自适应惯性权重的混沌粒子群算法研究

徐玉杰[1],仇 雷[2],刘 清[1]

(1. 南京师范大学 计算机科学与技术学院,江苏 南京 210046)
(2. 解放军八二医院 信息科,江苏 淮安 223001)

[摘要] 为了克服传统粒子群算法(PSO)的早熟和局部最优问题,提出了一种新的自适应惯性权重的混沌粒子群算法(ACPSO 算法).该算法采用分段 Logistic 混沌映射的方法产生初始种群,并根据种群的进化状态来动态调整惯性权重.在详细阐述算法的种群初始化过程和动态调整惯性权重的过程之后,对经典的测试函数分别采用几种改进的 PSO 算法和 ACPSO 算法对其进行了测试,与其他几种方法相比,ACPSO 算法的全局搜索能力有了显著的提高,并且能有效地避免早熟收敛问题,同时也说明 ACPSO 算法应用的可行性和有效性.

[关键词] PSO 算法,早熟收敛,混沌种群,自适应惯性权重
[中图分类号] TP18 [文献标志码] A [文章编号] 1672-1292(2012)01-0064-06

Chaotic Particle Swarm Optimization With Adaptive Inertia Weight

Xu Yujie[1], Qiu Lei[2], Liu Qing[1]

(1. School of Computer Science and Technology, Nanjing Normal University, Nanjing 210046, China)
(2. Department of Information, No 82 Hospital of PLA, Huaian 223001, China)

Abstract: To overcome the problem of premature convergence and local optimal in conventional particle swarm optimization (PSO), a new adaptive inertia weight chaos particle swarm optimization (ACPSO) is presented. The algorithm generates initial population with segmented logistic map, and varies inertia weight dynamically based on the evolutionary state of the population. After the detailed illustrations of how to generate initial population and how to adjust the inertia weight, this paper tests some classical functions with some improved PSO algorithms and ACPSO algorithm. Compared with other algorithms, the ACPSO algorithm not only has a great advantage of convergence property, but also avoids the premature convergence problem effectively, and at the same, it shows the feasibility and validity of the ACPSO algorithm.

Key words: PSO algorithms, premature convergence, chaos population, adaptive inertia weight

粒子群 PSO 算法(Particle Swarm Optimization)是在 1995 年由 Kennedy 和 Eberhart 提出的一种新型进化计算方法[1,2],属于求解全局最优化的群体智能算法.与其他全局优化算法(如遗传算法)一样,PSO 算法也存在着早熟收敛现象[3].目前解决这一问题的主要方法:一是对粒子群算法中的惯性权重参数进行调整,二是将 PSO 算法与其他优化算法(如遗传算法、模拟退火算法等)相结合[4-7].虽然改进之后的性能比原始的 PSO 算法有一定程度的提高,可还是不能从根本上解决早熟收敛问题.本文提出一种自适应惯性权重的混沌粒子群算法(ACPSO),利用混沌映射的遍历性和随机性将混沌映射用于粒子群算法的初始种群产生,并通过判断 PSO 算法的进化状态来自适应地调整惯性权重的值,实验结果表明:与传统的粒子群算法相比,ACPSO 算法的全局收敛性得到了显著提高,能有效避免粒子群优化算法中的早熟收敛问题.

1 基本粒子群优化算法及其早熟收敛问题

PSO 优化算法与其他进化寻优算法相类似,将优化的参数组合成群体,再通过环境的适应度使群体中

收稿日期:2011-12-22.
基金项目:国家自然科学基金(61103185)、江苏省高校自然科学基金(10KJD520004).
通讯联系人:刘 清,博士,教授,研究方向:智能测控技术.E-mail:njnulq@163.com

的个体向好的区域移动. 粒子群的个体(这里称作粒子)代表问题的一个可能解,每个粒子具有位置和速度两个特征. 设在 D 维搜索空间中,第 i 个粒子的位置可以表示成 X_i,粒子的速度表示成 V_i. 粒子位置坐标对应的目标函数值即可作为该粒子的适应度,算法通过适应度来衡量粒子的优劣. 算法首先初始化一群随机粒子,然后通过迭代找到最优解. 在每一次迭代中,粒子通过跟踪两个"极值"来更新自己: 一个是粒子本身所找到的最优解,即个体极值 $pBest_i$; 另一个是整个粒子群到目前为止找到的最优解,称之为全局极值 $gBest$,最后输出的 $gBest$ 就是算法得到的最优解. 粒子在每一次迭代中找到上述两个极值后,粒子通过一定的规则进化,第 ij 粒子从第 t 代进化到 $t+1$ 代,就根据下面两个公式来更新自己的速度与位置:

$$\boldsymbol{V}_i(t+1) = w\boldsymbol{V}_i(t) + c_1 rand(\boldsymbol{pBest}_i(t) - \boldsymbol{X}_i(t)) + c_2 rand(\boldsymbol{gBest}_i(t) - \boldsymbol{X}_i(t)), \tag{1}$$

$$\boldsymbol{X}_i(t+1) = \boldsymbol{X}_i(t) + \boldsymbol{V}_i(t+1), \tag{2}$$

其中,w 为惯性权重,其作用在于维护全局和局部搜索能力的平衡; c_1 和 c_2 为加速因子,均为正实数,表示将每个粒子推向 $pBest$ 和 $gBest$ 的加速度的权重; $rand$ 为 $[0,1]$ 范围内的随机数.

而 $pBest_i$ 和 $gBest$ 在每一维上迭代过程是:

$$\boldsymbol{pBest}_i(t+1) = \begin{cases} \boldsymbol{pBest}_i(t), \text{ if } F(\boldsymbol{X}_i(t+1)) > F(\boldsymbol{pBest}_i(t)), \\ \boldsymbol{X}_i(t+1), \text{ if } F(\boldsymbol{X}_i(t+1)) < F(\boldsymbol{pBest}_i(t)), \end{cases} \tag{3}$$

$$\boldsymbol{gBest}(t+1) = \min\{F(\boldsymbol{pBest}_0(t+1)), F(\boldsymbol{pBest}_1(t+1)), \cdots, F(\boldsymbol{pBest}_{D-1}(t+1))\}, \tag{4}$$

其中,$F(\cdot)$ 为适应度函数.

粒子群优化算法的结构相对简单,运行速度很快. 但是,算法运行过程中,如果某粒子发现一个当前最优位置,其他粒子将迅速向其靠拢. 如果该最优位置为一局部最优点,粒子群就无法在解空间内重新搜索,因此,算法陷入局部最优,出现了所谓的早熟收敛现象.

2 自适应惯性权重的混沌粒子群算法($ACPSO$)

2.1 位置和速度的混沌初始化

对于全局收敛的 PSO 算法,初始种群是影响收敛的重要因素[8]. 由于最优解的区域难以确定,如果初始种群选择的比较好,则算法经过迭代搜索之后会迅速达到全局最优. 而如果初始种群选择不当,可能一开始就会让算法陷入局部最优,从而影响整个算法的全局收敛性. 为了解决算法的初始种群对粒子群算法收敛性的影响,许多学者也提出了各种解决方法[9,10]. 本文根据混沌映射的特点,将它与改进的自适应粒子群算法相结合,构成混沌初始种群的自适应粒子群算法.

混沌是由确定性方程得到的具有随机性的运动状态,它是一种普遍存在的非线性现象,其行为复杂,类似于随机运动,具有初值敏感性、遍历性等特点[11]. 将混沌优与 PSO 算法相结合,利用混沌搜索的随机性和遍历性,能够很好地弥补 PSO 算法中用普通随机函数产生的初始种群所带来的缺陷. 使用混沌序列来初始化粒子的位置和速度,既不改变 PSO 算法初始时的随机性本质,也提高了种群的多样性和粒子搜索的遍历性. 本文采用文献[12]中的分段 Logistic 混沌映射来初始化各粒子的位置与速度,主要步骤是先在 D 维搜索空间产生随机向量分别作为 X_1 和 V_1,其每个分量的数值在(0,1)之间; 然后根据下式(5)迭代产生初始混沌种群的位置与速度,得到 $2*(N-1)$ 个向量,分别是 $X_2, X_3, \cdots, X_N$ 和 $V_2, V_3, \cdots, V_N$; 最后根据式(6)将初始的位置与速度分别映射到指定的搜索区间 $(-Sid, Sid)$ 范围之内,这样就完成了利用分段 Logistic 混沌映射来产生 ACPSO 算法的初始种群.

$$\boldsymbol{X}(i+1,d) = \begin{cases} 4*4*\boldsymbol{X}(i-1,d)*(0.5-\boldsymbol{X}(i-1,d)), & 0 < \boldsymbol{X}(i-1,d) < 0.5, \\ 1-4*4*(1-\boldsymbol{X}(i-1,d))*(\boldsymbol{X}(i-1,d)-0.5), & 0.5 \leqslant \boldsymbol{X}(i-1,d) < 1. \end{cases}$$

$$\boldsymbol{V}(i+1,d) = \begin{cases} 4*4*\boldsymbol{V}(i-1,d)*(0.5-\boldsymbol{V}(i-1,d)), & 0 < \boldsymbol{V}(i-1,d) < 0.5, \\ 1-4*4*(1-\boldsymbol{V}(i-1,d))*(\boldsymbol{V}(i-1,d)-0.5), & 0.5 \leqslant \boldsymbol{V}(i-1,d) < 1. \end{cases} \tag{5}$$

$$\boldsymbol{X}(i,d) = S_{id}*(2*\boldsymbol{X}(i,d)-1),$$

$$\boldsymbol{V}(i,d) = S_{id}*(2*\boldsymbol{V}(i,d)-1). \tag{6}$$

2.2 惯性权重的自适应调整

在公式(1)所描述的标准 PSO 算法中,惯性权重的作用主要表现在两个方面,其一是用来控制历史速度对当前速度的影响,另一个是用来平衡全局搜索和局部开发之间的平衡. 当 w 取值较大时,wV_i 项的值

加大,则搜索范围加大,提高了全局搜索能力,可以增加种群的多样性. 而 w 取值较小时,V 的速度变化范围减弱,增强局部挖掘最优解的能力,加速收敛. 在标准 PSO 算法中,w 值的取得通常是采用随迭代次数增加而线性递减的方式,在这种取值方法中,存在一些问题. 首先,如果在运行初期探测到较优点,则希望能迅速收敛于全局最优点,而 w 的线性递减减缓了算法的收敛速度; 其次,在算法的运行后期,随着 w 的减小,导致全局搜索能力下降,多样性减弱,容易陷入局部最优. 因此,为了克服粒子群的早熟收敛问题,可以采用根据种群当前的进化状态来动态改变惯性权重 w 的方法[13].

自适应惯性权重的粒子群算法主要有两个操作. 首先计算种群的适应度函数,通过计算出每一代种群中粒子的分布情况,来评估当前种群的进化状态(我们定义4个进化状态: 探测、开发、收敛、跳出),然后根据计算出的进化因子的值来判断当前种群处于哪一个状态,若处于收敛状态时,则使用最优学习策略来对全局最优值进行学习以避免种群进入局部最优. 惯性权重的值则是根据代表当前进化状态的进化因子的值来计算. 下面,首先给出种群的进化状态的判断方法.

2.2.1 进化状态的评估

评估当前种群的进化状态,其思想是根据当前种群中各粒子的分布情况来判断当前的进化状态,主要由以下几个步骤.

(1) 根据公式(7)计算出在当前状态下,第 i 个粒子与其他所有粒子的平均距离,在本文中使用欧几里德度量来计算:

$$d_i = \frac{1}{N-1}\sum_{j=1,j\neq i}^{N}\sqrt{\sum_{k=1}^{D}(x_i^k - x_j^k)^2}. \tag{7}$$

(2) 把全局最优粒子的 d_i 标记为 d_g,比较所有的距离找出最大距离 $d_{\max}$ 和最小距离 $d_{\min}$,根据公式(8)计算出代表当前进化状态的进化因子 f 的值:

$$f = \frac{d_g - d_{\min}}{d_{\max} - d_{\min}} \in [0,1]. \tag{8}$$

(3) 根据进化因子 f 的值,把 f 分为4个集合 S_1、S_2、S_3 和 S_4 分别来代表探测、开发、收敛和跳出这4个状态. 但由经验可知,对进化因子 f 状态过渡的划分是不确定的、模糊的,因此可以使用模糊分类的方法来划分种群的进化状态. 而模糊分类的关键是隶属函数,由经验所得隶属函数如下所示.

1) 探测阶段: 当 f 的值是中等到大的时候代表当前种群是属于 S_1 阶段,此时的隶属函数定义为:

$$\mu S_1(f) = \begin{cases} 0, & 0 \leqslant f \leqslant 0.4 \\ 5*f-2, & 0.4 < f \leqslant 0.6 \\ 1, & 0.6 < f \leqslant 0.7 \\ -10*f+8, & 0.7 < f \leqslant 0.8 \\ 0, & 0.8 < f \leqslant 1 \end{cases} \tag{9}$$

2) 开发阶段: 当 f 的值收缩时代表当前种群处于 S_2 阶段,此时的隶属函数定义为:

$$\mu S_2(f) = \begin{cases} 0, & 0 \leqslant f \leqslant 0.2 \\ 10*f-2, & 0.2 < f \leqslant 0.3 \\ 1, & 0.3 < f \leqslant 0.4 \\ -5*f+3, & 0.4 < f \leqslant 0.6 \\ 0, & 0.6 < f \leqslant 1 \end{cases} \tag{10}$$

3) 收敛阶段: 当 f 的值比较小时代表当前种群处于 S_3 阶段,此时的隶属函数定义为:

$$\mu S_3(f) = \begin{cases} 1, & 0 \leqslant f \leqslant 0.1 \\ -5*f+1.5, & 0.1 < f \leqslant 0.3 \\ 0, & 0.3 < f \leqslant 1 \end{cases} \tag{11}$$

4) 跳出阶段: 当 ACPSO 算法跳出局部最优时,全局最优粒子远离种群中心. 因此,当 f 的值最大时代表当前种群处于 S_4 阶段,此时的隶属函数定义为:

$$\mu S_4(f) = \begin{cases} 0, & 0 \leqslant f \leqslant 0.7 \\ 5*f-3.5, & 0.7 < f \leqslant 0.9 \\ 1, & 0.9 < f \leqslant 1 \end{cases} \tag{12}$$

由式(9) 到式(12) 可知,在过渡阶段,两个隶属函数都被激活,根据 f 则可判断当前种群处于两个状态,这样就不能确定种群最终的进化状态. 此时为了确定最终的种群状态,本文采用无规则的"单态模式"方法来去模糊化,如 $\mu S_2(f) > \mu S_1(f)$,则当前属于 S_2 状态.

2.2.2 惯性权重的动态调整

正如前面所述,惯性权重是用来平衡全局和局部的搜索能力,许多研究者认为 w 的值应在探测阶段比较大而在开发阶段比较小. 然而随时间的变化来改变 w 的值也不一定总是正确的,由于进化因子 f 的值在探测阶段比较大而在收敛阶段比较小. 因此,在本文中考虑,惯性权重 w 随着进化因子 f 的值做如下 Sigmoid 映射:

$$w(f) = \frac{1}{1 + 1.5e^{-2.6f}} \in [0.4, 0.9], \ \forall f \in [0,1] . \tag{13}$$

这样,w 的值随着进化因子 f 而不是时间单调,因此,w 适合由 f 所代表的任何进化状态.

2.2.3 最优化学习策略(*Elitist Learning Strategy*)

当种群处于局部最优时,要采取一定的机制跳出局部最优,本文是采用最优化学习的方法在种群陷入局部最优时来跳出局部最优. ELS 方法是随机的选取全局最优粒子 ***gBest*** 中的某一维(每一维被选中的概率是相同的),对选中的该一维进行如下的高斯扰动:

$$\boldsymbol{gBest}^d = \boldsymbol{gBest}^d + (\boldsymbol{X}^d_{\max} - \boldsymbol{X}^d_{\min}) \cdot Gaussian(\mu, \sigma^2) \tag{14}$$

其中, $[X_{\max}{}^d, X_{\min}{}^d]$ 是种群中所有粒子的每一维的最大值和最小值. $Gaussian(\mu, \sigma^2)$ 是以0为均值的高斯扰动,σ 为标准偏差的高斯扰乱,σ 也称为最优学习率. 和其他随时间变化的神经网络训练策略一样,σ 是随着迭代次数线性递减的,计算方法如下所示:

$$\sigma = \sigma_{\max} - (\sigma_{\max} - \sigma_{\min}) \frac{gen}{\max gen} . \tag{15}$$

其中,$\sigma_{\max}$ 和 $\sigma_{\min}$ 分别是 σ 的上下界,代表学习所能到达的范围. gen 为算法当前迭代次数,maxgen 为算法的最大允许迭代次数. 实验研究表明,当 $\sigma_{\max} = 1.0, \sigma_{\min} = 0.1$ 时,在大多数测试函数上都能达到好的性能. 故在本文的实验中取 $\sigma_{\max} = 1.0, \sigma_{\min} = 0.1$.

3 *ACPSO* 算法的性能分析

在本文中,我们选取经典的测试函数来对 ACPSO 算法的性能进行测试. 对于每个测试函数,分别采用基本 PSO 算法(SPSO)、线性递减惯性权重的 PSO 算法(LDWPSO)、随机惯性权重的 PSO 算法(RandWPSO)、带收缩因子的 PSO 算法(CFPSO) 以及本文提出的 ACPSO 算法进行测试,然后通过对比来说明 ACPSO 算法的优越性. 对于每个测试函数,都是进行 30 次迭代运算,取其平均值作为最终的运算结果.

(1) 单峰 Sphere 函数:

$$f(x) = \sum_{i=1}^{D} x_i^2, \quad \text{其中 } D = 30, x_i \in [-100, 100]^D,$$

该函数的全局最优值为 0,其函数图像如图 1 所示,各种 PSO 算法对其测试结果如图 2 所示.

(2) 多峰 Ronsenbrock 函数:

$$f(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2], \text{其中 } D = 30, x_i \in [-10, 10]^D,$$

该函数的全局最优值为 0,其函数图像如图 3 所示,各种 PSO 算法对其测试结果如图 4 所示.

(3) 多峰 Rastrigin 函数:

$$f(x) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10], \text{其中 } D = 30, x_i \in [-5.12, 5.12]^D,$$

该函数的全局最优值为 0,其函数图像如图 5 所示,各种 PSO 算法对其测试结果如图 6 所示.

通过上述 3 个测试函数,可以看出,与其他几种 PSO 算法相比,在相同的迭代次数和相同的范围之内,ACPSO 算法总是能取得最好的最优值. 与其它几种算法相比,体现了 ACPSO 算法的优越性,也体现了 ACPSO 算法的可行性和有效性.

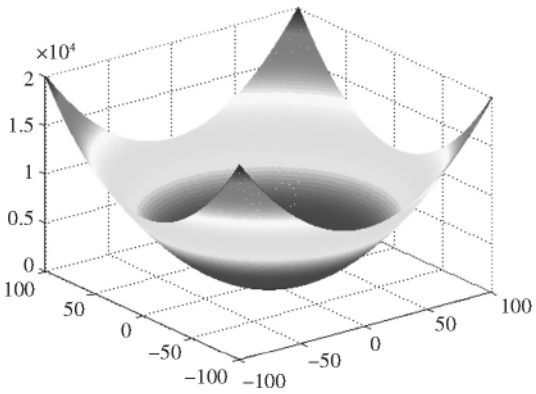

图1　单峰 Sphere 函数的图像

Fig.1　The unimodal Sphere function

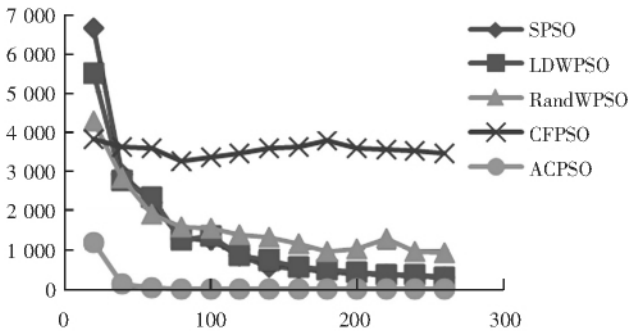

图2　Sphere 函数测试结果

Fig.2　The results of Sphere function

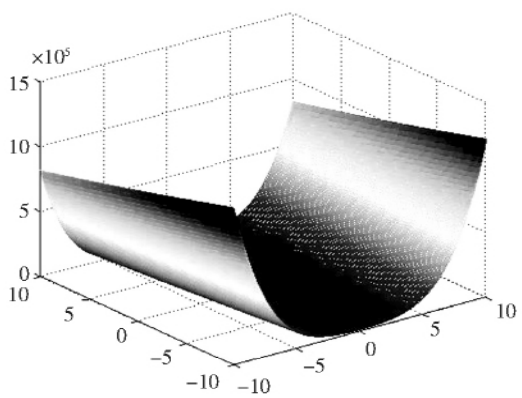

图3　多峰 Ronsenbrock 函数图像

Fig.3　The multimodal Ronsenbrock function

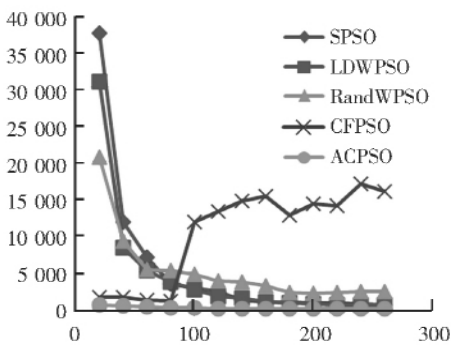

图4　Ronsenbrock 函数测试结果

Fig.4　The results of Ronsenbrock function

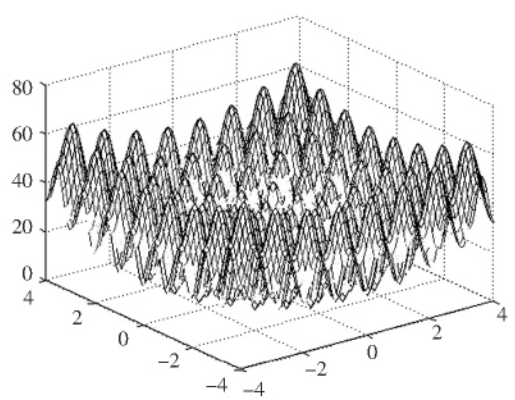

图5　多峰 Rastrigin 函数图像

Fig.5　The multimodal Rastrigin function

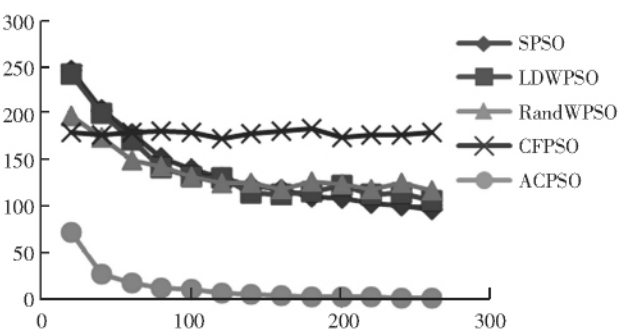

图6　Rastrigin 函数测试

Fig.6　The results of Rastrigin function

4　结语

本文首先介绍了标准粒子群算法的思想,指出它所存在的早熟收敛问题. 然后在此问题的基础上对标准粒子群算法进行了两点改进,其一是利用混沌的方法来产生初始种群; 其二是根据种群的进化状态改变自适应惯性权重,最后,通过对经典测试函数的测试,并与其他几种 PSO 算法的测试结果比较和分析,验证了自适应惯性权重的粒子群算法(ACPSO)的可行性和有效性.

[参考文献] (References)

[1] Kennedy J, Eberhart R. Particle swarm optimization [C] // Proc nternational Conference on Neural Networks. Australia: Perth, 1995: 1 942-1 948.

[2] Eberhart R C, Kennedy J. A new optimizer using particle swarm theory [C] // Proc 6th Int. Symp. Micro Machine and Human Science. Japan: Nagoya,1995: 39-43.

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

[4] Park J B, Jeong Y W, Lee W N. An improved particle swarm optimization for economic dispatch problems with non-smooth cost functions [C] // Power Engineering Society General Meeting. Montreal: 2006: 1-7.

[5] You Zhiyu, Chen Weirong, He Guojun, et al. Adaptive weight particle swarm optimization algorithms with constriction factor [C] // International Conference of Information Science and Management Engineering. China: Xi'an, 2010: 245-248.

[6] Fan S K, Liang Y C, Zahara E. A genetic algorithm and a particle swarm optimizer hybridized with Nelder-Mead simplex search [J]. Computers and Industrial Engineering, 2006,50(4): 401-425.

[7] Wang Xihuai, Li Junjun. Hybrid particle optimization with simulated annealing [C] // Proceddings of International Conference of Machine Learning and Cybernetics. China: Shanghai, 2004: 2 402-2 405.

[8] He Dakuo, Chang Hongrui, Chang Qing, et al. Particle swarm optimization based on the initial population of clustering [C] // International Conference on Natural Computation. China: Yantai, 2010: 2 664-2 667.

[9] 高鹰,谢胜利. 混沌粒子群算法 [J]. 计算机科学, 2004, 31(8): 13-15.
Gao Ying, Xie Shengli. Chaos particle swarm optimization [J]. Computer Science, 2004, 31(8): 13-15. (in Chinese)

[10] Wang H, Wu Z J, Wang J, et al. A new population initialization method based on space transformation search [C] // Fifth International Conference on Natural Computation, China: Tianjin, 2009: 332-336.

[11] Wang W, Peng L. Chaos particle swarm optimization combined with isolation niche [J]. Systems Engineering and Electronics, 2008, 30(6): 1 151-1 154.

[12] 范九伦,张雪峰. 分段 Logistic 混沌映射及其性能分析 [J]. 电子学报, 2009, 37(4): 720-725.
Fan Jiulun, Zhang Xuefeng. Piecewise logistic chaotic map and its performance analysis [J]. Acta Electronica Sinica, 2009, 37(4): 720-725. (in Chinese)

[13] Zhan Z H, Zhang J, Li Y, et al. Adaptive particle swarm optimization [J]. IEEE Transactions on Systems, Man and Cybernetics, 2009, 39(6): 1 362-1 381.

[责任编辑: 刘 健]