数值, $\text{fit}(X)$ 为其适应度值.

对于求函数最大值的 MFO 问题, 采用界限构造法转化为最小化问题求解, 适应度函数定义为

$$\text{fit}(X) = C_{\max} - f(X), \tag{3}$$

其中 $C_{\max}$ 为 $f(X)$ 的最大值估计. 由式 (2) 和 (3) 可知, X 的适应度值越小, 其适应度越优.

2.2 改进的微粒群算法

设规模为 N 的微粒群中的粒子在 D 维搜索空间中, 记粒子 $i(i = 1, 2, \cdots, N)$ 的飞行速度为 $V_i = \{V_{i1}, V_{i2}, \cdots, V_{iD}\}^{\mathrm{T}}$, 当前位置为 $X_i = \{X_{i1}, X_{i2}, \cdots, X_{iD}\}^{\mathrm{T}}$, 粒子 i 迄今为止搜索到的最优位置为 $P_i = \{P_{i1}, P_{i2}, \cdots, P_{iD}\}^{\mathrm{T}}$, 群体迄今为止搜索到的最优位置为 $P_g = \{P_{g1}, P_{g2}, \cdots, P_{gD}\}^{\mathrm{T}}$. 为了避免进化迭代过程中粒子离开搜索空间, 通常限定 $V_{\min} \leqslant V_{id} \leqslant V_{\max}$. 标准 PSO 算法为

$$V_{id}(t+1) = \omega(t)V_{id} + c_1\text{rand}(\cdot)(P_{id} - X_{id}(t)) + c_2\text{rand}(\cdot)(P_{gd} - X_{id}(t)). \tag{4}$$

其中: t 为 PSO 当前迭代代数, 惯性权重 ω 决定了微粒先前速度对当前速度的影响程度, V_{id} 为粒子 i 飞行速度矢量 V 的第 $d\,(1 \leqslant d \leqslant D)$ 维分量, c_1 为微粒个体的加速权重系数, c_2 为微粒群体的加速权重系数, rand(·) 为 [0,1] 范围内均匀分布产生的随机数. 由式 (4) 确定的速度由 3 部分组成: 式 (4) 右端第 1 项表示微粒自身的惯性, 第 2 项表示微粒对个体经历的认知, 第 3 项表示微粒对群体经历的认知.

多模态寻优中, 为了使每一个局部最优值都成为微粒群算法中某些微粒的“必经之路”[9], 本文对标准 PSO 算法作以下改进: 1) 在理想情况下, 微粒群中的每个个体对自身的认知代表了多峰寻优中的每一个局部最优值, 因此令 $c_2 = 0$ 以求取所有的局部优化值, 而不仅是全局最优值; 2) 考虑到式 (4) 中随机函数 rand(·) 的作用主要是增大微粒运动的随机性, 以便跳出局部最优点, 为了让微粒群尽快收敛到各个局部最优值, 将式 (4) 中的 rand(·) 去掉; 3) 因为惯性权重线性递减策略 LDIW[10] 简单、直观、有较好的寻优能力, 可使算法在全局搜索与局部搜索之间取得平衡. 所以 ω 采用的 LDIW 策略为

$$\omega(t) = \omega_{\text{start}} - (\omega_{\text{start}} - \omega_{\text{end}})\frac{t}{t_{\max}}. \tag{5}$$

其中: $t_{\max}$ 为 PSO 最大迭代代数; ω_{start}, ω_{end} 分别为惯性权重的最大值和最小值. 经过改进, 式 (4) 变为

$$V_{id}(t+1) = \omega(t)V_{id}(t) + c_1(P_{id} - X_{id}(t)). \tag{6}$$

位置变化按下式进行:

$$X_{id}(t+1) = X_{id}(t) + V_{id}(t+1). \tag{7}$$

为了避免进化迭代过程中粒子离开搜索空间, 限定 $X_{\min} \leqslant X_{id} \leqslant X_{\max}$.

2.3 Powell 搜索法

Powell 搜索法又称方向加速法, 是一个为无约束最优化问题设计的不使用导数的非线性直接局部搜索方法, 被认为是直接搜索法中比较有效的一种方法, 它不需要计算导数而只利用函数值, 从一点出发沿两个方向进行一维搜索即可得到极小值. 非线性是指初始点的搜索方向不是固定的, 随着初始点位置的变化其加速方向会改变, 并不是沿着一条直线进行搜索的. Powell 搜索法的步骤如下:

Step 1: 将 PSO 最后一代搜索到的粒子作为初始点 $x^{(0)} \in S$, 并设定 Powell 直接法搜索精度 ε. 给定 D 个初始的线性无关搜索方向 $d^{(i)}(i = 0, 1, \cdots, D-1)$, 一般取为 D 个坐标轴方向, 当函数的维数为 D 时, 需产生 D 个线性无关的搜索方向 $d^{(0)}, d^{(1)}, \cdots, d^{(D-1)}$, 令 $k := 0$.

Step 2: 从 $x^{(0)}$ 出发, 依次沿搜索方向 $d^{(0)}, d^{(1)}, \cdots, d^{(D-1)}$ 进行精确的一维搜索, 得到 $x^{(1)}, x^{(2)}, \cdots, x^{(D)}$, 即

$$f(x^{(i)} + \alpha_i d^{(i)}) = \min_{\alpha \in S} f(x^{(i)} + \alpha d^{(i)}), \tag{8}$$

$$x^{(i+1)} = x^{(i)} + \alpha_i d^{(i)},\ i = 0, 1, \cdots, D, \tag{9}$$

其中 α 和 α_i 为步长. α_i 由精确线性搜索得到, 即 α_i 为上面的一维最优化问题的解, α_i 可能为负值, 表明精

确线性搜索在整个实数轴 R 上进行.

Step 3: 令 $d^{(D)}=x^{(D)}-x^{(0)}$. 若 $||d^{(D)}||\leqslant\varepsilon$, 则得到解 $x^{(D)}$, 计算结束; 否则, 从 $x^{(D)}$ 出发, 沿 $d^{(D)}$ 进行精确线性搜索, 得到 $x^{(D+1)}$.

Step 4: 计算最大下降量所确定的指标 tl 为

$$f(x^{(\mathrm{tl})})-f(x^{(\mathrm{tl}+1)})=\max_{0\leqslant i\leqslant D}\{f(x^{(i)})-f(x^{(i+1)})\}. \tag{10}$$

Step 5: 若

$$f(x^{(0)})-2f(x^{(D)})+f(2x^{(D)}-x^{(0)})\geqslant 2(f(x^{(\mathrm{tl})})-f(x^{(\mathrm{tl}+1)})) \tag{11}$$

成立, 则表明 $d^{(0)},d^{(1)},\cdots,d^{(D-1)}$ 仍然线性无关, 下一轮的搜索方向依然是 $d^{(0)},d^{(1)},\cdots,d^{(D-1)}$. 令 $x^{(0)}:=x^{(D+1)}$, $k:=k+1$, 转至 Step 2.

Step 6: 如果式 (11) 不成立, 则表明 $d^{(0)},d^{(1)},\cdots,d^{(D-1)}$ 线性相关, 令 $d^{(\mathrm{tl}+i)}:=d^{(\mathrm{tl}+i+1)}$, $i=0,1,\cdots,D-\mathrm{tl}-1$, 确保新产生的搜索方向线性无关, $x^{(0)}:=x^{(D+1)}$, $k:=k+1$, 转至 Step 2.

2.4 Powell-IPSO 混合算法流程

Powell-IPSO 混合算法将 PSO 和 Powell 算法的优点相结合, 避免其缺点, 强调精确性、可靠性和计算时间的平衡. Powell-IPSO 算法流程如下:

Step 1: 随机初始化规模为 N 的一群微粒, 并进行参数初始化. 初始化参数包括: PSO 算法最大迭代次数 $t_{\max}$, 混合算法最大迭代次数 M, Powell 法搜索精度 ε, Powell 法搜索概率 p_p 等.

Step 2: 评价每个微粒的适应度.

Step 3: 若总迭代代数大于 M, 则停止迭代, 输出找到的所有极值点; 否则, 转至 Step4.

Step 4: 若 PSO 迭代代数 $t\leqslant t_{\max}$, 则根据式 (6) 和 (7)对微粒进行速度、位置更新, 并更新微粒 P_i 及 P_g.

Step 5: 若 rand< p_p, 则对 PSO 最后一代粒子运用 Powell 法搜索极值点.

Step 6: 极值点库管理, 将新产生的极值点加入极值点库 JZK.

Step 7: 重新初始化微粒的位置及速度.

Step 8: 将 JZK 中已搜索到的全局极值点赋给已初始化的微粒, 并转至 Step 3.

2.5 极值点的同峰判断算子

2.5.1 极值点库管理

将 Powell 搜索法找到的极值点依次与极值点库 JZK 中所有极值点进行同峰判断 (当算法进行第 1 次迭代, 此时 JZK 为空时, 取 Powell 直接法搜索到的最优点位置作为 JZK 中的第 1 点). 若同峰且大于已存在极值点的适应度值, 则舍弃该极值点; 若同峰但小于该极值点的适应度值, 则替代该极值点; 若异峰, 则将其加入到 JZK 集合中.

2.5.2 基于适应度的同峰判断算子

新加入的极值点与已发现的某个极值点是否同峰一直是求多极值点优化问题的难点. 目前大多数多模态优化算法是引入小生境的进化算法[11], 这些算法都存在遗漏峰值、算法复杂等缺点, 且大多需要如“峰的个数”、“峰间距”等一些苛刻的先验条件. 鉴于此, 本文提出了一种基于适应度的同峰判断方法-山峰探索法来判断同峰极值点, 原理如下: 设 A 为 JZK 中的极值点, B 为 Powell 搜索法找到的极值点, 在 A 和 B 之间插入 $(n-1)$ 个点将 A 和 B 分成 n 等分, 计算这 n 段的同峰算子, 即

$$\mathrm{Tf}(i)=\mathrm{fit}(X_{i+1})-\mathrm{fit}(X_i), \tag{12}$$

其中 X_i, X_{i+1} 为 A, B 之间的相邻两点, $1\leqslant i\leqslant n$. 如果 Tf 的值出现由正变负 (Tf 为正的次数和为负的次数均大于常数 count), 则意味着在 A, B 两点之间有山峰存在, A, B 是异峰极值点; 否则为同峰极值点.

3 仿真实验及分析

实验仿真环境如下: Windows XP 系统, 1.66 GHz 主频的 INTEL 处理器, 760 MB 内存, Matlab 7.0 仿真软件. 为了评价 Powell-IPSO 算法的多峰寻优能力、求解精度及收敛速度等求解性能, 引入如表 1 所示的多个形态各异、难度较大的基准函数进行测试. 其中: Domain range 为函数取值范围, Global optimum 为全局最优值, NGO(number of global optima) 为全局最优解的个数, NLO (number of local optima) 为局部最优解的个数. 函数 f_1 和 f_5 求取的是函数最小值, 其余求取的是最大值. 表 1 中: f_1 是 Shubert 测试函数, 共有

表 1　基准函数

Function	Domain range	Global optimum	NGO	NLO
$f_1=\left\{\sum_{i=1}^{5}i\cos[(i+1)x_1+i]\right\}\left\{\sum_{i=1}^{5}i\cos[(i+1)x_2+i]\right\}$	$-10\leqslant x_1,x_2\leqslant 10$	$f_{1\min}\approx-186.73$	18	742
$f_2=660-(x_1^2+x_2-11)^2-(x_1+x_2^2-7)^2$	$-6\leqslant x_1,x_2\leqslant 6$	$f_{2\max}=660$	4	0
$f_3=\left[b/(a+[x_1^2+x_2^2])\right]^2+(x_1^2+x_2^2)^2, a=0.05, b=3$	$-5.12\leqslant x_1,x_2\leqslant 5.12$	$f_{3\max}=3\,600$	1	4
$f_4=2+x_1\sin(4\pi x_1)-x_2\sin(4\pi x_2+\pi)$	$-1\leqslant x_1,x_2\leqslant 1$	$f_{4\max}\approx 3.26$	4	32
$f_5=x_1^2+x_2^2-0.3\cos(3\pi x_1)+0.3\cos(4\pi x_2)+0.3$	$-1\leqslant x_1,x_2\leqslant 1$	$f_{5\min}\approx-0.24$	2	10

760个最小值, 其中18个为全局最小值, 要求算法具有较好的维持种群多样性的能力; f_2是改进的Himmelbau函数, 为典型线性不可分的等高、非等距的二维多峰函数, 性能不佳的算法很难搜索到全部4个全局峰值; f_3为Needle-in-a-haystack问题, 全局最优解被次优解所包围, 隔断了模式的重组过程, 使得搜索长期陷入局部极值点; f_4为多峰函数, 有4个全局最大值, 另有32个局部极大值; f_5为Bohachevsky测试函数, 最小解约为−0.24, 分布在(0, +0.23), (0, −0.23). 实验参数设置如下: 针对不同函数PSO的粒子数N为20∼105, 加速因子$c_1 = 1.494\,45$, 惯性权重$\omega_{\text{start}} = 0.9$, $\omega_{\text{end}} = 0.4$, $t_{\max} = 30$, $M = 2 \sim 10$, $\varepsilon = 2.0 \times 10^{-8}$.

3.1 Powell法搜索概率的确定

定理1 (Powell法收敛性定理)[12] 设f是连续可微的一致凸函数, 水平集$\Omega(x^{(0)}) = \{x|f(x) \leqslant f(x^{(0)})\}$有界, 则Powell搜索法产生的点列$\{x^{(k)}\}$收敛于$f$在$\Omega(x^{(0)})$中的惟一极小值点.

由定理1可知, 不同的微粒初始点经过Powell搜索法有可能收敛到同一全局或局部极值, 导致峰值遗漏, 找到所有极值点的可能性降低(会破坏PSO算法的多样性), 故在本算法中设置Powell法搜索概率p_p. 表2为针对f_1, f_2, f_3, f_5的基于不同p_p值算法对比. 其中: Mean$_G$为搜索到的全局解均值, Best为所有全局解的最优值, Worst为全局解的最差值, NSGO (number of searched global optima)为30次测试搜索到的全局解的平均个数, NSLO (number of searched local optima)为30次测试搜索到的局部解的平均个数. "-"为无局部解, Time为算法测试30次的平均运行时间.

1) 收敛精度分析.

由表2可知, 当p_p较小($p_p = 0.1$)时, 除f_1外均可以搜索到所有的极值点, 由于Powell法(主要用于提高算法的求解精度)使用较少, 全局寻优及局部寻优的精度较差(Mean$_G$, Worst, Mean$_L$较差); 当p_p较大($p_p = 0.9$)时, 全局解的求解精度较高, 但搜索到的极值点较少(如f_1)或者极值点遗漏(如f_5), 对于f_1的局部解均值较优是由于搜索到了较小的局部值, 对于f_5的局部解均值较差是因为遗漏了取值较小的局部值; 当$p_p = 0.5$时, 既具有较高的求解精度又能搜索到所有的极值点(除f_1外).

2) 收敛速度分析.

由表2可知, 对于f_2和f_3这样能搜索到所有极值点的函数, 当p_p越大时, Powell搜索法运用得越多, 算法运行时间是增加的, 但增加不多. 对于函数f_1和f_5这样极值点较多的函数, 当p_p较小($p_p = 0.1$)时, 能更多或更早地搜索到极值点, 导致需同峰判断的个体增多且存在较多的重复搜索, 因此算法的运行时间最长; 当p_p较大($p_p = 0.9$)时, 由于极值点的遗漏, 同峰判断的次数减少, 算法的运行时间最短; 同时可知, 同峰判断的时间开销较大.

综上所述, p_p为0.5较为合适.

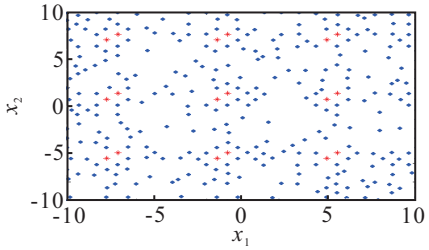
图1为当p_p=0.5时, 函数f_1和f_2解的分布, 图1中, "*"为全局点, "·"为局部点. 图2为当$p_p = 0.5$时, 函数f_1和f_4迭代过程中搜索到极值点的个数.

由图2(a)可见, 在迭代后期算法仍然可以搜索到较多的极值点, 搜索并没有停滞, 这说明该算法具有较强的全局搜索能力. 因此, 适当增大粒子数和延长混合算法最大迭代次数M可以搜索到f_1更多的极值点. 由图2(b)可知, 算法的收敛速度较快, 在第3次迭代时即可搜索到所有的极值点. 算法在迭代初期搜索到了较多的极值点, 但后期找到极值点的速率减慢, 这主要是由于后期搜索到的极值点与前面找到的极值点同峰, 表明在迭代后期具有较多的重复搜索. 但这些重复搜索提高了解的精度, 因为若两极值点同峰,

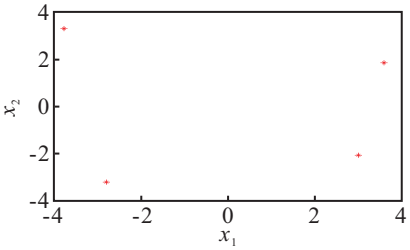
表2 基于不同p_p值的Powell-IPSO算法对比(测试30次)

Function	p_p	Mean$_G$	Best	Worst	Mean$_L$	NSGO	NSLO	Time/s
f_1 ($N = 105, M = 8$)	0.1	−185.534 535 198 455	−186.730 908 831 024	−131.159 190 084 600	−28.956 918 980 695	17.7	295.2	170.3
	0.5	−186.730 908 831 024	−186.730 908 831 024	−186.730 908 831 024	−32.874 508 266 700	18	218.8	132.7
	0.9	−186.730 908 831 024	−186.730 908 831 024	−186.730 908 831 024	−52.220 012 064 159	18	110.5	83.5
f_2 ($N = 20, M = 2$)	0.1	658.236 901 117 154	660	493.788 980 877 887	-	4	-	0.8
	0.5	660	660	660	-	4	-	1.2
	0.9	660	660	660	-	4	-	1.6
f_3 ($N = 30, M = 4$)	0.1	2 877.062 856 717 830	3 600	82.424 472 752 314	2748.782 337 384 844	1	4	19.3
	0.5	3 599.999 999 999 999	3 600	3 599.999 999 999 997	2 748.782 337 384 844	1	4	21.2
	0.9	3 599.999 999 999 999	3 600	3 599.999 999 999 997	2 748.782 337 384 844	1	4	22.3
f_5 ($N = 30, M = 4$)	0.1	−0.240 034 985 154	−0.240 034 985 154	−0.240 034 985 154	0.472 782 995 121	2	10	23.4
	0.5	−0.240 034 985 154	−0.240 034 985 154	−0.240 034 985 154	0.412 161 145 360	2	10	19.3
	0.9	−0.240 034 985 154	−0.240 034 985 154	−0.240 034 985 154	0.860 408 706 830	2	5.4	7.8

精度更高的极值点将替代精度低的极值点.

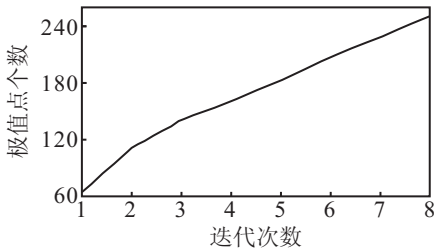


(a) f_1

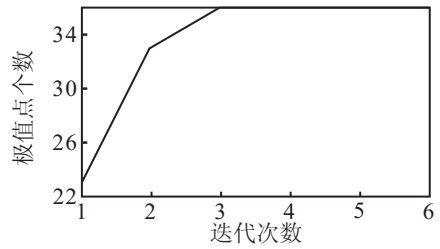


(b) f_2

图 1　函数 f_1 和 f_2 解的分布



(a) f_1



(b) f_4

图 2　函数 f_1 和 f_4 迭代过程中搜索到极值点个数

3.2　仿真实验

表 3 为针对函数 f_2 Powell-IPSO 算法与多种群协同进化策略 $m \times (1+\lambda)$-ES[2], 基于非线性共轭梯度法的混沌微粒群优化算法 NCGPSO[4]的计算结果对比. Time 为各算法测试 50 次的平均运行时间.

1) 收敛精度分析

由表 3 可知, 与 $m \times (1+\lambda)$-ES 和 NCGPSO 算法相比, 在 50 次测试中, Powell-IPSO 算法均能搜索到函数 f_2 的全部 4 个全局极值点, 并没有多余和遗漏极值点且全局极值点的函数值均为理论值 660. 因此, 与文献 [2,4] 相比, 解的质量得到了明显提高, 在搜索精度方面有明显优势.

2) 收敛速度分析

对于函数 f_2, NCGPSO 算法无平均计算时间用“-”表示, $m \times (1+\lambda)$-ES 算法在 Pentium III 的时间为 10.3 s. Powell-IPSO 算法的时间为 1.2 s (不同的计算机系统平均运行时间会有出入, 但不会相差 9 倍). 因此, 与文献 [2] 相比, Powell-IPSO 算法在收敛速度方面有明显优势. 表 4 为针对函数 f_4 的 4 个全局最大值, Powell-IPSO 算法与 NCGPSO[4]的计算结果对比. 限于篇幅, 其余 32 个局部点未列出, Powell-IPSO 算法测试 50 次的平均计算时间为 123.0 s.

由表 4 可知, 与 NCGPSO 算法相比, Powell-IPSO 算法每次均能准确找到函数 f_4 的 4 个理论全局极值 3.259 986 294 299 104, 解的质量得到了较大提高, 由于搜索到的全局极值点对应的函数值都相同, 解的一致性较好. 由于 NCGPSO 算法无平均计算时间, 无法在收敛速度方面进行比较. 图 3 为函数 f_4 解的分布. 从图 3 可知, Powell-IPSO 算法能够准确搜索到函数 f_4 的 36 个极值点. 综上所述, 针对函数 f_4, 仿真实验也表明了 Powell-IPSO 算法具有较强的全局搜索性能和较高的求解精度.

表 3　Powell-IPSO 算法与其他进化算法的计算结果对比 (函数 f_2 测试 50 次)

Method	序号	最优点集		函数值	Time/s
		x_1	x_2		
$m \times (1+\lambda)$-ES[2]	1	−2.805 116 02	3.131 279 53	659.999 999 96	10.3
	2	3.584 426 82	−1.848 071 22	659.999 999 96	
	3	− 3.779 329 09	−3.283 158 95	659.999 999 93	
	4	2.999 950 64	2.000 018 94	659.999 999 92	
NCGPSO[4]	1	−2.805 118	3.131 313	659.999 999 999 979 540	-
	2	3.584 436	−1.848 121	659.999 999 996 505 720	
	3	−3.779 306	−3.283 187	659.999 999 998 924 300	
	4	3.000 012	1.999 996	659.999 999 994 976 630	
Powell-IPSO ($N=20, M=2$)	1	−2.805 118 119 985	3.131 312 553 540	660	1.2
	2	3.584 428 341 811	−1.848 126 512 566	660	
	3	−3.779 310 271 030	−3.283 186 005 689	660	
	4	3.000 000 001 642	2.000 000 014 454	660	

表 4 Powell-IPSO 算法与 NCGPSO 的计算结果对比 (函数 f_4 测试 50 次)

method	序号	x_1	x_2	函数值
NCGPSO	1	0.634 920	0.634 927	3.259 986 293 038 022
	2	−0.634 921	−0.634 922	3.259 986 294 268 485
	3	−0.634 925	0.634 923	3.259 986 293 812 987
	4	0.634 922	−0.634 922	3.259 986 294 296 454
Powell-IPSO (N=60, M=6)	1	0.634 922 042 209	0.634 922 043 638	3.259 986 294 299 104
	2	−0.634 922 044 166	−0.634 922 043 039	3.259 986 294 299 104
	3	−0.634 922 042 862	0.634 922 043 325	3.259 986 294 299 104
	4	0.634 922 044 662	−0.634 922 043 602	3.259 986 294 299 104

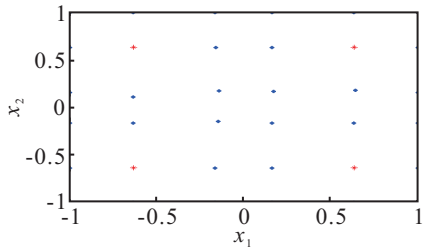


图 3 函数 f_4 解的分布

4 结 论

针对 MFO 问题, Powell-IPSO 混合算法克服了 PSO 算法求解精度较差、后期收敛速度慢等缺点, 发挥了传统数值优化方法 Powell 法在计算精度和收敛速度上的优势. 该算法通过改进的 PSO 算法对可行域内所有峰值进行搜索, 使得混合算法具有寻找所有极值点的全局搜索能力, 利用 Powell 法的强局部搜索能力, 提高了混合算法的精度和收敛速度. 由于该算法只利用了函数值信息而不需要计算导数, 适合求解可微和不可微的 MFO 问题, 与其他算法的比较表明了 Powell-IPSO 算法的有效性. 当函数具有较多局部极值点时, 需同峰判断的次数增加, 且存在较多的重复搜索, 因而导致算法时间开销增加较多. 如何避免极值点的重复搜索而不丢失极值点, 从而经过较少的迭代次数找到所有的极值点, 是需要进一步研究和解决的问题.

参考文献(References)

[1] 李敏强, 寇纪淞. 多模态函数优化的协同多群体遗传算法[J]. 自动化学报, 2002, 28(4): 497-504.
(Li M Q, Kou J S. Coordinate multi-population genetic algorithms for multi-modal function optimization[J]. Acta Automatica Sinica, 2002, 28(4): 497-504.)

[2] 王湘中, 喻寿益. 多模态函数优化的多种群进化策略[J]. 控制与决策, 2006, 21(3): 285-288.
(Wang X Z, Yu S Y. Multi-population evolution strategies for multi-modal function optimization[J]. Control and Decision, 2006, 21(3): 285-288.)

[3] 薛文涛, 吴晓蓓, 徐志良. 用于多峰函数优化的免疫粒子群网络算法[J]. 系统工程与电子技术, 2009, 31(3): 705-709.
(Xue W T, Wu X B, Xu Z L. Immune particle swarm network algorithm for multimodal function optimization[J]. Systems Engineering and Electronics, 2009, 31(3): 705-709.)

[4] 陈红安, 张英杰, 吴建辉. 基于非线性共轭梯度法的混沌微粒群优化算法[J]. 计算机应用, 2009, 29(12): 3273-3276.
(Chen H A, Zhang Y J, Wu J H. Chaotic particle swarm optimization algorithm based on the nonlinear conjugate gradient algorithm[J]. J of Computer Applications, 2009, 29(12): 3273-3276.)

[5] Kennedy J, Eberhart R C. Particle swarm optimization[C]. IEEE Conf on Neural Networks. Piscataway: IEEE Service Center, 1995: 1942-1948.

[6] 周龙甫, 师奕兵, 张伟. 拥有领导机制的改进粒子群算法[J]. 控制与决策, 2010, 25(10): 1463-1468.
(Zhou L F, Shi Y B, Zhang W. Improved particle swarm optimization with leadership[J]. Control and Decision, 2010, 25(10): 1463-1468.)

[7] PowellM J D. An efficient method for finding the minimum of a function of several variables without calculating derivatives[J]. Computer J, 1964, (7): 155-162.

[8] PowellM J D. A fast algorithm for nonlinearly constrained optimization calculations[M]. NewYork: Springer-Verlag, 1978: 144-175.

[9] 沈洪远, 彭小奇, 王俊年, 等. 基于改进的微粒群优化算法的山峰聚类法[J]. 模式识别与人工智能, 2006, 19(1): 89-93.
(Shen H Y, Peng X Q, Wang J N, et al. A mountain clustering based on improved PSO algorithm[J]. Pattern Recognition and Artificial Intelligence, 2006, 19(1): 89-93.)

[10] Eberhart R C, Shi Y. Guest editorial special issue on particle swarm optimization[J]. IEEE Trans on Evolutionary Computation, 2004, 8(3): 201-203.

(下转第354页)