文章编号: 1001-0920(2011)07-1060-05

Rosenbrock 搜索与动态惯性权重粒子群混合优化算法

贾树晋[1], 杜 斌[2]

(1. 上海交通大学 自动化系, 上海 200240; 2. 上海宝钢研究院 自动化所, 上海 201900)

摘 要: 为了提高复杂优化问题的优化精度和鲁棒性能, 提出两种将Rosenbrock搜索与动态惯性权重粒子群(DIPSO)相结合的混合算法, 即“协同”与“接力”混合算法. 两种算法充分利用了Rosenbrock搜索算法强大的局部搜索能力和DIPSO算法的全局寻优能力, 很好地平衡了算法的全局“探索”与局部“开发”. 通过4个典型基准函数的实验研究, 表明了所提出的算法具有优化精度高、鲁棒性强等特点, 适合于对高维多峰函数进行优化.

关键词: Rosenbrock搜索算法; 粒子群优化算法; 高维多峰函数优化

中图分类号: TP18 **文献标识码:** A

Hybrid optimized algorithms based on the Rosenbrock search method and dynamic inertia weight PSO

JIA Shu-jin[1], *DU Bin*[2]

(1. Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China; 2. Research Institute of Automation, Academy of Baosteel, Shanghai 201900, China. Correspondent: JIA Shu-jin, E-mail: jiashujin_1@163.com)

Abstract: In order to improve the accuracy and robustness performance of complex optimization problems, two hybrid algorithms, which combine the Rosenbrock search method and dynamic inertia weight PSO(DIPSO), are proposed in this paper. The algorithms make full use of the powerful local search ability of the Rosenbrock search method and the global optimization ability of DIPSO algorithm, which well balance the global “exploration” and the local “exploitation”. The experiment study in four typical benchmark functions show that the proposed algorithms have the characteristics of high accuracy and robustness, and are suitable for optimizing high-dimensional and multimodal functions.

Key words: Rosenbrock search method; PSO; high-dimensional and multimodal function optimization

1 引 言

工程中的大量问题最终可归结为函数的优化问题,这些函数大多存在高维、非线性、非凸和不可微等复杂特性, 而且有的属于多峰函数, 存在大量局部极值,使得基于导数的优化算法无能为力. Rosenbrock搜索算法是一种求解无约束最优化问题的直接方法[1-2], 它不需要函数的导数信息, 具有计算简单、收敛速度快、优化精度高等特点, 缺点是对初始值较敏感, 易陷入局部最优[2]. 基于群体智能理论的演化计算方法——粒子群算法(PSO)是由Kennedy和Eberhart于1995年提出的一种新型智能优化算法[3], 与其他智能优化算法(如遗传算法)相比, 具有算法简单、可调参数少、易实现和计算效率高等特点, 特别适合于连续函数的优化[4]. 但在解决复杂优化问题时存在早熟收敛、优化精度低与鲁棒性差等问题[4].

为了提高复杂问题的优化精度和鲁棒性能, 本文将Rosenbrock搜索算法与动态惯性权重粒子群算法相结合, 提出了两种混合优化算法(Rb-DIPSO). 这两种算法充分利用了Rosenbrock搜索算法强大的局部“开发”能力和粒子群算法的全局“探索”能力, 既有效地克服了Rosenbrock算法对初始值敏感、易陷入局部最优的缺点, 又避免了粒子群算法后期可能出现的早熟收敛、优化精度低的问题, 同时提高了算法的鲁棒性. 实验结果验证了本文算法的有效性.

2 混合优化算法

2.1 Rosenbrock 搜索算法

Rosenbrock搜索算法是无约束优化中不使用导数信息的直接方法, 由Rosenbrock于1960年首先提出[1], 后经Bazaraa等人[2]进行了改进. Rosenbrock算

收稿日期: 2010-04-12; **修回日期:** 2010-07-21.

作者简介: 贾树晋(1982–), 男, 博士生, 从事智能优化算法、生产计划与调度等研究; 杜斌(1957–), 男, 教授级高工, 博士生导师, 从事过程控制及生产计划优化等研究.

法又称转轴法, 包括探测和构造搜索方向 2 个阶段, 算法步骤如下[5]:

第 1 阶段: 探测阶段.

Step 1: 给定初始点 $x^{(1)}\in \boldsymbol{R}^n$, 单位正交方向 $d^{(1)}$, $d^{(2)},\cdots,d^{(n)}$, 一般取坐标方向, 步长 $\delta_1^{(0)},\delta_2^{(0)},\cdots,\delta_n^{(0)}$, 放大因子 $\alpha>1$, 缩减因子 $\beta\in(-1,0)$, 允许误差 $\varepsilon>0$. 置 $y^{(1)}=x^{(1)}$, $k=1$, $j=1$, $\delta_i=\delta_i^{(0)}$, $i=1,2,\cdots,n$.

Step 2: 如果 $f(y^{(j)}+\delta_j d^{(j)})<f(y^{(j)})$, 则令 $y^{(j+1)}=y^{(j)}$, $\delta_j:=\alpha\delta_j$; 反之, 则令 $y^{(j+1)}=y^{(j)}$, $\delta_j:=\beta\delta_j$.

Step 3: 如果 $j<n$, 则置 $j:=j+1$, 转 Step 2; 否则, 转 Step 4.

Step 4: 如果 $f(y^{(n+1)})<f(y^{(1)})$, 则令 $y^{(1)}=y^{(n+1)}$, 置 $j=1$, 转 Step 2; 如果 $f(y^{(n+1)})=f(y^{(1)})$, 则转 Step 5.

Step 5: 如果 $f(y^{(n+1)})<f(x^{(k)})$, 则转 Step 6; 否则, 如果对于每个 j, $|\delta_j|\leqslant\varepsilon$ 成立, 则停止计算, $x^{(k)}$ 作为最优解的估计, 反之则令 $y^{(1)}=y^{(n+1)}$, 置 $j=1$, 转 Step 2.

Step 6: 令 $x^{(k+1)}=y^{(n+1)}$, 如果 $\parallel x^{(k+1)}-x^{(k)}\parallel\leqslant\varepsilon$, 则取 $x^{(k+1)}$ 作为极小点的估计, 停止计算; 否则, 转 Step 7.

第 2 阶段: 构造新的正交搜索方向.

Step 7: 根据上一次的探测结果, 有 $x^{(k+1)}=x^{(k)}+\sum_{i=1}^{n}\lambda_i d^{(i)}$, 其中 λ_i 是在整个探测阶段中所有沿方向 $d^{(i)}$ 的步长的代数和. 定义一组方向 $p^{(1)},p^{(2)},\cdots,p^{(n)}$ 如下:

$$p^{(j)}=\begin{cases} d^{(j)},\ \lambda_j=0;\\ \sum\limits_{i=j}^{n}\lambda_i d^{(i)},\ \lambda_j\neq 0.\end{cases}$$

然后利用 Gram-Schmidt 正交化方法, 将向量组 $\{p^{(j)}\}$ 正交化, 令

$$q^{(j)}=\begin{cases} p^{(j)},\ j=1;\\ p^{(j)}-\sum\limits_{i=1}^{j-1}\dfrac{q^{(i)^{\mathrm{T}}}p^{(j)}}{q^{(i)^{\mathrm{T}}}q^{(i)}}q^{(i)},\ j\geqslant 2.\end{cases}$$

之后单位化, 令

$$\bar{d}^{(j)}=\frac{q^{(j)}}{\parallel q^{(j)}\parallel},$$

得到 n 个新的正交搜索方向.

Step 8: 令 $d^{(j)}=\bar{d}^{(j)}$, $\delta_j=\delta_j^{(0)}$, $j=1,2,\cdots,n$, 置 $y^{(1)}=x^{(k+1)}$, $k:=k+1$, $j=1$, 返回 Step 2.

2.2　粒子群优化算法

2.2.1　基本原理

粒子群优化算法源于对群体行为的模拟. 在 D 维搜索空间中, 粒子种群数为 PS, $V_i=(v_{i,1},v_{i,2},\cdots,v_{i,D})$ 和 $X_i=(x_{i,1},x_{i,2},\cdots,x_{i,D})$ 分别表示第 i 个粒子的速度和位置, 群体中的每个粒子代表优化问题的一个候选解, 粒子位置对应的目标函数值作为其适应值. 每个粒子在搜索时, 考虑了自身速度惯性, 自身搜索到的历史最好位置 $P_i=(p_{i,1},p_{i,2},\cdots,p_{i,D})$ 以及群体内所有粒子的历史最好位置 $P_g=(p_{g,1},p_{g,2},\cdots,p_{g,D})$, 在此基础上进行速度和位置的更新. 第 i 个粒子的速度和位置表示如下[4]:

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

$$x_{i,j}(k+1)=x_{i,j}(k)+v_{i,j}(k+1).\tag{2}$$

式中: $i=1,2,\cdots,\mathrm{PS}$; $j=1,2,\cdots,N$; ω 为惯性权重; c_1 和 c_2 为加速因子; r_1 和 r_2 为在 [0,1] 之间服从均匀分布的随机数; k 为迭代次数.

2.2.2　动态惯性权重粒子群算法

研究表明, 参数集 $\{\omega,c_1,c_2\}$ 及速度更新策略对粒子群算法的性能具有重要影响, 已有许多文献[6-7]对其进行了理论和实验研究. 本文在以往研究的基础上, 提出一种动态惯性权重粒子群算法 (DIPSO), 核心思想是: 每个粒子的 ω 根据其适应值大小进行非线性动态调整, 适应值好的粒子选择较小的 ω, 以增强局部搜索能力; 适应值差的粒子选择较大的 ω, 以增强算法对新区域的搜索. 动态惯性权重的表达式为

$$\omega=\omega_{\min}+\frac{(\omega_{\max}-\omega_{\min})(f-f_{\min})}{f_{\max}-f_{\min}}.\tag{3}$$

其中: $\omega_{\max}$ 和 $\omega_{\min}$ 分别为惯性权重的上下限; $f_{\max}$ 和 $f_{\min}$ 对应每代粒子的最大最小目标函数值, f 为每个粒子对应的目标函数值; 速度更新策略[8]如下:

$$v_{i,j}(k+1)=\begin{cases}\omega v_{i,j}(k)+c_1r_1[p_{i,j}(k)-x_{i,j}(k)]+\\ c_2r_2[p_{g,j}(k)-x_{i,j}(k)],\ x_{\min}<x_{i,j}(k)<x_{\max};\\ 0,\ \text{else}.\end{cases}\tag{4}$$

其中 $x_{\max}$ 和 $x_{\min}$ 为粒子搜索空间的上下限. 这种速度更新策略的好处在于, 当粒子处于搜索空间边界时, 将其速度设为 0, 通过 P_i, P_g 的共同作用将其拉回搜索区间内, 以减少粒子在“边界徘徊”的时间, 可有效提高搜索效率.

2.3　混合优化算法

2.3.1　“协同”混合算法 (Rb-DIPSO1)

“协同”混合算法是指在每个迭代周期内, 首先使用 DIPSO 探索较大的解空间, 然后使用 Rosenbrock 搜索算法对最优粒子的邻域进行精细开发, 体现了“探索一点, 开发一点”的思想.

算法步骤如下:

Step 1: 设置 Rosenbrock 搜索算法和 DIPSO 算法的参数, 并初始化种群中各粒子的位置和速度.

Step 2: 评价初始种群中的所有粒子, 将粒子的当前位置设为 P_i, 初始种群中的最佳位置设为 P_g.

Step 3: 使用 Rosenbrock 搜索算法对最优个体进行局部搜索, 并用新位置替换原种群最优位置 P_g.

Step 4: 按式 (4) 和 (2) 更新粒子的速度和位置, 评价新种群中的所有粒子, 并更新 P_i 与 P_g.

Step 5: 若满足算法终止条件, 则输出 P_g 及其适应值, 并停止算法; 否则, 转 Step 3.

2.3.2 “接力”混合算法 (Rb-DIPSO2)

“接力”混合算法是指经 DIPSO充分探索后, 再将“接力棒”交给 Rosenbrock 搜索的算法, 完成对部分优秀粒子邻域的深度开发, 体现了“探索-开发-探索”的接力思想.

算法步骤如下:

Step 1: 设置 Rosenbrock 搜索算法和 DIPSO 算法的参数, 并初始化种群中各粒子的位置和速度.

Step 2: 评价初始种群中的所有粒子, 将粒子的当前位置设为 P_i, 初始种群中的最佳位置设为 P_g.

Step 3: 按式 (4) 和 (2) 更新粒子的速度和位置, 评价新种群中的所有粒子, 并更新 P_i 与 P_g.

Step 4: 若种群最优粒子适应值连续 n 代未得到改善 (接力条件), 则对种群粒子按适应值优劣进行排序, 挑选前 S 个精英粒子构成候选粒子, 置 $i=1$, 转 Step 5; 否则, 转 Step 3 继续执行 DIPSO 算法.

Step 5: 对第 i 个粒子执行 Rosenbrock 搜索算法, 直到 Rosenbrock 算法收敛, 用得到的新位置替代原粒子最优位置 P_i, 同时更新 P_g.

Step 6: 若满足算法终止条件, 则输出种群最优位置 P_g 及其适应值, 并停止算法; 否则, 转 Step 7.

Step 7: 若候选粒子均完成 Rosenbrock 搜索, 则转 Step 3; 否则, 置 $i=i+1$, 转 Step 5.

3 实验研究

本文选用 4 个典型的基准函数[6-7,9]比较所提出的算法与其他几种混合算法 (SM-IPSO[9], DEPSO[10], Powell-LLPSO[11], NM-PSO[12])的性能. 为便于分析, 将上述混合算法所对应的确定性局部搜索算法 (Rosenbrock 算法, Powell 算法, 单纯形法 (记为 NM-Simplex)) 一并进行比较研究. 基准函数信息见表 1.

表 1 基准函数信息

函数名称	维数 D	范围	最优值	阈值	特 性
Sphere	100	$[-100, 100]$	0	0.01	高维单峰
Rosenbrock	30	$[-30, 60]$	0	1	非凸病态
Griewank	100	$[-600, 600]$	0	0.01	高维多峰
Ackley	100	$[-30, 30]$	0	0.01	高维多峰

算法性能指标包括: 1) 优化精度. 固定函数评价次数 (3×10^5) 作为所有算法的终止条件, 每个测试函数与算法独立运行 20 次, 用最佳目标函数值均值和标准差作为评价准则. 2) 鲁棒性. 通过统计算法达到规定阈值的实验次数占总实验次数的比例进行评价. 3) 收敛速度. 算法在固定函数评价次数内满足规定阈值所需的平均函数评价次数. 阈值如表 1 所示.

本文算法参数设置如下: $\omega_{\max}=0.9$, $\omega_{\min}=0.558$, $c_1=2.04$, $c_2=0.94$, “接力”算法中 $n=100$, $S=5$; 其余算法使用原文推荐参数. 种群大小 PS 如表 2 所示. 其中 NM-PSO 的 PS 与函数维数有关, DEPSO 对种群大小较敏感, 根据原文选用 30 和 100 两种 PS 进行实验, 只记录较好的结果.

3 种性能指标的实验结果如表 2 和表 3 所示 (最好结果用黑体标出), 混合算法运行 20 次时随函数评价次数的平均进化曲线如图 1 所示, 从中可以得到以下结论:

1) 3 种局部搜索算法中, Rosenbrock 搜索算法的优化精度和鲁棒性最好, Powell 算法的收敛速度最快, 单纯形算法的性能最差, 这一差异从一定程度上影响了相应混合算法的性能. 混合算法的总体性能好于对应的局部搜索算法, 其中 Rb-DIPSO, SM-IPSO 和 DEPSO 的表现最好, 鲁棒性能相当, 收敛速度各有擅长, 优化精度 Rb-DIPSO 更胜一筹. 但对单峰函数 Sphere 而言, Rb-DIPSO 的收敛速度指标较差.

2) Rb-DIPSO 2 对 Rosenbrock 函数的优化性能最好. 因为其他混合算法本质上都属于“协同”混合策略, 所以对 Rosenbrock 函数而言, “接力”混合策略可能比“协同”混合策略更好.

表 2 算法优化精度比较

算 法	PS	Sphere	Rosenbrock	Griewank	Ackley
Rb-DIPSO1	30	3.75 e-17(2.51 e-18)	7.98 e-01(1.68 e+00)	**2.22 e-17(4.96 e-17)**	**8.18 e-14(2.33 e-14)**
Rb-DIPSO 2	30	2.83 e-14(2.78 e-15)	**6.75 e-16(2.13 e-15)**	4.12e-14(6.21e-14)	4.12 e-08(1.94 e-08)
Powell-LLPSO	30	2.93 e-11(3.63 e-11)	7.98 e-01(1.78 e+00)	4.64 e-03(3.32 e-02)	2.93 e+00(8.37 e+00)
NM-PSO	$3D$+1	3.30 e-11(7.38 e-11)	1.37 e+01(1.02 e+01)	5.50 e-02(9.80 e-02)	3.21 e-01(2.58 e-01)
SM-IPSO	6×10	**1.69 e-17(2.67 e-17)**	7.90 e-01(3.40 e-01)[9]	1.50 e-08(1.00 e-08)[9]	2.60 e-03(1.00 e-03)[9]
DEPSO	30/100	9.72 e-15(1.55 e-14)	7.98 e-01(1.78 e+00)	5.06 e-07(5.75 e-07)	2.03 e-03(2.12 e-03)
Rosenbrock	–	1.17 e-12(1.06 e-12)	2.60 e+00(2.35 e+00)	5.60 e-02(3.21 e-02)	7.76 e+00(1.09 e+01)
Powell	–	8.80e-11(8.21e-11)	6.18e+01(8.78e+01)	2.73e-01(4.05e-01)	1.93e+01(8.90e-02)
NM-Simplex	–	1.05 e+04(4.87 e+03)	7.03 e+05(1.32 e+05)	6.09 e+02(1.85 e+02)	1.94 e+01(8.94 e-02)

表 3　算法鲁棒性及收敛速度比较

算　法	Sphere	Rosenbrock	Griewank	Ackley
Rb-DIPSO1	100%(125 760)	80%(121 430)	100%(108 760)	100%(114 850)
Rb-DIPSO2	100%(118 040)	**100%(115 240)**	100%(106 650)	**100%(111 670)**
Powell-LLPSO	**100%(4 672)**	80%(56 850)	60%(16 880)	30%(49 640)
NM-PSO	100%(112 750)	0%(–)	65%(179 290)	35%(279 540)
SM-IPSO	100%(12 460)	85%(259 580)[9]	**100%(42 460)**[9]	100%(202 470)[9]
DEPSO	100%(92 380)	80%(153 330)	100%(161 200)	100%(196 850)
Rosenbrock	100%(127 500)	40%(144 065)	80%(141 570)	60%(141 020)
Powell	100%(4 670)	40%(32 350)	40%(8 180)	0%(–)
NM-Simplex	0%(–)	0%(–)	0%(–)	0%(–)

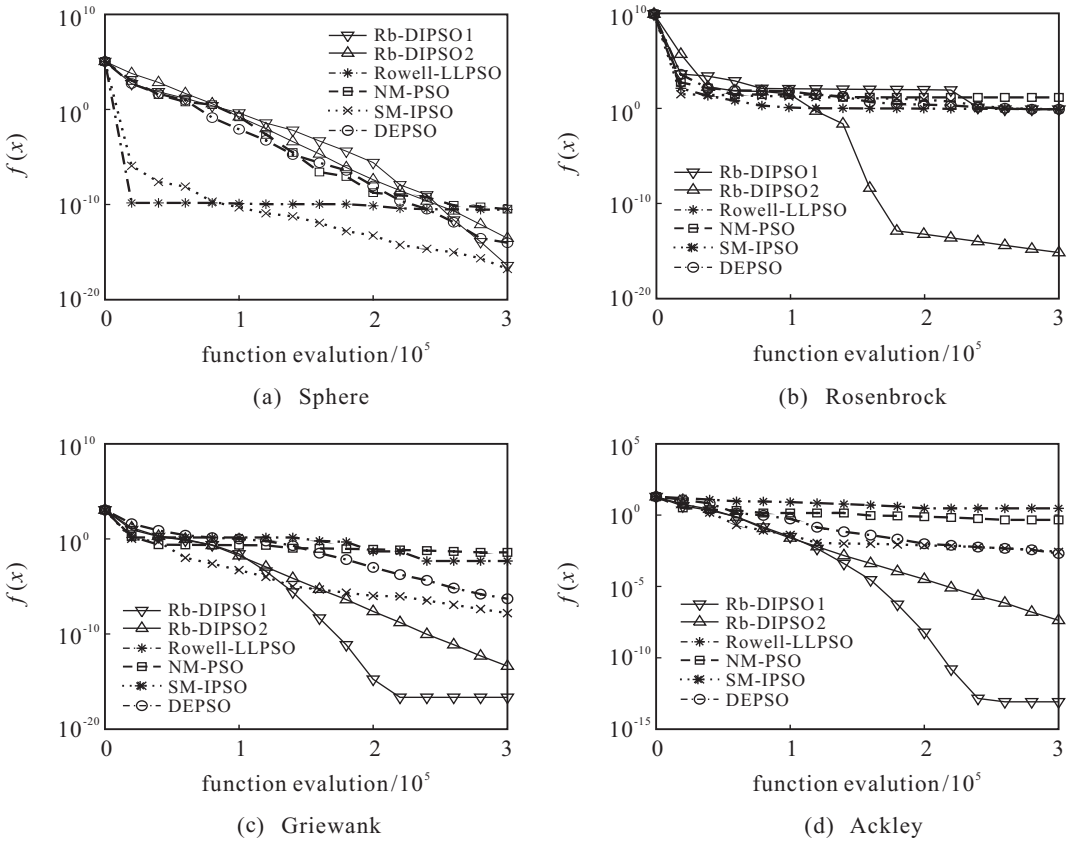

图 1　6种混合算法随函数评价次数的平均进化曲线

3) 由图 1 可以看出, 在进化初期, Rb-DIPSO 1 在 DIPSO 与 Rosenbrock 算法协同作用下, 其进化速度要优于仅有 DIPSO 起作用的 Rb-DIPSO 2; 在进化中期, DIPSO 有早熟收敛的趋势, Rb-DIPSO 1 进化速度有所下降, 而 Rb-DIPSO 2 则通过作用在精英粒子上的 Rosenbrock 局部搜索, 仍保持相对较快的进化速度; 在进化后期, Rb-DIPSO 1 得益于 DIPSO 的动态惯性权重, 使适应值较差的粒子仍具有较大的速度惯性, 有利于跳出收敛态, 找到新的最优点, 继而通过 Rosenbrock 与 DIPSO 的协同作用改善进化速度, 而此时由于 Rb-DIPSO 2 只有一种算法起作用, 进化速度相对较差, 最终其优化精度次于 Rb-DIPSO 1. 由于算法收敛时发生在进化中期, Rb-DIPSO 2 的收敛速度好于 Rb-DIPSO 1.

4) Powell-LLPSO 的收敛速度最快, 但它对高维多峰函数的优化精度和鲁棒性较差, 适合于优化单峰函数和低维多峰函数 (如 30 维).

5) NM-PSO 和 SM-IPSO 都选用单纯形法作为局部搜索算法, 得益于微粒速度概率逃逸算子和单纯形顶点的强制扩张算子的引入, SM-IPSO 的优化性能优于 NM-PSO. 相比 Rb-DIPSO, SM-IPSO 对高维多峰函数的优化精度较差, 这主要是由于局部搜索算法的性能差距. 另外, 应注意 SM-IPSO 进化初期的进化速度优于 Rb-DIPSO, 原因是 Rb-DIPSO 进化初期起主要作用的是 DIPSO, 而 SM-IPSO 则是奇数群体 (粒子群) 与偶数群体 (单纯形) 并行起作用, 故而能搜索更大的空间. 待 SM-IPSO 粒子群接近早熟收敛时, 单纯形算法也可能陷入局部最优, 尽管在微粒速度概率逃逸算子和单纯形顶点强制扩张算子的作用下不致于收敛到局部最优, 但进化速度明显放缓. 根据结论 3),

Rb-DIPSO在进化中后期凭借Rosenbrock强大的局部搜索能力和具有动态惯性权重的DIPSO, 仍能保持较快的进化速度, 最终得到更好的优化精度.

6) 不同于其他混合算法, DEPSO提供了一种新颖的自适应混合策略, 它没有使用局部搜索算法, 而是通过周期性计算“相对成功率”在粒子群与差分进化之间作选择. 在进化初期, 粒子群算法起主要作用, 在进化后期, 差分进化算法起主要作用[10]. 由于进化机制的不同, 在粒子群早熟收敛后通过差分进化算法作用于粒子个体最优种群, 仍有一定的进化空间, 因此进化曲线有较明显的“两段式”进化特征. 与Rb-DIPSO相比, DEPSO对高维多峰函数的优化精度较差, 主要原因是: ① 当差分进化算法也发生收敛后, DEPSO缺少跳出收敛态的有效机制; ② 对高维多峰函数而言, 进化后期的精细邻域搜索对提高优化精度至关重要[4], 而DEPSO缺少有效局部搜索机制, 使得在进化后期无法进一步提高精度.

4 结 论

本文将Rosenbrock搜索算法与动态惯性权重粒子群算法相结合, 提出了两种不同的混合算法. 通过在4个典型基准函数上的实验研究表明, 相对于其他几种混合算法, 该类算法具有优化精度高、鲁棒性强、适合优化高维多峰函数的特性. 然而, 该类算法存在进化初期进化速度偏低以及对单峰函数优化时收敛速度较慢的不足, 如何对其进行改善是需要进一步研究的课题. 同时,“接力”混合算法中候选粒子的选择策略及接力条件对算法的影响也有待进一步研究.

参考文献(References)

[1] Rosenbrock H H. An automatic method for finding the greatest or least value of a function[J]. Computer J, 1960, 3(3): 175-184.

[2] Bazaraa M S, Shetty C M. Nonlinear programming: Theory and algorithms[M]. New York: Wiley, 1979.

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

[4] Kennedy J, Eberhart R C, Shi Y. Swarm intelligence[M]. San Francisco: Morgan Kaufman Publishers, 2001.

[5] 陈宝林. 最优化理论与算法[M]. 2版. 北京: 清华大学出版社, 2005.
(Chen B L. Optimization theory and algorithms [M]. 2nd ed. Beijing: Tsinghua University Press, 2005.)

[6] Eberhart R C, Shi Y. Comparing inertia weights and constriction factors in particle swarm optimization[C]. Proc 2000 Conf Evolutionary Computation. San Diego, 2000: 84-88.

[7] Clerc M, Kennedy J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space[J]. IEEE Trans on Evolutionary Computation, 2002, 6(1): 58-73.

[8] Carlisle A, Dozier G. An off-the-shelf PSO[C]. Proc of the Workshop on Particle Swarm Optimization. Indianapolis, 2001: 1-6.

[9] 张勇, 巩敦卫, 张婉秋. 一种基于单纯形法的改进微粒群优化算法及其收敛性分析[J]. 自动化学报, 2009, 35(3): 289-298.
(Zhang Y, Gong D W, Zhang W Q. A simplex method based improved particle swarm optimization and analysis on its global convergence[J]. Acta Automatica Sinica, 2009, 35(3): 289-298.)

[10] Xin Bin, Chen Jie, Peng Zhihong, et al. An adaptive hybrid optimizer based on particle swarm and differential evolution for goal optimization[J]. Science China Information Sciences, 2010, 53(5): 980-989.

[11] 刘国志, 苗晨. Powell搜索法和局部收缩微粒群算法的混合算法[J]. 辽宁石油化工大学学报, 2008, 28(3): 70-74.
(Liu G Z, Miao C. Hybrid powell search and the local constrain approach particle swarm optimization with linear varying inertia weight for unconstrained optimization[J]. J of Liaoning University of Petroleum & Chemical Technology, 2008, 28(3): 70-74.)

[12] Fan S K S, Zahara E. A hybrid simplex search and particle swarm optimization for unstrained optimization[J]. European J of Operational Research, 2007, 181(2): 527-548.

(上接第1059页)

[11] 于义彬, 王本德, 柳彭, 等. 具有不确定信息的风险型多目标决策理论及应用[J]. 中国管理科学, 2003, 11(6): 9-13.
(Yu Y B, Wang B D, Liu P, et al. Risky multiobjective decision-making theory and its application[J]. Chinese J of Management Science, 2003, 11(6): 9-13.)

[12] Slowinski R, Vanderpooten D. A generalized definition of rough approximations based on similarity[J]. IEEE Trans on Knowledge and Data Engineering, 2000, 12(2): 331-336.