文章编号: 1001-0920 (2015) 10-1779-06 DOI: 10.13195/j.kzyjc.2014.1111

一种采用抽样策略的PSO算法

姜建国, 叶 华, 马亚华

(西安电子科技大学 计算机学院，西安 710071)

摘 要: 原始粒子群优化算法 (PSO) 和各种改进方法存在着参数取值固定、收敛精度低等问题. 为此, 提出一种采用抽样策略的粒子群优化算法 (SS-PSO). 通过拉丁超立方抽样 (LHS) 策略更新粒子速度和位置, 以加快收敛速度; 提出一种基于随机采样的最优位置修正方法, 以微调全局最优; 提出“双抽样”LHS 局部搜索方法, 以提高收敛精度. 与其他新近提出的两个算法进行对比, 结果显示 SS-PSO 在一定程度上提高了算法的性能.

关键词: 粒子群优化算法；抽样策略；局部搜索；全局优化

中图分类号: TP306.1 **文献标志码:** A

Particle swarm optimization algorithm via sampling strategy

JIANG Jian-guo, *YE Hua*, *MA Ya-hua*

(School of Computer Science and Technology，Xidian University，Xi'an 710071，China．Correspondent：JIANG Jian-guo，E-mail：jgjiang@mail.xidian.edu.cn)

Abstract: There're some issues such as fixed parameters' value and easy to fall into local optimum in classical PSO algorithm and many improvements. Therefore, an improved PSO algorithm via sampling strategy(SS-PSO) is proposed. Firstly, replacement of the particles' speed and location via Latin hypercube sampling(LHS) is proposed for speeding up the convergence process. Then, correction of the global best location via random sampling is proposed for fine tuning the global best location. Finally, "double sampling" LHS is proposed for local search to improve the convergence precision. Two new improvements are used to compare with the SS-PSO algorithm. The results show that the SS-PSO algorithm can improve the PSO's algorithm performance in some extent.

Keywords: particle swarm optimization；sampling strategy；local search；global optimization

0 引 言

近年来, 学者们提出了一系列新颖的群智能优化算法. 其中, 粒子群优化算法[1]以其流程简单、易于实现而得到较多青睐, 并被推广到各个应用领域[2-7]. 然而, 粒子群算法容易早熟收敛、收敛精度低等缺陷使其应用受到了限制. 对此, 学者们从不同角度提出了多种改进方法[8-15]. 文献 [8] 研究了粒子群进化公式中随机性的相关问题, 并提出了基于与全局最优位置距离的粒子群算法; 文献 [9] 研究了算法的速度与惯性权重的关系, 并提出了根据速度信息调整参数的粒子群算法; 文献 [10] 提出了一种加快粒子群收敛速度的向心加速粒子群算法; 文献 [11] 提出了综合学习粒子群算法, 利用一种新的粒子综合学习方式改善了算法的性能; 文献 [12] 提出了一种基于振荡权重和加速因子的粒子群算法; 文献 [13] 提出了采用加权粒子的 PSO 算法; 文献 [14] 提出了一种基于中断因子的粒子群算法, 用于使算法尽快跳出局部最优; 文献 [15] 将梯度信息和吸引-排斥机制融合, 提出了基于多样性导向的 PSO 算法. 以上算法虽然在一定程度上改善了原始算法的性能, 但是在收敛精度和稳定性上仍然存在欠缺.

本文提出一种采用抽样策略的粒子群优化算法 (SS-PSO). 针对基本算法收敛较慢、参数取值固定的问题, 提出基于 LHS[16]的粒子位置和速度更新方式, 从多组中间粒子中抽取位置和速度, 克服参数取值固定的影响, 提高收敛速度; 针对全局最优位置容易停滞的问题, 提出基于随机采样的最优位置修正方法, 及时发现更优的空间位置; 针对易于陷入局部最优的问题, 使用“双抽样”方法进行 LHS 局部搜索, 提高收敛精度. 最后针对几个常用的测试函数进行了仿

收稿日期: 2014-07-15；**修回日期:** 2014-10-21.

基金项目: 国防基础科研计划项目(A1120132007).

作者简介: 姜建国(1956–), 男, 教授, 从事图形图像处理、优化理论与算法等研究；叶华(1989–), 男, 硕士生, 从事群智能优化的研究.

真测试. 测试结果表明, 相比于经典PSO算法, 经过改进的新算法的性能有一定的提高. 基本粒子群算法和拉丁超立方抽样算法的步骤详见文献[1]和文献[16], 文中不再赘述.

1 改进算法

1.1 基于LHS抽样策略和中间粒子选择的粒子更新方式

在PSO中, 参数ω、c_1和c_2如何变化往往通过经验值或实验指定[17-18], 缺乏灵活性. 有学者提出利用分层算法改进PSO[19], 取得了良好的效果. 然而, 由于群体较多, 较大地增加了计算量. 对此, 文中提出一种新思路, 即通过LHS抽样产生多个粒子的速度和位置, 再通过一定的规则, 从这些速度和位置向量中选择恰当的元素作为下一轮迭代中粒子的速度和位置. 基本步骤如图1所示.

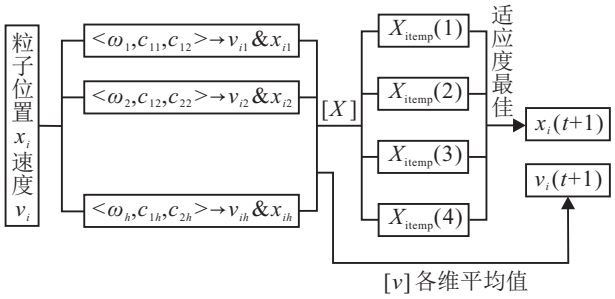

图1 基于LHS和中间粒子选择的粒子更新过程

算法详细步骤如下.

Step 1 确定粒子更新过程中ω、c_1和c_2的抽样区间以及抽样规模h.

Step 2 对ω、c_1和c_2进行LHS抽样, 产生h组样本. 对于第i个粒子, 每一组样本按照下式产生速度v_{ik}和位置x_{ik} $(k=1,2,\cdots,h)$:

$$v_{ik}=\omega_k v_i+c_{1k}r_1(p_i-x_i)+c_{2k}r_2(p_g-x_i), \quad (1)$$

$$x_{ik}=x_i+v_{ik}. \quad (2)$$

Step 3 对于产生的h组位置向量, 可以构成大小为$h\times D$的矩阵X, 矩阵的每一行代表一个粒子的位置. 从矩阵X中按照下列各式提取出4个中间粒子.

1) 中间粒子1 $(X_{\text{itemp}}(1))$. 该粒子的每一维等于X对应维的平均值, 即

$$X_{\text{itemp}}(1)_j=\frac{\sum_{m=1}^{h}X_{m,j}}{h},\ j=1,2,\cdots,D. \quad (3)$$

2) 中间粒子2 $(X_{\text{itemp}}(2))$. 该粒子的每一维等于X对应维的中值, 即

$$X_{\text{itemp}}(2)_j=\text{median}(X(:,j)),\ j=1,2,\cdots,D. \quad (4)$$

3) 中间粒子3 $(X_{\text{itemp}}(3))$. 该粒子的每一维等于X对应维最大值和最小值中绝对值较大者, 即

$$X_{\text{itemp}}(3)_j=\begin{cases}\min(X(:,j)), & |\min(X(:,j))|>|\max(X(:,j))|;\\ \max(X(:,j)), & \text{else.}\end{cases}$$
$$j=1,2,\cdots,D. \quad (5)$$

4) 中间粒子4 $(X_{\text{itemp}}(4))$. 该粒子的每一维等于X对应维最大值和最小值中绝对值较小者, 即

$$X_{\text{itemp}}(4)_j=\begin{cases}\min(X(:,j)), & |\min(X(:,j))|<|\max(X(:,j))|;\\ \max(X(:,j)), & \text{else.}\end{cases}$$
$$j=1,2,\cdots,D. \quad (6)$$

Step 4 计算式(3)～(6)所表示的4个中间粒子的适应度值, 取其中适应度值最佳(对于求最小值问题, 即适应度值最小)的中间粒子的位置作为下一轮迭代的第i个粒子的位置.

Step 5 速度的更新. 在Step 2所述的过程中, 也产生了大小为$h\times D$的速度矩阵, 记为V, 其每一行代表一个速度向量. 为了反映粒子运动的平均趋势, 设置第i个粒子在下一轮迭代中的速度v_i为V中每一维的平均值.

$$V(t+1)_j=\frac{\sum_{m=1}^{h}V_{m,j}}{h},\ j=1,2,\cdots,D. \quad (7)$$

在基本粒子群算法及各改进方式中, 各个参数的取值都是由某个模型指定的. 这样, 在下一次迭代中, 各个参数的值唯一并且确定. 这种方式是现在通用的方式, 不过由于参数是确定的, 粒子下一次迭代中的位置和速度也随之确定, 粒子没有其他选择, 缺乏灵活性. 而在采用LHS的抽样方法中, 通过抽样, 可以取得多组参数, 得到多个中间粒子, 可以选择其中最优的粒子进入下一次迭代, 并且下一代的种群可以体现出群体进化的趋势, 有利于提升个体在进化中的灵活性和种群的收敛速度.

表1统计了在不同问题中, 4个中间粒子被取到作为下一轮迭代粒子位置的比例, 展示了该方法对不同性质问题的灵活适应性, 其中每个测试函数独立运行10次, 维数为30维.

表1 各中间粒子被取到的比例统计$(30D)$ %

函数名	粒子1	粒子2	粒子3	粒子4
Sphere	2.58	0.66	0	96.78
Salomon	27.49	25.87	9.26	37.38
Schwefel	19.95	27.04	32.95	32.06

1.2 基于随机采样的最优位置修正方法

为了进一步提高收敛精度, 及时发现群体中可使适应度更优的位置, 本节提出基于随机采样的全局最优位置修正方法, 过程如下所示:

for $j = 1, 2, \cdots, D$

$t_1 = t_2 = p_g$;

$\mathrm{avg}(p_g) = \dfrac{\min(p_g) + \max(p_g)}{2}$;

$t_1(j) = \min(p_g) + \mathrm{rand}() \times (\mathrm{avg}(p_g) - \min(p_g))$;

$t_2(j) = \mathrm{avg}(p_g) + \mathrm{rand}() \times (\max(p_g) - \mathrm{avg}(p_g))$;

$$p_g(j) = \begin{cases} t_1(j), \text{ if } f(t_1) < f(p_g) \text{ and } f(t_1) < f(t_2); \\ t_2(j), \text{ if } f(t_2) < f(p_g) \text{ and } f(t_2) \leqslant f(t_1); \\ p_g(j), \text{ else} \end{cases}$$

endfor

其中: $\min(p_g)$ 表示全局最优位置 p_g 中所有元素的最小值, $\max(p_g)$ 表示全局最优位置 p_g 中所有元素的最大值, $\mathrm{avg}(p_g)$ 表示 $\min(p_g)$ 和 $\max(p_g)$ 的平均值. 该过程在每次迭代结束之后执行一次, p_g 从第 1 维到第 D 维更新一遍, 该过程有较大的可能搜索到搜索空间中使粒子适应度更优的位置, 从而提高收敛精度.

1.3 基于 LHS 的双抽样局部搜索

在现有的改进算法中, 有些采用了局部搜索措施, 且得到了较为优良的效果[20-21]. 然而, 在传统的局部搜索中, 往往存在搜索范围小、搜索随机性大, 以及所谓的“两步前进, 一步后退”效应[22]等问题. 因此, 本文提出基于 LHS 的双抽样局部搜索方法. 该方法对全局最优位置 p_g 的每一维进行两次 LHS 抽样. 第 1 次是粗抽样, 在较大范围内抽样, 找到使函数值变优的粗略位置; 第 2 次是细抽样, 在更小范围内进行抽样, 找到使函数值更优的精细位置. 第 1 次搜索区间设定为

$$\begin{cases} d_{\mathrm{fmin}}(i) = \max\left(p_g(i) \times \left(1 - \dfrac{S \times \mathrm{rand}}{A}\right), x_{\min}\right), \\ d_{\mathrm{fmax}}(i) = \min\left(p_g(i) \times \left(1 + \dfrac{S \times \mathrm{rand}}{A}\right), x_{\max}\right). \end{cases} \tag{8}$$

其中: $d_{\mathrm{fmin}}(i)$ 表示第 i 维的搜索下界; $d_{\mathrm{fmax}}(i)$ 表示第 i 维的搜索上界; $p_g(i)$ 表示全局最优位置的第 i 维; rand 表示 (0,1) 范围内的随机数; $S = \mathrm{sign}(p_g(i))$, 即符号函数; A 表示一个常量, 为避免搜索范围过大, 此处设置 $A = 10$. 第 1 次抽样的规模为 h_1.

第 2 次搜索区间为

$$\begin{cases} d_{\mathrm{smin}}(i) = \max(p_g(i) \times (1 - S \times \alpha), d_{\mathrm{fmin}}(i)), \\ d_{\mathrm{smax}}(i) = \min(p_g(i) \times (1 + S \times \alpha), d_{\mathrm{fmax}}(i)). \end{cases} \tag{9}$$

其中: $d_{\mathrm{smin}}(i)$ 表示第 i 维的搜索下界; $d_{\mathrm{smax}}(i)$ 表示第 i 维的搜索上界; α 表示缩放因子; $S = \mathrm{sign}(p_g(i))$. 由式 (9) 可知, 第 2 次的搜索区间不大于第 1 次. 第 2 次抽样的规模为 h_2.

对于第 i 维 ($i = 1, 2, \cdots, D$), 先后进行两次抽样, 每次抽样结束后, 取使适应度最佳的值作为第 i 维的值. 若当前函数评价次数大于最大函数评价次数, 则搜索停止.

1.4 算法流程描述

Step 1: 初始化相关参数, 在搜索范围内随机产生初始种群和初始速度.

Step 2: 执行如 1.1 节所述的粒子更新过程, 所有粒子更新一次后, 迭代次数加一, 执行如 1.2 节所述的全局最优粒子位置修正过程. 若在此过程中粒子速度或者位置中某一维超出搜索范围, 则在搜索范围中随机产生一个值作为该维的值; 若在此过程中函数评价次数超过 I, 则转入 Step 4.

Step 3: 如果迭代次数大于 $T_{\max}$, 则停止迭代, 执行如 1.3 节所述的 LHS 双抽样局部搜索; 否则转入 Step 2.

Step 4: 算法停止, 输出结果.

2 实验与分析

为了验证新算法的有效性, 本节以 Matlab 为仿真工具, 进行多个仿真实验. 实验采用较常用的 7 个测试函数, 分别是 sphere、griewank、ackley、rastrigin、schwefel、rosenbrock 和 schaffer's f7. 其函数表达式、搜索范围见文献 [8] 和文献 [23]. 实验的参数设置如下:

1) 测试维数 $D = 30$, 种群规模 $n = 20$, 最大迭代次数 $T_{\max} = 100$.

2) 为保证双抽样局部搜索被至少执行一次, 根据文中算法的特点, 最大函数评价次数 I 应大于 $(5 \times n + 2 \times D) \times T_{\max} = 16\,000$, 因此可设置最大函数评价次数 $I = 30\,000$.

3) $[x_{\min}, x_{\max}]$ 由测试函数确定. $\omega_{\min} = 0.4, \omega_{\max} = 0.95, c_{1\,\min} = 1.25, c_{1\,\max} = 2.75, c_{2\,\min} = 0.5, c_{2\,\max} = 2.25$. 根据精细搜索的特点, 拟设定 1.3 节中缩放因子 $\alpha = 0.1$, 其他参数 (h、h_1、h_2) 由实验设定.

2.1 参数设置实验

本节将对如下 3 个参数的设置进行实验: 1.1 节中的抽样规模参数 h, 1.3 节中的抽样规模参数 h_1 和 h_2. 这 3 个参数对改进算法的性能有较大影响, 为了合理地选择参数, 文中对不同参数、不同水平进行了对比实验, 并以此作为参数设计的依据.

在 1.1 节中, 抽样规模参数 h 用来产生 h 组参数以及相应的粒子, 并由这些粒子生成 4 个中间粒子. 针对 h 的不同取值, 文中进行了实验. 算法为基本算法加上 1.1 节的改进, 不加入其他改进. 因此, h_1 和 h_2 无需设置, 其他参数如前文所述. 每个函数每个参数测 10 次, 实验结果如表 2 所示.

表 **2** 1.1 节抽样规模参数 h 的取值测试结果

取值	函数名	最优值	平均值	用时 / s
h=5	sphere	1.249 0e-20	1.328 1e-18	0.43
	ackley	1.338 9e-11	9.187 7e-11	0.57
h=10	sphere	2.184 1e-18	1.132 1e-15	0.59
	ackley	2.793 4e-10	7.162 8e-10	0.72
h=20	sphere	9.944 1e-17	1.548 7e-15	0.78
	ackley	8.483 7e-10	4.672 2e-09	0.94
h=30	sphere	8.691 0e-16	1.492 4e-13	0.97
	ackley	4.863 4e-09	1.829 7e-08	1.14

由表 2 可知, 随着 h 的增大, 大部分函数的寻优结果反而变差, 取较小的 h 不仅可以减少计算量, 也可以提高算法的性能. 因此, 下文中的 h 取值为 5.

在 1.3 节中, h_1 和 h_2 这两个参数用来调节 LHS 抽样时的网格密度. 针对 h_1 和 h_2 的不同取值, 文中进行了实验. 每个函数每组参数测 10 次, 实验结果如表 3 所示.

表 **3** 1.3 节抽样规模参数 h_1 和 h_2 的取值测试结果

取值	函数名	最优值	平均值	用时 / s
h_1=10	rosenbrock	15.197 0	51.412 1	0.61
h_2=10	schaffer's f7	1.969 8e-10	4.514 7e-09	1.77
h_1=20	rosenbrock	0.098 1	31.014 7	0.59
h_2=20	schaffer's f7	3.852 3e-10	1.034 0e-08	1.73
h_1=30	rosenbrock	0.337 3	34.180 0	0.58
h_2=30	schaffer's f7	2.737 9e-10	1.716 6e-08	1.72
h_1=50	rosenbrock	0.007 2	31.974 1	0.57
h_2=50	schaffer's f7	3.789 9e-10	1.218 1e-08	1.70

由表 3 可知, 无论 h_1 和 h_2 取何值, 结果的精度、用时相差均不大. 总体来说, 当 h_1 和 h_2 的取值较大时, 结果较好、用时更少, 因此, 可取 $h_1 = h_2 = 50$.

2.2 运行时间分析

原始 PSO 算法的时间复杂度为 $O(n \times T_{\max})$, 其中: n 为种群规模, $T_{\max}$ 为最大迭代次数. 综合分析文中提出的改进方法, 其时间复杂度为

$$O(n \times T_{\max} \times D) + O(k \times (h_1 + h_2) \times D).$$

其中 k 表示局部搜索被执行的次数, 对于不同函数, k 的值一般不同.

由上面的分析可知: 对于不同的函数, 文中算法执行局部搜索的次数一般不同, 难以给出准确的时间复杂度评价; 对于不同的算法, 每一轮迭代中的函数评价次数也都不一致, 而算法的时间主要花费在适应度函数的评价上. 因此, 下面的实验中, 所有算法都在相同的最大函数评价次数 I 下比较. 这样, 算法的时间复杂度均为 $O(I)$, 在这样的条件下比较更加科学和合理.

为了说明改进算法的时间性能, 将改进算法的效果和运行时间与基本 PSO 算法进行对比. 最大函数评价次数 I 如前所述, 为 30 000, 每个函数测试 10 次. 实验结果如表 4 所示, 其中用时指每次计算的平均用时.

表 **4** 运行时间和运行效果的对比测试结果

函数名	算法	最优值	平均值	用时 / s
sphere	PSO	159.015 9	5.406 7e+03	0.32
	SS-PSO	1.270 0e-54	1.434 3e-51	0.55
rosenbrock	PSO	28.832 1	6.934 6e+03	0.34
	SS-PSO	0.200 8	49.062 5	0.60

由表 4 可知, 在相同函数评价次数下, 新算法的平均用时普遍较原始算法有所增加, 不过平均用时均在可接受范围, 并且寻优精度相较原始算法提高较多.

2.3 改进措施有效性测试

为了验证 SS-PSO 中各改进措施的有效性, 本节分别测试了基本 PSO 算法 (算法 1)、在其上加入 1.1 节改进 (算法 2)、同时加入 1.1 节改进和 1.2 节改进 (算法 3) 以及最终算法, 每个函数均为测试 30 次后取其均值, 实验结果如表 5 所示. 由表 5 中可知, 随着改进措施的依次添加, 算法的收敛精度逐步提升, 也说明了各改进措施的有效性.

表 **5** 各改进措施的有效性测试结果

函数名	算法 1	算法 2	算法 3	SS-PSO
sphere	4 650.2	3.461 8e-19	3.405 4e-51	7.936 2e-52
rastrigin	117.271 3	8.385 2	5.283 1e-17	4.027 5e-17
schwefel	−6 710.1	−9 032.9	−12 004.0	−12 561.4
rosenbrock	5.826 8e+05	38.640 3	39.892 5	32.185 8

2.4 与同类算法对比测试

本节选取最近公开发表的两种效果较好的改进粒子群算法与本文进行对比实验. 参数设置如下: $n = 20$, $D = 30$, $I = 30\,000$, 每个函数独立测试 50 次. 所有算法的最大函数评价次数一致, 对比算法[8-9]按照其算法思想以及如上所述参数设置进行仿真. 结果见表 6, 其中: 函数 F1 ∼ F7 分别表示 sphere、griewank、ackley、rastrigin、schwefel、rosenbrock、schaffer's f7. 算法 A1 ∼ A4 分别表示 PSO、PSODDS[8]、APSO-VI[9]、SS-PSO.

由表 6 可知, 虽然在本文算法的运行结果中, 个别函数的最优值、最差值和方差劣于对比算法, 不过对于所有测试函数的平均值, 本文算法均优于对比算法, 并且对于大多数函数, 本文算法的最优值和标准差更好, 从而验证了本文算法在收敛精度和稳定性上较对比算法均有优势, 本文算法是有效的.

为了验证新算法在收敛速度上的优势, 图 2 和图 3 给出了 griewank 和 rosenbrock 函数的进化曲线.

从图 2 和图 3 可以看出, 在相同迭代次数下, 本文算法 (SS-PSO) 下降更快, 收敛效果更好, 取得了较其他算法更佳的收敛速度. 综上所述, 本文算法表现出了良好的优化性能.

表 **6**　与其他算法的对比测试结果

函数	算法	最优值	最劣值	平均值	标准差
F1	A1	1.283 3e+02	2.016 4e+04	5.853 3e+03	5.48e+03
	A2	1.645 2e-39	2.843 1e-37	2.321 0e-38	4.19e-38
	A3	0	10.081 3	1.204 9	3.235 2
	A4	1.684 2e-56	1.679 9e-51	5.765 0e-53	2.56e-52
F2	A1	1.846 7	199.304 1	52.095 5	47.293 3
	A2	0	0.046 5	0.008 7	0.011 0
	A3	0	9.067 6	0.772 3	2.462 6
	A4	0	0.041 7	0.003 4	0.008 0
F3	A1	2.784 8	17.911 1	13.041 7	4.544 4
	A2	1.400 9e-10	2.316 8	0.921 7	0.814 8
	A3	8.881 8e-16	9.969 6	1.027 8	0.223 6
	A4	4.440 9e-15	4.879 8e-15	4.770 3e-15	3.07e-16
F4	A1	29.275 2	234.273 9	124.724 2	46.136 2
	A2	24.874 0	100.490 6	49.968 2	17.403 1
	A3	1.088 6	87.891 2	54.155 3	41.074 0
	A4	4.738 1e-17	7.833 0e-15	6.823 2e-16	7.63e-16
F5	A1	−8 552.3	−4 799.8	−6 406.1	874.4
	A2	−9 035.6	−7 006.1	−8 190.0	1 557.9
	A3	−10 203.3	−5 638.9	−8 115.6	1 059.3
	A4	−12 568.1	−12 549.9	−12 560.3	9.528 3
F6	A1	4 899.1	3.847 4e+06	8.146 0e+05	8.17e+05
	A2	2.100 8	98.919 1	33.359 9	26.900 4
	A3	28.793 1	120.174 6	58.748 4	44.708 8
	A4	0.166 9	83.734 3	31.229 6	23.502 1
F7	A1	20.096 1	189.886 4	133.699 1	31.586 9
	A2	4.360 8	12.375 9	9.133 1	4.947 5
	A3	5.617 8	16.761 4	10.844 7	9.505 9
	A4	1.466 1e-10	3.254 7e-08	3.941 2e-09	5.65e-09

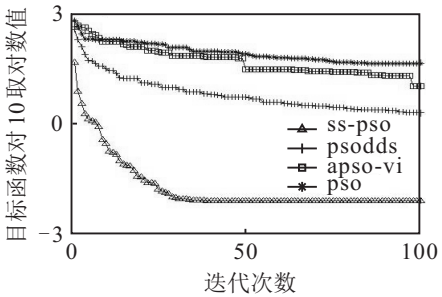

图 2　griewank 函数进化曲线

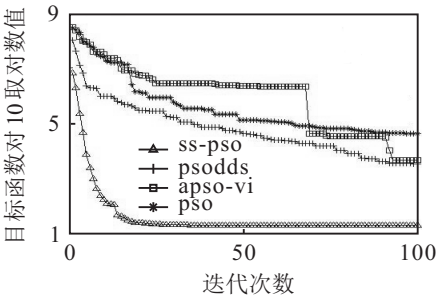

图 3　rosenbrock 函数进化曲线

3　结　论

本文提出了一种采用抽样策略的粒子群优化算法(SS-PSO). 通过采用LHS方法更新粒子的位置和速度, 采用随机采样方法优化全局最优位置, 基于LHS双抽样方法进行局部搜索等措施改进了原始粒子群算法. 进行了仿真实验, 并将结果与其他算法进行了对比. 本研究结果将LHS等抽样方法与PSO融合, 提出了独特的粒子更新方式和局部搜索方法, 并显示了新算法相对于经典算法具有更好的收敛性能. 由于LHS抽样的随机性, 导致新算法的收敛精度受到了限制, 并且计算量较大, 这些都是今后需要改进的方向.

参考文献(**References**)

[1] Kennedy J, Eberhart R. Particle swarm optimization[C]. Proc IEEE Int Conf on Neural Networks. Perth: Perth IEEE Press, 1995: 1942-1948.

[2] Qu B Y, Suganthan P N, Das S. A distance-based locally informed particle swarm model for multimodal optimization[J]. IEEE Trans on Evolutionary Computation, 2013, 17(3): 387-402.

[3] 温涛, 盛国军, 郭权, 等. 基于改进粒子群算法的Web服务组合[J]. 计算机学报, 2013, 36(5)：1031-1046.
(Wen T, Sheng G J, Guo Q, et al. Web service composition based on modified particle swarm optimization[J]. Chinese J of Computers, 2013, 36(5): 1031-1046.)

[4] 胥小波, 蒋琴琴, 郑康锋, 等. 基于混沌粒子群的IDS告警聚类算法[J]. 通信学报, 2013, 34(3): 105-110.
(Xu X B, Jiang Q Q, Zheng K F, et al. IDS alert clustering algorithm based on chaotic particle swarm optimization[J]. J on Communications, 2013, 34(3): 105-110.)

[5] Sadeghi J, Sadeghi S, Niaki S T A. Optimizing a hybrid vendor-managed inventory and transportation problem with fuzzy demand: An improved particle swarm optimization algorithm[J]. Information Sciences, 2014, 272: 126-144.

[6] Nedic V, Cvetanovic S, Despotovic D, et al. Data mining with various optimization methods[J]. Expert Systems with Applications, 2014, 41(8): 3993-3999.

[7] Feng H M, Liao K L. Hybrid evolutionary fuzzy learning scheme in the applications of traveling salesman problems[J]. Information Sciences, 2014, 270: 204-225.

[8] Jin X, Liang Y Q, Tian D P, et al. Particle swarm optimization using dimension selection methods[J]. Applied Mathematics and Computation, 2013, 219(10): 5185-5197.

[9] Xu G. An adaptive parameter tuning of particle swarm optimization algorithm[J]. Applied Mathematics and Computation, 2013, 219(9): 4560-4569.

[10] Beheshti Z, Hj Shamsuddin S M. CAPSO: Centripetal accelerated particle swarm optimization[J]. Information Sciences, 2014, 258: 54-79.

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

[12] 朱喜华, 李颖晖, 李宁, 等. 基于群体早熟程度和非线性周期振荡策略的改进粒子群算法[J]. 通信学报, 2014, 35(2): 182-189.
(Zhu X H, Li Y H, Li N, et al. Improved PSO algorithm based on swarm prematurely degree and nonlinear periodic oscillating strategy[J]. J on Communications, 2014, 35(2): 182-189.)

[13] Li N J, Wang W J, James Hsu C C, et al. Enhanced particle swarm optimizer incorporating a weighted particle[J]. Neurocomputing, 2014, 124: 218-227.

[14] Liu H, Ding G, Wang B. Bare-bones particle swarm optimization with disruption operator[J]. Applied Mathematics and Computation, 2014, 238: 106-122.

[15] Han F, Liu Q. A diversity-guided hybrid particle swarm optimization based on gradient search[J]. Neurocomputing, 2014, 137: 234-240.

[16] McKay M D, Beckman R J, Conover W J. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code[J]. Technometrics, 1979, 21(2): 239-245.

[17] 刘杨, 田学锋, 詹志辉, 等. 粒子群优化算法惯量权重控制方法的研究[J]. 南京大学学报: 自然科学版, 2011, 47(4): 364-371.
(Liu Y, Tian X F, Zhan Z H, et al. Research on inertia weight control approaches in particle swarm optimization[J]. J of Nanjing University: Natural Sciences, 2011, 47(4): 364-371.)

[18] 张治俊, 罗辞勇, 张帆, 等. 采用振荡参数策略的粒子群优化算法[J]. 重庆大学学报, 2011, 34(6): 36-41.
(Zhang Z J, Luo C Y, Zhang F, et al. Particle swarm optimization with oscillating parameter strategy[J]. J of Chongqing University, 2011, 34(6): 36-41.)

[19] 金敏, 鲁华祥. 一种遗传算法与粒子群优化的多子群分层混合算法[J]. 控制理论与应用, 2013, 30(10): 1231-1238.
(Jin M, Lu H X. A multi-subgroup hierarchical hybrid of genetic algorithm and particle swarm optimization[J]. Control Theory & Applications, 2013, 30(10): 1231-1238.)

[20] Qu B Y, Liang J J, Suganthan P N. Niching particle swarm optimization with local search for multi-modal optimization[J]. Information Sciences, 2012, 197: 131-143.

[21] Li M, Kang H, Zhou P. Hybrid optimization algorithm based on chaos, cloud and particle swarm optimization algorithm [J]. J of Systems Engineering and Electronics, 2013, 24(2): 324-334.

[22] Van den Bergh F, Engelbrecht A P. A cooperative approach to particle swarm optimization[J]. IEEE Trans on Evolutionary Computation, 2004, 8(3): 225-239.

[23] 姜建国, 李锦, 龙秀萍, 等. 采用小生境技术的混合蛙跳算法[J]. 计算力学学报, 2012, 29(6): 960-965.
(Jiang J G, Li J, Long X P, et al. A shuffled frog leaping algorithm using niche technology[J]. Chinese J of Computational Mechanics, 2012, 29(6): 960-965.)

(责任编辑：齐 霁)