

多粒子角色协同作用的混合粒子群优化算法

吴逸庭*, 戴月明, 纪志成, 吴定会

(江南大学 物联网工程学院, 江苏 无锡 214122)

(*通信作者电子邮箱 jzjwyt@sina.com)

摘要:针对粒子群优化(PSO)算法易陷入局部最优和后期收敛速度慢的问题,提出一种多粒子角色协同作用的混合粒子群算法(MPRPSO)。引入粒子角色的概念,将种群粒子分成探索粒子(EP)、巡逻粒子(PP)和局部开发粒子(LEP)三类角色,在每次迭代中利用探索粒子以标准 PSO 算法搜索解空间,用基于混沌的巡逻粒子加强全局搜索,并在陷入局部最优时替代部分探索粒子,恢复种群活力。最后通过局部开发粒子的单维异步邻域搜索加强算法局部搜索能力,加快收敛。实验独立运行 30 次,所提算法在粒子角色比例为 0.8:0.1:0.1 的条件下,在 Sphere、Rosenbrock、Ackley 和 Quadric 函数中获得的平均值分别为 $2.352E-72$ 、 $4.678E-29$ 、 $7.780E-14$ 和 $2.909E-14$,尤其在 Rastrigrin 与 Griewank 函数中能收敛到最优解 0,优于其他对比算法。实验结果表明,所提算法在优化性能上有所提高,并有一定的鲁棒性。

关键词:粒子群优化算法;粒子角色;混沌;邻域搜索;协同

中图分类号: TP301.6; TP18 **文献标志码:** A

Hybrid particle swarm optimization algorithm with cooperation of multiple particle roles

WU Yiting*, DAI Yueming, JI Zhicheng, WU Dinghui

(School of Internet of Things Engineering, Jiangnan University, Wuxi Jiangsu 214122, China)

Abstract: Concerning the problem that Particle Swarm Optimization (PSO) falls into local minima easily and converges slowly at the last stage, a kind of hybrid PSO algorithm with cooperation of multiple particle roles (MPRPSO) was proposed. The concept of particle roles was introduced into the algorithm to divide the population into three roles: Exploring Particle (EP), Patrolling Particle (PP) and Local Exploiting Particle (LEP). In each iteration, EP was used to search the solution space by the standard PSO algorithm, and then PP which was based on chaos was used to strengthen the global search capability and replace some EPs to restore population vitality when the algorithm trapped in local optimum. Finally, LEP was used to strengthen the local search to accelerate convergence by unidimensional asynchronous neighborhood search. The 30 times independent runs in the experiment show that, the proposed algorithm in the conditions that particle roles ratio is 0.8:0.1:0.1 has the mean value of $2.352E-72$, $4.678E-29$, $7.780E-14$ and $2.909E-14$ respectively in Sphere, Rosenbrock, Ackley and Quadric, and can converge to the optimal solution of 0 in Rastrigrin and Griewank, which is better than the other contrastive algorithms. The experimental results show that proposed algorithm improves the optimal performance with certain robustness.

Key words: Particle Swarm Optimization (PSO) algorithm; particle role; chaos; neighborhood search; cooperation

0 引言

粒子群优化(Particle Swarm Optimization, PSO)算法是由美国社会心理学家 Kennedy 博士和电气工程师 Eberhart 博士在 1995 年提出的一种基于群体智能的算法,其主要思想来源于对鸟类群体行为的研究^[1]。由于其概念简单,相比其他的算法具有控制参数少、容易实现等优点,目前已经被社会科学、自然科学等各个领域的学者广泛应用^[2]。

类似于其他的群智能算法,PSO 算法在迭代中,极易陷入局部最优或者过早收敛而产生“早熟现象”。近年来,为了克服算法的早熟收敛,提升算法的精度,不少学者做了大量的工作。文献[3]提出一种自适应排斥因子,控制个体最优和全

局最优的距离以提升种群跳出局部最优的能力,并通过实验表明其有效性;文献[4]结合了并行定向变异与序列二次规划法,在一定程度上实现个体搜索与全局搜索的统一;文献[5]则提出一种多尺度协同变异的粒子群算法,在算法不同阶段采用相应的变异策略,提高了算法的收敛速度和稳定性;文献[6]基于混合择优机制,依概率根据解集信息熵或 Sigma 值确定其全局极值,提升了算法的多样性和分布性;文献[7]提出了一种基于增强多样性的优化算法,有效地提高了算法的收敛精度和寻优能力,并在实际应用中取得不错的效果;文献[8]融合了混沌云模型和粒子群算法的思想,提出了基于混沌云模型和粒子群算法的混合优化算法,获得了比较理想的效果。

收稿日期:2014-02-27;修回日期:2014-04-13。

基金项目:国家 863 计划项目(2013AA040405);江苏省产学研联合创新基金资助项目(BY2012055)。

作者简介:吴逸庭(1988-),男,江苏无锡人,硕士研究生,主要研究方向:人工智能、软件工程、软件测试;戴月明(1964-),男,江苏常熟人,副教授,硕士,主要研究方向:人工智能、模式识别、数据挖掘;纪志成(1959-),男,浙江杭州人,教授,博士,主要研究方向:风能转换系统控制、复杂非线性系统控制、网络控制、智能控制;吴定会(1970-),男,安徽庐江人,副教授,博士,主要研究方向:新能源控制。

由于粒子群算法存在着收敛速度与早熟收敛的矛盾,大部分算法改进仅着眼于其中某一方面,这使得改进的效果十分有限。为了有效地克服粒子群优化算法陷入早熟收敛和后期收敛速度较慢的缺陷,受到文献[9]的启发,本文在此提出了一种新的基于种群多粒子角色协同作用的混合粒子群算法(hybrid particle swarm optimization algorithm with cooperation of multiple particle roles, MPRPSO),对种群中的粒子进行分类,结合多种搜索策略的优点进行分工合作,协同寻优,有效克服了一些标准粒子群算法中存在的固有缺陷。

1 标准粒子群算法

设在 D 维搜索空间中,有一个包含 N 个粒子的粒子种群,其中将第 i 个粒子的位置定义为 $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{iD})$,将第 i 个粒子的速度定义为 $\mathbf{v}_i = (v_{i1}, v_{i2}, \dots, v_{iD})$,粒子本身迄今为止找到的个体最优解为 $\mathbf{pb}_i (i = 1, 2, \dots, N)$,整个种群迄今为止找到的全局最优解为 \mathbf{gb} 。每个粒子按照式(1)、式(2)更新自身的位置和速度:

$$\mathbf{v}_i(t+1) = \omega * \mathbf{v}_i(t) + c_1 * \text{rand}_1() * [\mathbf{pb}_i(t) - \mathbf{x}_i(t)] + c_2 * \text{rand}_2() * [\mathbf{gb}(t) - \mathbf{x}_i(t)] \quad (1)$$

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \quad (2)$$

$$v_{ij}(t) = \begin{cases} v_{\max}, & v_{ij}(t) > v_{\max} \\ v_{ij}(t), & -v_{\max} \leq v_{ij}(t) \leq v_{\max} \\ -v_{\max}, & v_{ij}(t) < -v_{\max} \end{cases} \quad (3)$$

其中: $\text{rand}_1()$ 和 $\text{rand}_2()$ 是在区间 $[0, 1]$ 内均匀分布的随机数; c_1 和 c_2 是学习因子,通常取 $c_1 = c_2 = 2$; ω 是惯性权重,决定了粒子对当前速度的继承程度,可以在 $0.1 \sim 0.9$ 取值。采用惯性权重线性递减策略, t 时刻的惯性权重可以表示为:

$$\omega(t) = (\omega_{\max} - \omega_{\min})(T_{\max} - t)/T_{\max} + \omega_{\min} \quad (4)$$

其中: T_{\max} 代表最大迭代次数; ω_{\max} 为初始惯性权重值; ω_{\min} 为最大迭代次数时的惯性权重值。惯性权重线性递减策略使得算法更为稳定和精确^[10]。算法采用了如式(3)的吸收墙策略, v_{\max} 为算法搜索边界,即对于第 i 个粒子的第 j 维 $v_{ij}(t)$,按式(3)对应条件进行更新,当 $-v_{\max} \leq v_{ij}(t) \leq v_{\max}$ 时,则按式(1)所得值进行更新。

2 多粒子角色协同作用的混合粒子群算法

2.1 粒子角色

标准粒子群算法容易陷入局部最优,这是由于随着算法的迭代进行,粒子多样性不断下降,粒子逐渐趋同,最后收敛于局部极值点。粒子的聚集现象造成探索能力的极大浪费,显然这是种群寻优的“盲目性”造成的。

通过对自然界中生物种群的观察发现,一个种群的繁荣稳定,在于种群各成员之间密切的分工协作,以蜂群为例,一个庞大的蜂群主要由3种角色组成:工蜂、雄蜂和蜂王,不同的子种群成员各司其职,使得整个种群能够健康发展。

将这种思想借鉴到粒子群算法中,本文将整个种群中的粒子划分为3类子群,即探索粒子(Exploring Particle, EP)、巡逻粒子(Patrolling Particle, PP)和局部开发粒子(Local Exploiting Particle, LEP),对3种粒子角色采用不同的寻优策略,并共享寻优信息,对整个解集进行协同开发。

2.1.1 探索粒子

EP的寻优策略与标准粒子群算法类似。粒子在算法运行过程中按式(1)、(2)进行迭代搜索。在算法陷入局部最优时,能自适应淘汰失活粒子,并引入具有较好适应度的外部粒

子。EP作为整个种群的主体部分,承担着对整个解空间的主要寻优任务。

2.1.2 巡逻粒子

针对粒子群算法易陷入局部最优的特点,引入基于混沌搜索的PP的概念。其主要作用有两个:1)通过混沌搜索强大的空间探索能力,对整个解空间进行搜索,有利于算法跳出局部最优;2)在算法陷入进化停滞时,将自身较优的搜索信息注入到EP中,以增加种群粒子多样性。

以逻辑自映射函数作为混沌模型,模型如下:

$$y_{n+1} = 1 - 2 * y_n^2; n = 0, 1, 2, \dots, -1 < y_n < 1 \quad (5)$$

由于混沌搜索具有随机性、遍历性和初值敏感性,利用这些性质可以优化搜索。首先产生一组随机初值 $\mathbf{Y}_0 = (y_{01}, y_{02}, \dots, y_{0D})$,每一维均是在区间 $(-1, 1)$ 内均匀分布且不为0的随机数,然后按照式(5)得到混沌序列 \mathbf{Y}_n ,再利用式(6),将这组值映射到输入变量的取值区间中:

$$x_{nj} = \frac{b_j - a_j}{2} y_{nj} + \frac{b_j + a_j}{2}; j = 1, 2, \dots, D \quad (6)$$

其中 a_j, b_j 为第 j 维输入变量的取值上下限。为了使混沌搜索更具有代表性,每个PP每次迭代均混沌搜索 M 次,取其中的最优位置作为本次迭代的更新位置,从而得到一个适应度较好的PP种群。

以全局最优粒子的进化停滞迭代数作为PP信息注入的触发条件。当全局最优粒子的进化停滞迭代数达到事先设定的阈值 K 时,表明种群失去了活力,此时将PP替代EP中若干适应度低的个体,能增加EP多样性,有利于算法跳出局部最优。

2.1.3 局部开发粒子

EP和PP在求解高维函数时局部开发能力不足,尤其在算法迭代后期,全局最优值可能就在最优粒子附近,但是由于种群陷入停滞状态或振荡状态而不能搜索到。在文献[11]的基础上,本文引入主要用于局部精细搜索的局部开发粒子,采用单维异步更新的邻域搜索策略。

记具有 N 个粒子的种群迄今为止搜索到的最优位置为 \mathbf{gb} ,LEP各维的邻域搜索公式为:

$$x_{ij}(t+1) = gb_j(t) + R_{ij}[x_{ij}(t) - gb_j(t)] \quad (7)$$

其中: $k \in \{1, 2, \dots, N\}$,且 $k \neq i$; R_{ij} 是一个在区间 $[0, 1]$ 内均匀分布的随机数,控制 $gb_j(t)$ 与种群某个粒子相应维度之间的搜索范围。随着种群收敛,邻域的范围会逐渐缩小。在每次迭代中,LEP有一个 D 次的内循环,以整个种群迄今为止搜索到的全局最优位置为起始点,采取单维异步更新策略,从第一维度开始,根据式(7)进行单维偏移搜索,每搜索一次要结合剩余还没有更新的维度位置计算出粒子的适应度值,采取试探机制,判断更新前后适应度值的大小,选取适应度值较优的位置作为该维的位置。LEP的更新公式如式(8)所示:

$$x_{ij}(t+1) = \begin{cases} x_{ij}(t+1), & f(\dots, x_{ij}(t+1), \dots) < f(\dots, gb_j(t), \dots) \\ gb_j(t), & f(\dots, x_{ij}(t+1), \dots) \geq f(\dots, gb_j(t), \dots) \end{cases} \quad (8)$$

LEP弥补了前两种粒子在高维函数中局部搜索能力不足的缺陷,在一定程度上提高了算法的收敛速度。

在MPRPSO中,3种粒子角色共享搜索信息。EP、PP和LEP所占比例分别为 ε, λ 和 θ ,其中 $\varepsilon + \lambda + \theta = 1$ 。由于不同的粒子角色所承担的搜索任务不同,为了获得最佳的寻优效率,需要对不同粒子角色所占比例作出平衡。

2.2 MPRPSO 算法基本流程

将3种依据不同搜索策略的粒子角色有机结合起来,其

基本算法流程如下。

步骤 1 初始化粒子群及相关参数,选取合适的粒子总数 N ,每个粒子的维度为 D ,其中算法最大迭代次数为 T_{\max} ,根据粒子的初始位置,确定每个粒子的个体最优位置和种群的全局最优位置。

步骤 2 将粒子分为 3 类粒子角色,依据粒子角色的不同采取如下相应的寻优策略:

1) 根据式(1)~(2)对每个 EP 的位置和速度进行更新;

2) 根据式(5)~(6),利用混沌变量做混沌搜索,对 PP 的位置进行更新;

3) 将种群迄今为止搜索到的全局最优位置作为 LEP 此次迭代的起始位置,根据式(7)作单维异步邻域搜索,根据式(8)更新粒子的各维的位置。

步骤 2 的伪代码描述如下:

```
for i = 1 to N
    % EP 更新策略
    if i <= N * ε then
        根据式(1)~(2)更新粒子速度和位置;
    end if
    % PP 更新策略
    if i > N * ε && i <= N * (ε + λ) then
        % 粒子每次迭代混沌搜索 M 次
        for time = 1 to M
            根据式(5)~(6)作混沌搜索;
            if f(xi(t)) < temp1(i) then
                将 xi(t) 作为粒子本次混沌搜索的最优位置;
                temp1(i) = f(xi(t));
            end if
        next time
    end if
    % LEP 更新策略
    if i > N * (1 - θ) then
        xi(t) = gb(t - 1);
        for j = 1 to D
            根据式(7)对粒子作单维异步搜索;
            根据式(8)更新 xij(t) 的位置;
        next j
    end if
next i
```

步骤 3 计算所有粒子的适应度,更新 EP 的个体最优值 pb_i 和整个种群的全局最优值 gb 。若 gb 没有被更新,则全局最优粒子的进化停滞迭代数 k 加 1;否则置 0。若 k 达到阈值 K ,则用 PP 替代掉 EP 中相应数量的适应度较低的个体。

步骤 4 如果满足算法终止条件,则输出最优解;否则迭代次数 t 加 1,转步骤 2。

3 仿真实验结果及分析

为了验证 MPRPSO 算法的有效性,本文选取了如表 1 中的 6 个在测试中经常使用到的 Benchmark 函数,并与标准粒子群优化(Standard Particle Swarm Optimization, SPSO)算法、高速收敛粒子群优化(PSO algorithm with High Speed Convergence, PSO-HSC)算法^[12]和鲑鱼效应粒子群优化(Catfish Particle Swarm Optimization, CatfishPSO)算法^[13]进行比较。PSO-HSC 具有收敛速度较快的特点,而 CatfishPSO 能比较有效地跳出局部最优,这两种算法都是具有典型特征的粒子群改进算法。

表 1 Benchmark 函数定义

函数名	函数定义	作用域	最小值
Sphere	$f_1 = \sum_{i=1}^n x_i^2$	$[-100, 100]^n$	0
Rosenbrock	$f_2 = \sum_{i=1}^n [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-30, 30]^n$	0
Rastrigrin	$f_3 = \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i) + 10]$	$[-5.12, 5.12]^n$	0
Griewank	$f_4 = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos(x_i / \sqrt{i}) + 1$	$[-600, 600]^n$	0
Ackley	$f_5 = 20 + e - 20e^{-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}} - \frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i)$	$[-32, 32]^n$	0
Quadric	$f_6 = \sum_{i=1}^n \left(\sum_{j=1}^i x_j \right)^2$	$[-100, 100]^n$	0

实验的参数设置如下:所有算法的粒子规模 $N = 50$; $\omega_{\max} = 0.8$, $\omega_{\min} = 0.2$; $c_1 = c_2 = 2$; 迭代次数 $T_{\max} = 500$; MPRPSO 算法中,各种粒子的比例为 $\varepsilon = 0.8$, $\lambda = \theta = 0.1$,混沌搜索次数 $M = 10$,最优粒子进化停滞迭代数阈值 $K = 5$; PSO-HSC 算法中,停滞判定代数 $MAX = 10$ 。实验中除 Sphere 函数取 30 维外,其余函数均为 50 维解空间,同时为减小偶然性,对每个函数分别运行 30 次并统计相关实验结果,实验结果如表 2 所示。为了更直观地比较各种算法的寻优性能,图 1 中给出了每种算法所对应的测试函数适应度进化曲线。

表 2 函数测试实验结果

函数	算法	最优值	最差值	平均值	标准差
f_1	SPSO	0.014	0.128	0.051	0.027
	PSO-HSC	3.480E-55	7.019E-51	8.468E-52	2.071E-51
	CatfishPSO	1.082E-49	9.353E-37	3.417E-38	1.681E-37
	MPRPSO	2.400E-77	2.909E-71	2.352E-72	6.140E-72
f_2	SPSO	1.458E+2	7.476E+2	3.731E+2	1.454E+2
	PSO-HSC	1.592E-26	2.422E-13	5.906E-14	9.385E-14
	CatfishPSO	6.163E-17	1.774E-5	3.553E-6	7.092E-6
	MPRPSO	0	1.122E-28	4.678E-29	3.375E-29
f_3	SPSO	48.519	1.343E+2	84.264	20.173
	PSO-HSC	3.304E-8	7.671E-2	9.000E-3	2.285E-2
	CatfishPSO	8.125E-8	1.990	0.805	0.736
	MPRPSO	0	0	0	0
f_4	SPSO	0.975	1.123	1.044	0.032
	PSO-HSC	5.268E-4	2.702E-3	1.082E-3	1.014E-3
	CatfishPSO	2.575E-3	6.826E-2	1.539E-2	2.447E-2
	MPRPSO	0	0	0	0
f_5	SPSO	3.349	6.276	4.673	0.723
	PSO-HSC	1.421E-14	3.197E-14	2.309E-14	4.832E-15
	CatfishPSO	4.974E-14	6.750E-14	5.860E-14	5.561E-15
	MPRPSO	5.684E-14	1.101E-13	7.780E-14	1.259E-14
f_6	SPSO	3.200E+2	8.816E+2	5.363E+2	1.326E+2
	PSO-HSC	4.104E-8	1.350E-6	3.573E-7	4.044E-7
	CatfishPSO	5.234E-7	1.809E-5	5.666E-6	5.430E-6
	MPRPSO	1.193E-15	2.388E-13	2.909E-14	4.294E-14

从表 2 可以看出,无论是从寻优的精度还是从算法的稳定性上看,MPRPSO 算法都取得了一定的提高。对于单峰函数

Sphere(f_1),新算法具有比其他3种算法更好的收敛速度和寻优精度,方差更小,表现出更好的鲁棒性;并且对于多峰函数,新算法也具有较好的求解精度和稳定性,在复杂多峰函数 Rastrigrin(f_3)和 Griewank(f_4)中均能收敛到最优解,虽然在 Ackley 函数(f_5)的求解精度上与其他两种改进算法相差无几,但也优于 SPSO 算法。这说明 MPRPSO 算法有较强的全局搜索能力,能够有效摆脱局部最优,而且提高了粒子群的收敛速度,具有一定的鲁棒性,在算法精度方面有一定的优势。

从图1中也可以直观地看出,MPRPSO 算法在绝大多数函数中具有更快的收敛速度和更精确的搜索精度,尤其是在 f_3 和 f_4 的结果中,粒子适应度值超出了最小精度值而显示为0,由于纵坐标是粒子适应度值的 \lg 对数,所以后期的曲线无法显示。引入基于混沌搜索的 PP 能够有效对算法陷入局部最优的情况进行规避,LEP 的精细搜索使得算法具有较强的局部开发能力和较快的收敛速度,多种粒子角色协同作用使得粒子的搜索效率更高。

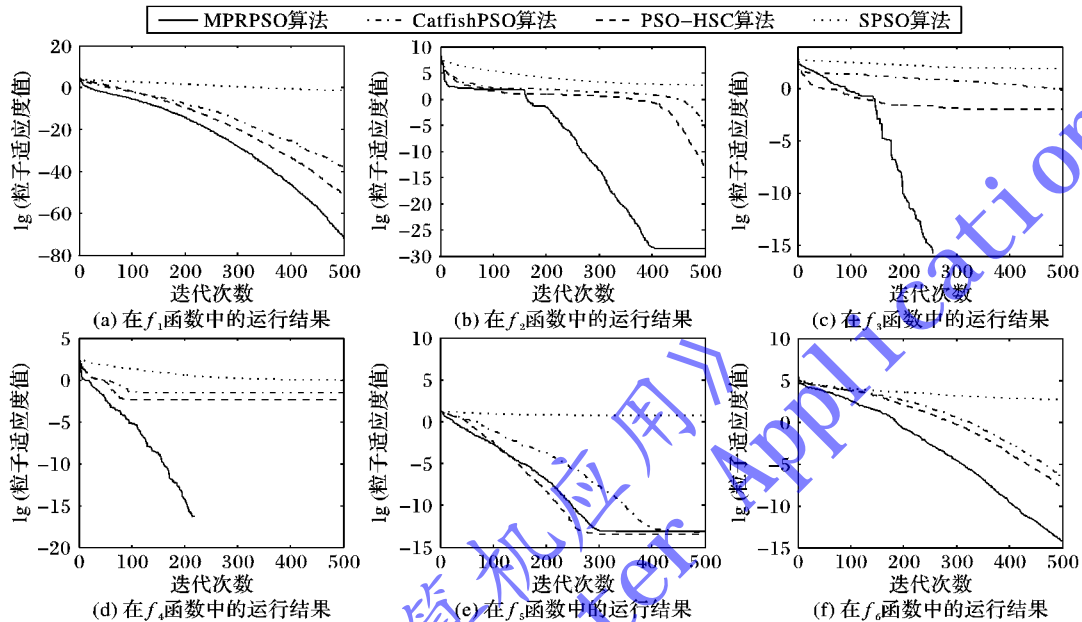


图1 6个测试函数的优化对比结果

下面讨论一下不同粒子角色对于算法性能的影响。选取了几组比较具有代表性的 ε 、 λ 和 θ 值,参数设置为解集维度 $D=100$,最大迭代次数 $T_{\max}=1000$,其余参数与前一实验一致。为了不失一般性,每次实验均运行30次。算法在不同的粒子成分下对测试函数的寻优结果如表3所示。

表3 不同粒子成分对应的寻优对比

ε	λ	θ	函数	最优值	最差值	平均值	平均收敛代数	平均运行时间/s
0.80	0.10	0.1	f_1	4.947E-151	2.870E-143	2.870E-144	1000	4.944
			f_3	0	0	0	296	2.144
0.90	0.1	0.1	f_1	2.216E-153	7.091E-146	1.375E-146	1000	4.192
			f_3	0	0.995	0.298	425	2.464
0.90	0.10	0.1	f_1	8.409	20.915	15.032	1000	1.634
			f_3	23.854	43.105	35.186	1000	2.571
1	0	0	f_1	7.902	20.423	13.076	1000	1.376
			f_3	1.063E+2	3.125E+2	1.930E+2	1000	2.024
0.70	0.10	0.2	f_1	2.053E-151	4.747E-147	6.231E-148	1000	8.668
			f_3	0	0	0	257	3.135
0.70	0.20	0.1	f_1	1.949E-144	6.877E-138	1.198E-138	1000	6.453
			f_3	0	0	0	293	2.459

表3中列举了比较典型的单峰函数 Sphere(f_1)和复杂多峰函数 Rastrigrin(f_3)的实验结果。兼具 EP、PP 和 LEP 的算法的结果相对于其他只含两种粒子的算法的寻优结果具有一定的优势。对于主要测试算法寻优精度的单峰函数 Sphere(f_1),LEP 使得算法的寻优精度远远高于没有 LEP 的算法,说

明 LEP 具有较强的局部搜索能力,有 PP 而没有 LEP 的算法结果与标准粒子群算法(即 $\varepsilon=1$)相差无几,这是由于 PP 侧重于全局寻优,而对局部收敛意义不大,不会对算法的局部收敛性能产生太大的影响。而对于具有大量局部极值的复杂多峰函数 Rastrigrin(f_3),PP 的引入一定程度上避免了算法的早熟收敛,使得算法的寻优精度相对于标准粒子群算法有了一定提高,LEP 则有效提高了算法的收敛速度,使算法能够收敛到最优解,但是只有 LEP 而没有 PP 的算法虽然有时也能收敛到最优,但往往会陷入局部最优,导致算法的稳定性不佳。3种不同角色的粒子协同作用,才能使算法获得最好的寻优效果。

另外,从表3中也可以看出,PP 的比重增大对于算法的性能没有太大提升,而 LEP 的比重增大虽然提高了收敛速度,但对精度提升不大。在运行时间方面,混合算法尤其是 LEP 的引入使得开销时间有所增加,但相对于获得的寻优精度,基本在可接受的范围之内。综合考虑算法时间开销和算法寻优性能,取 $\varepsilon=0.8$, $\lambda=\theta=0.1$ 是比较合适的。

4 结语

MPRPSO 算法针对标准粒子群算法迭代后期收敛速度慢、粒子同化现象严重并极易陷入局部最优的问题,引入了依据不同搜索策略的种群粒子角色协同作用的算法,将自然界中种群角色分工的思想借鉴到粒子群算法中,对不同的子种群采取相应的寻优策略,优化配置,提升粒子的寻优参与度,充分利用了种群的寻优能力,提高了粒子的搜索效率。仿真结果也表明所提算法具有有效性。将改进算法应用于实际工

程中,在实践中检验算法性能,将是下一步工作的重点。

参考文献:

- [1] SUN J, FANG W, WU X, *et al.* Quantum-behaved particle swarm optimization: theory and application [M]. Beijing: Tsinghua University Press, 2011: 6-7. (孙俊, 方伟, 吴小俊, 等. 量子行为粒子群优化: 原理及其应用[M]. 北京: 清华大学出版社, 2011: 6-7.)
 - [2] SUBRAMANIAN P, RAMKUMAR N, NARENDRANA T T, *et al.* A technical note on 'analysis of closed loop supply chain using genetic algorithm and particle swarm optimization' [J]. International Journal of Production Research, 2012, 50(2): 593-602.
 - [3] CHEN M, LIU Y. Improved particle swarm optimization based on adaptive rejection factor [J]. Journal of Computer Applications, 2013, 33(8): 2269-2272. (陈明, 刘衍民. 基于自适应排斥因子的改进粒子群算法[J]. 计算机应用, 2013, 33(8): 2269-2272.)
 - [4] NIE L, ZHANG T, GUO L. Hybrid particle swarm optimization algorithm based on parallel directional turbulence [J]. Application Research of Computers, 2013, 30(6): 1633-1635. (聂立新, 张天侠, 郭立新. 并行定向扰动的混合粒子群优化算法[J]. 计算机应用研究, 2013, 30(6): 1633-1635.)
 - [5] TAO X, LIU F, LIU Y, *et al.* Multi-scale cooperative mutation particle swarm optimization algorithm [J]. Journal of Software, 2012, 23(7): 1805-1815. (陶新民, 刘福荣, 刘玉, 等. 一种多尺度协同变异的粒子群优化算法[J]. 软件学报, 2012, 23(7): 1805-1815.)
 - [6] ZHONG Z, LI X, PANG S. Multi-objective immune particle swarm optimization algorithm with a hybrid global best selecting strategy [J]. Computer Engineering and Applications, 2013, 49(13): 43-47. (钟昭明, 李向阳, 逢珊. 混合择优的多目标免疫粒子群优化算法[J]. 计算机工程与应用, 2013, 49(13): 43-47.)
 - [7] HYUNWOOK B, TAE-HYOUNG K, JAENA R, *et al.* Diversity-enhanced particle swarm optimizer and its application to optimal flow control of sewer networks [C]// Proceedings of the 2013 Science and Information Conference. Piscataway: IEEE Press, 2013: 977-978.
 - [8] LI M, KANG H, ZHOU P, *et al.* Hybrid optimization algorithm based on chaos, cloud and particle swarm optimization algorithm [J]. Journal of Systems Engineering and Electronic, 2013, 24(2): 324-334.
 - [9] YAN Z, DENG C, ZHOU J, *et al.* A novel two-subpopulation particle swarm optimization [C]// Proceedings of the 10th World Congress on Intelligent Control and Automation. Piscataway: IEEE Press, 2012: 4113-4117.
 - [10] KOREL B. Automated software test data generation [J]. IEEE Transactions on Software Engineering, 1990, 16(8): 870-879.
 - [11] SHI X, SUN H, LI J, *et al.* Particle swarm optimization algorithm with fast convergence and adaptive escape [J]. Journal of Computer Applications, 2013, 33(5): 1308-1312. (史小露, 孙辉, 李俊, 等. 具有快速收敛和自适应逃逸功能的粒子群优化算法[J]. 计算机应用, 2013, 33(5): 1308-1312.)
 - [12] ZHU H, WU Y. A PSO algorithm with high speed convergence [J]. Control and Decision, 2010, 25(1): 20-30. (朱海梅, 吴永萍. 一种高速收敛粒子群优化算法[J]. 控制与决策, 2010, 25(1): 20-30.)
 - [13] CHUANG L, TSAI S, YANG C. Catfish particle swarm optimization [C]// Proceedings of the 2008 IEEE Swarm Intelligence Symposium. Piscataway: IEEE Press, 2008: 1-5.
-
- (上接第2294页)
- [12] GUO L, YANG Y. A classification algorithm of defect prediction for software modules based on fuzzy support vector machine [J]. Journal of Nanjing University: Natural Sciences, 2012, 48(2): 221-227. (郭丽娜, 杨杨. 一种基于模糊支持向量机软件模块缺陷检测算法[J]. 南京大学学报: 自然科学版, 2012, 48(2): 221-227.)
 - [13] LI Y, LIU Z, ZHANG H. Review on ensemble algorithms for imbalanced data classification [J]. Application Research of Computers, 2014, 31(5): 1287-1291. (李勇, 刘战东, 张海军. 不平衡数据的集成分类算法综述[J]. 计算机应用研究, 2014, 31(5): 1287-1291.)
 - [14] LÓPEZ V, FERNÁNDEZ A, MORENO-TORRES J G, *et al.* Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. Open problems on intrinsic data characteristics [J]. Expert Systems with Applications, 2012, 39(7): 6585-6608.
 - [15] GARCÍA S, HERRERA F. Evolutionary undersampling for classification with imbalanced datasets: proposals and taxonomy [J]. Evolutionary Computation, 2009, 17(3): 275-306.
 - [16] MENZIES T, TURHAN B, BENER A, *et al.* Implications of ceiling effects in defect predictors [C]// Proceedings of the 4th International Workshop on Predictor Models in Software Engineering. New York: ACM Press, 2008: 47-54.
 - [17] PELAYO L, DICK S. Evaluating stratification alternatives to improve software defect prediction [J]. IEEE Transactions on Reliability, 2012, 61(2): 516-525.
 - [18] RIQUELME J C, RUIZ R, RODRIGUEZ D, *et al.* Finding defective modules from highly unbalanced datasets [J]. Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos, 2008, 2(1): 67-74.
 - [19] PELAYO L, DICK S. Applying novel resampling strategies to software defect prediction [C]// NAFIPS '07: Annual Meeting of the North American Fuzzy Information Processing Society. Piscataway: IEEE Press, 2007: 69-72.
 - [20] ZHENG J. Cost-sensitive boosting neural networks for software defect prediction [J]. Expert Systems with Applications, 2010, 37(6): 4537-4543.
 - [21] QUINLAN J R. C4.5: programs for machine learning [M]. San Francisco: Morgan Kaufmann Publishers, 1993.
 - [22] ZHOU Z. Ensemble methods: foundations and algorithms [M]. Boca Raton: CRC Press, 2012: 15-20.
 - [23] SCHAPIRE R. The strength of weak learnability [J]. Machine Learning, 1990, 5(2): 197-227.
 - [24] FREUND Y, SCHAPIRE R E. A decision-theoretic generalization of on-line learning and an application to boosting [C]// Proceedings of the Second European Conference on Computational Learning Theory. Berlin: Springer, 1995: 23-37.
 - [25] BREIMAN L. Bagging predictors [J]. Machine Learning, 1996, 24(2): 123-140.
 - [26] SONG Q B, JIA Z H, SHEPPERD M, *et al.* A general software defect-proneness prediction framework [J]. IEEE Transactions on Software Engineering, 2011, 37(3): 356-370.
 - [27] SHULL F, BASILI V, BOEHM B, *et al.* What we have learned about fighting defects [C]// Proceedings of the 8th International Symposium on Software Metrics. Washington, DC: IEEE Computer Society, 2002: 249-258.