文章编号: 1001-0920 (2013) 10-1573-05

一种新的求解动态连续优化的分层粒子群算法

朱庆保[1], 徐晓晴[1], 朱世娟[2]

(1. 南京师范大学 a. 计算机科学与技术学院, b. 江苏省信息安全保密技术工程中心, 南京 210097; 2. 安庆师范学院 计算机信息学院, 安徽 安庆 246001)

摘 要: 为了高效求解动态连续优化问题, 提出一种分层粒子群优化算法. 该算法将动态函数定义域分成 Q 个子空间, 每个空间用一个粒子群作为第一层进行独立搜索, Q 个子空间的最优粒子再组成一个全局粒子群进行全局搜索, 以达到全局牵引的作用, 同时提出探测环境和响应环境的策略. 利用经典的动态函数对算法进行测试, 结果表明所提出算法能够迅速适应环境变化和跟踪最优解的变化, 效果令人满意.

关键词: 粒子群算法; 动态优化; 连续优化; 分层

中图分类号: TP301.6 **文献标志码:** A

A new hierarchical PSO algorithm for solving dynamic and continuous optimization problems

ZHU Qing-bao[1], *XU Xiao-qing*[1], *ZHU Shi-juan*[2]

(1a. School of Computer Science and Technology, 1b. Jiangsu Research Center of Information Security & Privacy Technology, Nanjing Normal University, Nanjing 210097, China; 2. School of Computer Information, Anqing Normal University, Anqing 246001, China. Correspondent: ZHU Qing-bao, E-mail: zhuqingbao@njnu.edu.cn)

Abstract: In order to efficiently solve the dynamic and continuous optimization problems, a hierarchical PSO algorithm is proposed. The solution space of a dynamic optimization problem is divided into Q subspaces. One particle swarm is assigned to each subspace to search independently as the first layer. Then the best particles in the Q particle swarms as the second layer search the global domain so as to guide the other particles in each subspace. Moreover, an environment detection and response strategy is proposed. Numerical experiments on the classical dynamic function demonstrate that the algorithm can adapt dynamic environment and track a changing optimum quickly.

Key words: particle swarm optimization; dynamic optimization; continuous; layering

0 引 言

优化问题的目标函数、约束条件和解随时间发生变化称为动态优化, 其难点在于最优解的位置是动态变化的, 算法需要跟踪变化的极值. 基于群体协作的粒子群算法 (PSO) 具有简单易实现、全局搜索能力强等优点, 是解决这类问题的首选算法[1]. 然而, PSO 算法在运行到后期时, 对环境的变化失去适应能力, 不能跟踪变化了的极值, 所以很多学者针对动态优化问题对 PSO 算法进行了改进, 其基本思想均是提高算法的多样性, 使算法适应环境的变化. 如: Blackwell 等[2]提出了一种带电粒子 PSO (CPSO), 通过增加加速因子模拟带电粒子对其他带电粒子的排斥效果, 以此保持系统的多样性, 应对环境的动态变化; 文献 [3] 提出了一种自适应惯性权重并以辅群提高搜索多样性的方法; AutomicSwarm 方法[4]在种群中设定一部分个体作为带电粒子, 当两个粒子的距离在一定范围内时, 粒子之间存在着互相排斥的力, 这样, 便保持了群体的多样性, 使算法具有较好的全局探测能力; IPSO 方法[4]是一种改进的 PSO 方法. 为了提高算法的搜索多样性, 一些学者还提出了其他算法, 如连续多蚁群算法[5]、带电粒子群算法[6]、树形结构的 PSO 算法 (H-PSO)[7]、引入量子微粒的算法[8]、基于种群多样性的微粒群算法[9]、复合粒子群算法[10]、对偶遗传算法[11]、自适应免疫算法[12]等.

上述算法增加多样性的方式具有随机性, 需要的计算量和时间消耗较大. 另一方面, 这些算法方法

收稿日期: 2012-06-08; **修回日期:** 2012-09-06.

基金项目: 国家自然科学基金项目(60673102, 61073118); 江苏省高校自然科学基金项目(10KJD520004).

作者简介: 朱庆保(1955–), 男, 教授, 博士生导师, 从事人工智能、机器人技术等研究; 徐晓晴(1988–), 女, 硕士生, 从事智能技术及应用的研究.

比较单一, 虽然在一定程度上改进了环境的适应能力, 但对提高收敛速度并没有帮助甚至起相反的作用. 鉴于此, 本文提出了一种基于分层的动态PSO算法(LDPSO), 该算法由若干分区分组粒子群实施基层的分区局部搜索, 由每组最优粒子组成全局搜索粒子群实施全局搜索, 使各子群在并行独立搜索的同时又能相互交流学习, 从而高效地跟踪变化的极值点; 同时, 提出了一种环境检测和环境响应方法, 以增加算法搜索的多样性.

1 PSO算法及在动态连续优化中的局限性

标准PSO算法及其参数见文献[9], 粒子j在i维上按下式更新速度和位置:

$$v_{ji}(t)=\omega v_{ji}(t)+c_1r_{1i}(t)(p_{ji}(t)-x_{ji}(t))+ c_2r_{2i}(t)(p_{gi}(t)-x_{ji}(t)), \tag{1}$$

$$x_{ji}(t+1)=x_{ji}(t)+v_{ji}(t+1), \tag{2}$$

其中P_g和P_j分别为全局极值和个体极值. 动态优化问题需要尽可能地跟踪最优解的变化轨迹, 但PSO算法存在如下局限性: 1) 每个粒子通过跟踪P_j和P_g逐渐逼近更优解, 但动态优化问题中的P_j和P_g会随时间发生变化, 此时粒子陷入对先前环境的寻优, 难以有效逼近动态变化的最优位置; 2) 动态优化问题变量的状态是时变的, 但PSO算法在进化后期会出现收敛的状态, 失去对环境的适应能力.

2 LDPSO算法的基本策略

2.1 问题描述和相关定义

一般动态环境下非约束的最优化问题描述为

$$\min f(X_t,t),\ X_t\in S_t\subseteq R^D;$$
$$\min_i\leqslant x_i\leqslant\max_i,\ i=1,2,\cdots,D. \tag{3}$$

其中: $X_t=(x_1,x_2,\cdots,x_D,t)$; $S_t\subseteq R^D$为随时间变化的解空间, D为维数; $f(X_t,t)$为目标函数, 也是关于时间t的函数.

定义 1 对于最小值动态优化问题(式(3)), 若存在$X_t^*\in S_t$, 使得对于$\forall X_t\in S_t$均有

$$f(X_t^*,t)\leqslant f(X_t,t) \tag{4}$$

成立, 则称X_t^*为$f(X_t,t)$在S_t上第t代的全局最优解. 将式(4)中的不等号反向, 即可得到关于求最大值最优解的定义. 本文算法以求解最小值为例.

2.2 分层粒子群算法策略

2.2.1 解空间的划分和分组方法

首先将优化问题的解空间按定义域划分为Q个子空间, 划分方法为

$$h_s^i=(\max_i-\min_i)/Q,$$
$$i=1,2,\cdots,D,\ s=1,2,\cdots,Q. \tag{5}$$

其中: h_s^i为第s个子空间i维上的搜索长度, $[\min_i,\max_i]$为第i维的定义区间. 任意子空间任意维上的搜索区间可以表示为

$$[\min_i+(s-1)h_s^i,\ \min_i+sh_s^i]. \tag{6}$$

每个子空间分配一组粒子, 每组均为N个, 各组粒子在分配的子空间内进行独立的搜索.

2.2.2 粒子位置的混沌初始化

利用混沌序列产生初始种群的初始位置, 为后续搜索多样性奠定基础, 步骤如下.

Step 1: 产生$N\times Q$个D维的混沌粒子位置. 利用下式产生D个混沌变量:

$$l_{s,j}^{i+1}=\mu l_{s,j}^i(1-l_{s,j}). \tag{7}$$

其中: $\mu=3.99$, $i=0,1,\cdots,D-1$, $j=1,2,\cdots,N$, $s=1,2,\cdots,Q$, i为混沌变量的序列号. 令$i=0$, $l_{s,j}^0=\text{rand}(\ ),j=1,2,\cdots,N$, $l_{s,j}^0$为第j个粒子在第s个子空间混沌向量的初值, rand( ) $\in[0,1]$为[0,1]间的随机值. 利用式(7)得到第1个子空间第1个粒子的D维混沌向量$(l_{1,1}^1,l_{1,1}^2,\cdots,l_{1,1}^D)$, 同理, 产生另外$N(Q-1)$个混沌向量, 由这$NQ$个混沌向量组成初始混沌种群.

Step 2: 产生$N\times Q$个D维的粒子位置. 利用下式完成从混沌变量到粒子位置的映射:

$$x_{s,j}^i=\min_i+(s-1)h_s^i+h_s^il_{s,j}^i. \tag{8}$$

其中: $x_{s,j}^i$为第j个粒子在第s个子区间内第i维的分量, $l_{t,j}^i$为第j个粒子在第s个子区间内对应的第i维的混沌分量.

2.2.3 对粒子位置进行评价

对N个粒子的初始位置利用下式进行评价:

$$y_s^j=f(x_{s,j}^1,x_{s,j}^2,\cdots,x_{s,j}^D). \tag{9}$$

其中: y_s^j为第s个子空间第j个粒子的适应值, $f(\cdot)$为根据求解问题构造的适应值函数. 由式(9)可得到

$$y_s^{\min}=\min(y_s^1,y_s^2,\cdots,y_s^N),\ s=1,2,\cdots,Q, \tag{10}$$

其中$y_s^{\min}$为本代第s个子空间内的最好解. 每组粒子在各子空间分别搜索MAX代后, 利用下式得到全局最优解:

$$y_{\min}=\min(y_1^{\min\,l},y_2^{\min\,l},\cdots,y_Q^{\min\,l}), \tag{11}$$

$y_s^{\min\,l}$为第s个子空间内得到的最优解.

2.2.4 分层策略

设任意一组粒子得到的最优解为$(X_s^{\min\,l},y_s^{\min\,l})$, $s=1,2,\cdots,Q$, 则组成第2层的精英粒子为

$$\{(X_1^{\min\,l},y_1^{\min\,l}),(X_2^{\min\,l},y_2^{\min\,l}),\cdots,(X_Q^{\min\,l},y_Q^{\min\,l})\},$$

记为$(X_{s2}^{\min\,l},y_{s2}^{\min\,l})$. 该精英粒子群在全局范围内根据式(1)和(2)进行全局搜索, 得到个体极值为$p_{s2}^{\min\,l}$, 全局极值为$p_{g2}^{\min\,l}$.

2.2.5 环境变化的检测

为了检测解空间中的环境是否发生变化, 对每个组的最优解利用下式进行检测:

$$C(t)=|f(X_s^{\min l}(t))-f(X_s^{\min l}(t-1))|. \quad (12)$$

其中: t 为迭代次数, $f(X_s^{\min l}(t-1))$ 为 $t-1$ 代第 s 组粒子得到的最优解值, $X_s^{\min l}$ 为最优解在解空间的位置; $f(X_s^{\min l}(t))$ 为解空间同一位置到第 t 代时的适应值. σ 的值通过实验确定, 如果 $C(t)\leqslant\sigma$, 则视为环境没有发生变化或变化不大.

2.2.6 环境变化的响应

假设探测到环境已发生变化, 如果此时粒子分布过密, 则调整粒子位置. 设该组内最优粒子与其他粒子的平均距离为 $\mathrm{Diver}_s(t)$, 粒子分布的平均距离为 $\Delta_s(t)$, 利用下式判断粒子分布情况:

$$\mathrm{Diver}_s(t)=\frac{1}{N}\sum_{j=1,j\neq i}^{N}\sqrt{\sum_{i=1}^{D}(x_{s,i}^{\min l}-x_{s,j}^{i})^2}, \quad (13)$$

$$\Delta_s(t)=\frac{1}{N\times D}\sum_{i=1}^{D}h_s^i. \quad (14)$$

当 $\mathrm{Diver}_s(t)<\Delta_s(t)$ 时, 多样性较差, 需要利用下式更新当前组的粒子位置:

$$x_{s,j}^{i}(t+1)=$$
$$x_{s,j}^{i}(t)+\eta(x_{s,j}^{i}(k)-x_{s,i}^{\min l})+\xi(x_{s,j}^{i}(t)-P_g^i). \quad (15)$$

其中: $\eta=\mathrm{rand}(\,)H[s][i]/2$, $\xi=\mathrm{rand}(\,)H[s][i]/2$.

2.3 算法步骤

分层粒子群算法步骤如下.

Step 1: 分组初始化.

Step 1.1: 设置总的最大迭代次数为 MAX, 代数计数器 $t=0$, 各组每次的迭代次数为 A, 计数器 $m=0$, 按式 (5) 划分定义域, 每个子空间分配一组粒子, 分组数为 Q, 每组种群大小为 N, ω 初值设为 0.9, $c_1=c_2=2$, 各组的全局最优解 $y_s^{\min l}=\infty$;

Step 1.2: 初始化各粒子的速度和位置 x_j^i;

Step 1.3: 利用 $f(\cdot)$ 得到各粒子的适应值 $y_j=f(X)$, 则有 $y_{\min}=\min(y_1,y_2,\cdots,y_N)$. 令当前全局最优解 $P_g=f^{-1}(y_{\min})$, 每个个体的极值 $P_j=X_j$.

Step 2: 粒子按式 (1) 和 (2) 更新速度和位置. 惯性权重满足线性递减, 即 $\omega=0.9-0.5(\mathrm{MAX}-t)/\mathrm{MAX}$, 其中 t 为当前已迭代次数. 同时按下式进行边界检查并转换:

$$x_j^i=\begin{cases}\min\limits_i+sh_s^i-\Delta x,\ x_j^i>\min\limits_i+sh_s^i;\\ \min\limits_i+(s-1)h_s^i+\Delta x,\ x_j^i<\min\limits_i+(s-1)h_s^i.\end{cases}$$
$$j=1,2,\cdots,N,\ i=1,2,\cdots,D,\ s=1,2,\cdots,Q. \quad (16)$$

其中: $\Delta x=\mathrm{rand}(0,h_s^i)$, $\mathrm{rand}(0,h_s^i)$ 产生 $(0,h_s^i)$ 间的随机数.

Step 3: 利用式 (9) 分别对各分区粒子位置 X_j 进行评价. 由式 (10) 得到各区的本代最好解 $y_s^{\min}$. 若 $y_s^{\min}<y_s^{\min l}$, 则令 $y_s^{\min l}=y_s^{\min}$, 得到各区最优粒子位置 $X_s^{\min l}(t)=f^{-1}(y_s^{\min l})$. 同时, 更新 $Pg=f^{-1}(y_{\min})$ 和每个粒子的当前最好解, 即若 $y_s^j<P_j$, 则令 $P_j=f^{-1}(y_s^j)$.

Step 4: 令 $m=m+1$, 如果 $m<A$, 则转至 Step 2.

Step 5: 将 $X_s^{\min l}(t)(s=1,2,\cdots,Q)$ 作为第 2 层 PSO 搜索的个体, 记为 $X_{s2}^{\min l}=X_s^{\min l}(t)$. 个体极值为 $P_{s2}^{\min l}=X_s^{\min l}(t)$, 全局极值为 $P_{s2}^{\min l}=f^{-1}(y_{\min})$.

Step 6: 第 2 层个体按式 (1) 和 (2) 进行二次搜索, 并利用上述同样算法对各粒子位置进行评价, 同时对各区的最优粒子进行更新, 如果 $y_{s2}^{\min l}<y_s^{\min l}$, 则 $X_s^{\min l}=X_{s2}^{\min l}$, $y_s^{\min l}=y_{s2}^{\min l}$.

Step 7: 以一定条件更新目标函数 $f(t)$, 本实验中, 每经历 Δg 代, 令函数以偏移量 offset 发生一次动态变化.

Step 8: 设置 η 为常数, 利用式 (12) 计算各区的 $C(t)$, 如果 $C(t)\leqslant\eta$, 则转至 Step 10.

Step 9: 响应环境变化.

Step 9.1: 利用式 (13) 和 (14) 计算各区的种群多样性 $\mathrm{Diver}_t(t)$ 和 $\Delta_s(t)$;

Step 9.2: 若 $\mathrm{Diver}_t(t)>\Delta_s(t)$, 则 s 区粒子较为分散, 无需重新定位粒子的位置, 转至 Step 10;

Step 9.3: 若 $\mathrm{Diver}_t(t)<\Delta_s(t)$, 则利用式 (15) 将该区粒子向组内最优粒子周围扩散, 并按式 (16) 进行边界检查并转换, 直到满足 $\mathrm{Diver}_t(t)>\Delta_s(t)$, 同时更新 $X_{s,j}(t)$ 对应的 $f(X_{s,j}(t))$.

Step 10: 部分组的极值更新.

Step 10.1: 重新计算上一代中该组各个体最优位置 $P_{s,j}(t-1)$ 在当前环境中的适应值 $f(P_{s,j}(t-1))$, 将较优者作为当前环境的个体极值, 记为 $P_{s,j}(t)$;

Step 10.2: 比较组内所有个体极值 $P_{s,j}(t)$, 选择最好的作为该组全局极值, 记为 $Pg_s(t)$.

Step 11: 令 $t=t+1$, 如果 $t>$MAX 或算法得到的最优值与理论最优值之差 $\Delta y_{\min}$ 满足精度要求 $\Delta y_{\min}<1\mathrm{e}-5$, 则转至 Step 12, 否则转至 Step 2.

Step 12: 利用式 (11) 计算全局最优解 $y_{\min}$ 并输出.

3 计算机数值仿真实验和算法比较

为了验证算法的有效性, 本文进行了大量的数值仿真实验, 实验设备为一般 PC 机, Intel(R) 赛扬 CPU 2.66 GHz, 256 内存, 仿真软件 VC++6.0.

表 1 分层优化效果测试

环境变化	维数	分组参数		分组分层优化结果		分组不分层优化结果	
		m_g	m_p	before	after	before	after
offset=0.01 $\Delta g=10$	30	3	10	186.05	47.3	210.23	90.2
offset=0.1 $\Delta g=50$	30	3	12	89.25	89.25	230.9	170.2
offset=1 $\Delta g=100$	30	4	10	187.64	74.35	238.5	141.3

表 2 3 种算法的比较结果

offset	参数设置		本文算法(LDPSO)				DGCMPSO		Diversity-PSO	
			before		after		before	after	before	after
	m_g	m_p	无检测	有检测	无检测	有检测				
1e-5	3	10	129.25	128.311	0	0	220.36	0.37	163.3	0
1e-3	3	10	134.37	132.17	0	0	222.59	15.98	169.05	0
1e-1	3	10	130.52	126.79	53.62	51.2	219.17	115.18	166.6	57.95
1.0	3	10	142.31	112.44	95.41	89.16	220.25	186.74	168.3	112.6

表 3 5 种算法平均误差

方法	实验 1	实验 2	实验 3	实验 4	实验 5	实验 6
PSO	0.457	0.939	0.707	0.756	0.488	0.853
AutomicSwarm	0.108	0.206	0.232	0.289	0.261	0.249
PSOwDOW	0.018	0.086	0.067	0.079	0.045	0.094
IPSO	0.037	0.071	0.051	0.101	0.096	0.06
LSPSO	4.686 e-05	5.185 4 e-05	5.148 4 e-05	6.500 5 e-05	2.399 4 e-05	3.766 9 e-05

3.1 动态测试函数介绍

由于抛物线的动态变化易于控制, 这一领域的很多文献均利用下式作为测试函数:

$$f(X)=\sum_{i=1}^{D}(X_i-\text{offset})^2, \tag{17}$$

其中 offset 用于控制抛物线的动态变化, 每经历 Δg 代, 函数发生一次动态变化. 为了便于与文献 [9,13] 作比较, 本文也用该函数对算法进行测试.

3.2 对算法的综合测试

表 1 为算法的综合测试结果, 是独立运行 100 次的平均值. 其中: before 和 after 分别为环境变化前后收敛到最优解所需的平均代数; m_g 和 m_p 分别为分组数和每组粒子数; offset $=0.01, 1.0, 10$, 对应的变化间隔为 $\Delta g=10, 50, 100$; 最优解控制精度 $\Delta y_{\min}<1\text{e}-5$. 可见, 分层后效果优越, 在 30 维上, 新环境里寻找最优的效率几乎是不分层的 2 倍.

3.3 与其他算法的比较

将本文算法与 Diversity-PSO算法[9]和DGCMPSO算法[13]进行对比. DGCMPSO 算法通过混沌变异加强种群多样性; Diversity-PSO 算法通过检测某一粒子的前后变化判断环境变化, 然后采取多样性检测, 如果较差, 则不断进行逃逸运动; 本文算法只有当个别组出现粒子集中情况时才进行有方向的扩散运动. 表 2 给出了这 3 种算法收敛速度的比较, 参数设置与文献 [9,13] 完全相同. 本文算法各组内循环 2 次, 分层循环一次才完成一次大循环迭代, 为了公平比较, 在表 2 中, 这样一次大循环被累计为 3 次迭代. 表 2 中的“无检测”指取消算法中的环境检测和响应. 由表 2 可见, 此时各算法每次迭代的算法复杂度基本相同, 但本文算法明显优于其他两种算法, 环境变化越大, 环境检测效果越明显.

为了进一步比较算法的先进性, 利用下式[4]:

$$f(X,Y)=\max_{i=1,2,\cdots,N}[H_i-R_i\sqrt{(X-X_i)^2+(Y-Y_i)^2}] \tag{18}$$

对算法进行测试, 并与相关文献进行比较. 其中: $f(X,Y)$ 为在 (X,Y) 位置的适应值, N 为环境中锥体的个数, (X_i,Y_i) 为第 i 个锥体的顶点, H 为第 i 个锥体的高度, R 为第 i 个锥体的斜度.

文献 [4] 对多种动态优化算法进行比较的结果和本文 LSPSO 算法的实验结果均列于表 3, 具体实验参数和方法与文献 [4] 完全相同, 且实验过程去掉了环境检测和环境响应. 由表 3 可见, LSPSO 算法的优化误差远小于其他方法.

4 结 论

针对利用粒子群算法进行连续动态优化时易出现多样性丢失、不易及时快速地跟踪动态变化极值的问题, 本文提出了一种分层的粒子群优化算法, 同时提出了动态环境检测策略和响应策略. 在分组基础上

的分层策略使得各组的最优粒子有相互学习的机会,从而能更好地指导各组粒子寻找解空间中潜在的解,极大地提高了种群在环境发生变化之前收敛的可能性. 另一方面, 环境发生变化后, 结合各组最优粒子密度的多样性检测, 有选择地对趋于集中的粒子进行有方向的扩散移动, 从而增强了种群的多样性, 以适应环境的变化. 仿真实验表明, 利用本文算法求解动态环境优化问题, 能够及时快速地对变化的环境作出合理响应, 效果令人满意.

参考文献(References)

[1] Angeline P J. Tracking extrema in dynamic environments[C]. Proc of the 6th Int Conf on Evolutionary Programming. Indianapolis: Springer, 1997, 1213: 336-345.

[2] Blackwell T M, Peter J B. Dynamic search with charged swarms[C]. Proc of the Genetic and Evolutionary Computation Conference. San Francisco: Morgan Kaufmann Publishers, 2002: 19-26.

[3] Iman Rezazadeh, Mohmmad Reza Meybodi. Particle swarm optimization algorithm in dynamic environments: Adapting inertia weight and clustering particles[C]. UKSim 5th European Symposium on Computer Modeling and Simulation. 2011: 76-82.

[4] 段晓东, 徐平, 王存睿, 等. 动态环境下粒子分群与种群多样性的关系研究[J]. 计算机科学, 2009, 36(3): 161-162.

(Duan X D, Xu P, Wang C R, et al. Relations between particle division and population diversity under dynamic environment[J]. Computer Science, 2009, 36(3): 161-162.)

[5] Pu Li-ming, Yu Huan-jun, Chen De-zhao. Continuous multi ant-colony optimization and its application in dynamic optimization problems of chemical processes[J]. J of Chemical Engineering of Chinese Universities, 2008, 22(5): 871-876.

[6] Blackwell T M, Bentley P. Don't push me! Collision-avoiding swarms[C]. Proc of the IEEE Congress on Evolutionary Computation. Honolulu, 2002: 1691-1696.

[7] Janson S, Middendorf M. A hierachical particle swarm optimizer for dynamic optimization problems[C]. Applications of Evolutionary Computing. Springer, 2004: 513-524.

[8] Blackwell T M, Branke J. Multi-swarms, exclusion and anti-convergence in dynamic environments[J]. IEEE Trans on Evolutionary Computation, 2006, 10(4): 459-472.

[9] 胡静, 曾建潮, 谭瑛. 动态环境下基于种群多样性的微粒群算法[J]. 系统仿真学报, 2007, 19(21): 4932-4935.

(Hu J, Zeng J C, Tan Y. Particle swarm optimizer based on diversity of particle in dynamic environments[J]. J of System Simulation, 2007, 19(21): 4932-4935.)

[10] Liu Li-li, Yang Sheng-xiang. Particle swarm optimization with composite particles in dynamic environments[J]. IEEE Trans on Systems, Man and Cybernetics, Part B: Cybernetics, 2010, 40(6): 1634-1648.

[11] Yang S. Non-stationary problem optimization using the primal-dual genetic algorithm[C]. Proc of the 2003 Congress on Evolutionary Computation. Leicester, 2003: 836-844.

[12] Zhang Zhu-hong, Qian Shu-qu. Adaptive immune algorithm and its track to dynamic function optimization[J]. Pattern Recognition and Artificial Intelligence, 2007, 20(1): 85-94.

[13] Dong Dian-min, Jie Jing, Zeng Jian-chao, et al. Chaos-mutation-based particle swarm optimizer for dynamic environment[C]. Proc of Int Conf on Intelligent System and Knowledge Engineering. 2008: 1032-1037.

下 期 要 目