文章编号: 1001-0920(2009)06-0859-05

适应性粒子群寻优算法 II

罗辞勇, 陈民铀, 韩 力
(重庆大学 a. 输配电装备及系统安全与新技术国家重点实验室, b. 电气工程学院, 重庆 400044)

摘 要: 适应性粒子群寻优算法 I(APSO- I)是在有序的决策中始终引入随机的、不可预测的决定. 为解决 APSO-I 算法收敛深度不够的问题,提出适应性粒子群寻优第 II 代算法(APSO- II). APSO- II算法是将有序(标准 PSO 粒子群寻优)和无序(自适应寻优)进行适当的分离,以发挥各自的优势. 在自适应寻优阶段,通过在最优粒子邻域空间探寻更优化的解. 一但新的优化解被发掘,便利用标准 PSO 快速寻优. 典型复杂函数优化的仿真结果表明,APSO- II在收敛速度和收敛深度上均优于 DPSO(耗散型 PSO), HPSO(自适应层次 PSO), AEPSO(自适应逃逸 PSO)和 APSO- I.

关键词: 粒子群算法; 适应性; 随机; 收敛
中图分类号: TP18 **文献标识码**: A

Adaptive particle swarm optimization algorithm II

LUO Ci-yong, CHEN Min-you, HAN Li
(a. State Key Laboratory of Power Transmission Equipment and System Security and New Technology, b. School of Electrical Engineering, Chongqing University, Chongqing 400044, China. Correspondent: LUO Ci-yong, E-mail: luociyong @cqu. edu. cn)

Abstract: Adaptive particle swarm optimisation-I (APSO-I) simulates the complex behaviour of social swarm and overlaps the inscrutable decision on the rational behaviour. APSO- II is proposed to overcome the poor convergence depth of APSO- I. The APSO- II algorithm divides the order action (the standard PSO) and the random exploration (the adaptive optimisation) to show the advantage of two optimisation methods. In the stage of adaptive optimisation, the optimal solution is searched in the adjacent space of the best particle. Once the optimal solution is found, the standard PSO will be applied to rapidly explore. Experimental simulations show that APSO-II algorithm is better than DPSO(Dissipation PSO), HPSO(Hierarchical PSO), AEPSO(Adaptive escape PSO), and APSO-I algorithms in the capacity of convergence speed and convergence depth.

Key words: Particle swarm optimisation(PSO); Adaptive; Random; Convergence

1 引 言

PSO(Particle swarm optimization)优化算法是由 Kennedy 和 Eberhart[1]于 1995 年提出的一种基于种群搜索的自适应进化计算技术. 算法是受到飞鸟和鱼类活动的规律性启发,用组织社会行为代替进化算法的自然选择机制,通过种群间个体的协作来实现对问题最优解的搜索.

粒子群优化算法存在早熟收敛现象,尤其是在比较复杂的多峰搜索问题中. 为了避免 PSO 早熟收敛,一些学者提出了通过控制种群的多样性来提高算法性能. Xie 等[2]提出了耗散型 PSO(DPSO),通过对微粒速度或位置引入一个小概率随机变异操作来增强种群多样性,使该算法能够有效地进行全局搜索. 但过大的变异率在增加种群多样性的同时,也将导致种群发生混乱,使种群不能进行精确的局部搜索,延缓了算法的收敛速度. Ratnaweera 等[3]提出了自适应层次 PSO(HPSO)算法,在自组织算法的基础上给出一种变异操作的自适应参数选择方式. 该算法消除了速度公式的惯性部分,其发生变异的条件是微粒速度为零,使微粒不能快速、有效地逃出局部极小点. 赫然等[4]提出了自适应逃逸 PSO(AEPSO)算法,当微粒飞行速度过小时,通过逃逸运动使微粒能够有效地进行全局和局部搜索,具有较快的收敛速度. 但该算法存在不稳定性. 孟红

收稿日期: 2008-06-10; **修回日期**: 2008-10-21.
基金项目: 重庆市自然科学基金项目(CSTC2008BB6163).
作者简介: 罗辞勇(1973—),男,安徽灵璧人,讲师,博士,从事智能控制理论及应用、阻抗成像的研究; 陈民铀(1954—),男,重庆人,教授,博士生导师,从事智能控制、数据建模等研究.

记[5]、陈如清等[6]将混沌序列混沌寻优引入粒子群优化算法中,利用混沌遍历性使得混沌优化算法跳出局部最优解.但算法平均收敛精度不高.文献[6]将整个粒子群分为 PSO 子群和混沌子群,将有序寻优和无序寻优进行分离.这一思想给本文的研究带来很大启发.

Millonas 认为,社会性的群体寻优应是在完全有序和混沌之间的平衡[7].根据这一观点,在先前工作中作者[8]提出了适应性粒子群算法 Ⅰ(APSO-Ⅰ).APSO-Ⅰ本质是在有序的决策中始终引入随机的、不可预测的决定.APSO-Ⅰ算法的优点是:在秩序和随机行为共同支配的群体行为中,随机性被降低,同时稳定性得到了提高.其缺点是收敛深度不够.造成 APSO-Ⅰ算法收敛深度不够的原因在于,持续地引入体现自适应性的随机行为,干扰了标准 PSO 的寻优目标,从而降低了标准 PSO 算法的收敛速度和深度.

在本文中,有序寻优和无序寻优被适当分离,自适应性项将被有条件地引入,并对粒子进行筛选,形成了第 Ⅱ 代适应性粒子群优化算法(APSO-Ⅱ).文中采用 4 个 Benchmark 函数进行测试,并与 HPSO,DPSO,AEPSO 和 APSO-Ⅰ算法进行对比分析.结果显示,APSO-Ⅱ具有较好的寻优性能.

2 APSO-Ⅱ算法

APSO-Ⅱ算法的思想与社会朝代变迁的模型比较接近.每个朝代经过几代发展(相对有序)后就会停滞不前,社会就会进入混乱状态.在相对无序的混乱中,自适应的社会行为会寻求一个新的优化结果.混乱最终导致一个新朝代的诞生,从而进入一个相对有序的快速发展阶段.在如此反复的历史进程中,社会不断前进并得到优化.

APSO-Ⅱ算法划分成不断交替的两个阶段.第 1 阶段从一个随机化的粒子群分布开始,利用标准 PSO 算法进行寻优.标准 PSO 算法具有运算速度快、搜索能力强、迅速陷入局部最优等特点.若判断陷入局部最优后,则切换到第 2 阶段(自适应性阶段).在最优粒子的基础上叠加自适应性项,按照指定半径在邻域空间寻优.如果发现更优粒子,则退出该阶段,再次进入标准 PSO 寻优阶段.如此反复,直到达到最大迭代次数.

2.1 第 1 阶段 ——标准 PSO 算法

Kennedy 和 Eberhart 于 1995 年提出 PSO 的最初版本.其后 Shi 和 Eberhart 引入了惯性权重对算法进行了改进[9],形成了当前的标准版本[10].在每次迭代过程中,粒子的位置由下式确定:

$$v_i(t) = wv_i(t-1) + c_1 r_1 (p - x_i(t-1)) + c_2 r_2 (p_g - x_i(t-1)), \quad (1)$$

$$x_i(t) = v_i(t) + x_i(t-1). \quad (2)$$

式中:x_i 为粒子位置;v_i 为粒子飞行速度;w 为惯性权重;c_1,c_2 为加速度;r_1,r_2 为在[0,1]范围变化的随机数;p_i 为粒子在历史中的最好位置;p_g 为整个群体中的最好位置.

设定种群规模为 N,总迭代次数为 N_t,维数为 D.用值域随机初始化粒子位置 x_i,个体最优位置 $p_i = x_i$,v_i 初始为 0,迭代计数器 $t = 0$.此阶段算法步骤如下:

1) $t = t + 1$,按式(1)和(2)更新 v_i 和 x_i.

2) 计算 x_i 适应度,更新 p_i 和 p_g.

3) 判断是否陷入局部最优,是则跳出标准 PSO 寻优,进入第 2 阶段;否则向下执行.

4) 若 $t < N_t$,则跳到步骤 1);否则整个寻优结束.

2.2 陷入局部最优进入自适应阶段的判据

标准 PSO 算法粒子总是向全局最优、个体最优飞行,经过一定次数的迭代后,可能发生全局最优、个体最优和所有粒子都相同的情况.这时按式(1)和(2)迭代,所有粒子的速度为 0,粒子的位置不再发生变化,即发生早熟收敛.将全部粒子的均方根速度作为陷入局部最优的判据似乎是理想的选择.参数选择:$w = [0.95, 0.35]$,$c_1 = [2.5, 0.5]$,$c_2 = [0.5, 2.5]$,30 维 Rosenbrock 函数,$N_t = 1000$.采用标准 PSO 算法观察全局最优适应度值和均方根速度的变化.图 1 显示,迭代至 500 代后最优适应度值不再发生变化,但均方根速度却一直不收敛为 0.这表明均方根速度不适合作为判断条件.

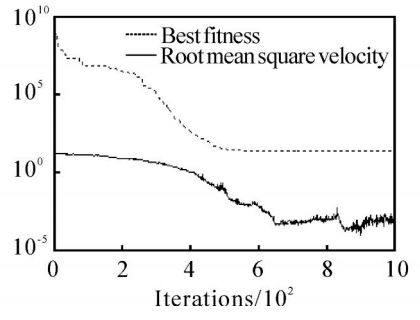

图 1 最优适应度和均方根速度

为尽快甚至提前跳出局部最优,本文中将 10 代全局历史最优适应度不变或无明显改变作为陷入局部最优的判据[6].若陷入局部最优,则进入自适应寻优阶段.

2.3 第 2 阶段 ——自适应寻优阶段

自适应寻优引入另外一群粒子 NX_i,种群规模也为 N.这一阶段的算法描述如下:

1) $t = t + 1$,按式(1)和(2)更新 v_i 和 x_i.

2) 计算 x_i 适应度,更新 p_i 和 p_g.

3) 按下式更新 NX_i 粒子各维的值:

$$NX_{id} = p_{gd} + r_3 A\beta R_{id};$$

$$A = \begin{cases} 1, & r_3 < \delta; \\ 0, & r_3 \geq \delta; \end{cases}$$

$$0 < d < D. \qquad (3)$$

式中: R_{id} 为各维值域范围; β 为搜索系数,用于调节搜索半径; r_3 为在 [0,1] 范围变化的随机数; δ 为自适应概率. 式(3)表示在全局最优粒子位置为中心的邻域空间内随机搜索.

4) 计算 NX_i 的适应度值.

5) 将 NX_i 和 x_i 共 $2N$ 个粒子按适应度值从小到大排序,选择前 N 个适应度值小的粒子形成下一次迭代粒子 x_i.

6) 如果此时 x_i 中不存在小于全局适应度值的粒子,则转步骤 7);否则更新 p_g,退出自适应寻优阶段,进入标准 PSO 算法.

7) 若 $t < N_t$,则跳到步骤 1);否则整个寻优结束.

3　性能验证

为分析 APSO-Ⅱ算法的收敛稳定性和全局搜索性能,本文选取4个Benchmark优化问题进行分析. 表 1 给出了 4 个 Benchmark 函数的定义和搜索空间,它们的理论最优解均为 0.

表 1　Benchmark 测试函数

函数	数学公式	搜索空间
Sphere	$f_1(x) = \sum_{i=1}^{D} x_i^2$	(-100,100)
Rosenbrock	$f_2(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	(-50,50)
Rastrigrin	$f_3(x) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	(-10,10)
Griewank	$f_4(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos(x_i/\sqrt{i}) + 1$	(-300,300)

选择 DPSO,HPSO,AEPSO 和 APSO-Ⅰ,与 APSO-Ⅱ算法进行性能比较. 所有实验的维数 $D = 30$,种群规模 $N = 60$,最大迭代次数 $N_t = 4000$,最大速度为 0.5,收敛因子为 0.01,循环次数为 200. 表 2 显示了各个算法的实验参数设计. 其中: DPSO,

表 2　微粒群算法参数设置

算法	w	c_1	c_2	其他参数
DPSO	0.7	1.4	1.4	$C_v = 0.001$, $C_l = 0$
HPSO	0.0	[2.5,0.5]	[0.5,2.5]	$v = 0.05$
AEPSO	0.7	1.4	1.4	$K_1 = 10$, $K_2 = 10$, $T_d = 1.0e{-}10$
APSO-Ⅰ	[0.95,0.35]	[2.5,0.5]	[0.5,2.5]	$\beta_{max} = 0.001$, $c_p = 0.05$
APSO-Ⅱ	[0.95,0.35]	[2.5,0.5]	[0.5,2.5]	$\beta = 1.0e{-}7$, $\delta = 0.1$

表 3　Benchmark 测试函数的测试结果对比

函数	算法	最优解	平均解	最劣解	最快收敛代数	平均收敛代数	收敛概率 %
f_1	DPSO	7.59e-15	8.67e-14	5.74e-13	210	238	100
	HPSO	3.42e-28	2.84e-25	1.87e-24	592	658	100
	AEPSO	7.28e-32	1.78e-29	1.03e-28	140	232	100
	APSO-Ⅰ	2.63e-17	3.56e-17	4.43e-17	992	1050	100
	APSO-Ⅱ	0	7.01e-27	6.74e-25	24	111	100
f_2	DPSO	1.05e-01	2.51e+01	7.62e+01	-	-	0
	HPSO	1.51e-01	2.06e+01	8.10e+01	-	-	0
	AEPSO	2.18e-04	8.90e+00	6.91e+01	1880	3845	16
	APSO-Ⅰ	4.11e+00	3.22e+01	6.17e+01	-	-	0
	APSO-Ⅱ	4.10e-16	4.08e-01	2.18e+01	22	367	95
f_3	DPSO	4.02e-08	4.82e-01	2.99e+00	1674	3281	62
	HPSO	2.98e+00	9.79e+00	2.29e+01	-	-	0
	AEPSO	0	3.98e-02	9.95e-01	1352	2446	96
	APSO-Ⅰ	1.19e+01	2.89e+01	5.27e+01	-	-	0
	APSO-Ⅱ	0	0	0	20	60	100
f_4	DPSO	4.22e-15	1.35e-02	7.38e-02	198	1999	54
	HPSO	1.11e-16	7.58e-03	9.53e-02	648	1921	65
	AEPSO	0	3.84e-01	8.88e-01	220	3650	10
	APSO-Ⅰ	9.60e-04	5.07e-03	3.24e-02	2086	2778	90
	APSO-Ⅱ	0	5.01e-03	1.57e-01	22	312	95

HPSO,AEPSO 和 APSO- Ⅰ算法分别选用文献[2-4,8]中的有关参数.测试结果见表 3.

对于 Sphere 函数,几种算法均得到了 100 % 的收敛概率.APSO- Ⅱ获得了最快的收敛速度,平均收敛代数为 111 代,收敛深度仅次于 AEPSO 算法.

多维 Rosenbrock 函数是公认很难极小化的病态二次函数,山谷非常多,寻优时极易陷入局部极小点,且寻找全局最小点非常困难.APSO- Ⅱ的平均收敛概率为 95 %,明显高于其他几种算法.APSO- Ⅱ的平均解较其他算法提高了 1 ～ 2 个数量级.并且最劣解当中,APSO- Ⅱ的值是最小的.

对于 Rastrigrin 函数,AEPSO 和 DPSO 取得了较好的收敛概率和收敛深度.DPSO 的收敛概率是 62 %.AEPSO 存在收敛到 0 的概率,但平均解 3.98e-2 没有小于收敛因子,其解是不可预测的.而 APSO-II 收敛概率是 100 %,平均收敛代数仅为 60 代.

对于 30 维多模态 Griewank 函数,APSO- Ⅱ算法在收敛概率、收敛速度、收敛深度以及最优解指标上均获得了最佳性能.

4 讨 论

测试结果表明,APSO- Ⅱ与 DPSO,HPSO,AEPSO 以及 APSO- Ⅰ相比,在收敛概率、收敛速度、收敛深度和最优解指标上均获得了较佳的性能.

DPSO,HPSO 和 AEPSO 均表明,通过随机操作可增强多样性.DPSO 是进行极小概率下的变异;在 DPSO 算法的基础上,HPSO 将发生变异的行为调整为速度为 0 的时候;AEPSO 则把发生变异的行为调整为速度小于不断下降的阈值.

APSO- Ⅰ是在标准 PSO 公式中,持续引入了一个体现随机行为的项.APSO- Ⅰ算法突出了非理性对寻优的贡献,但在一定程度上破坏了秩序对寻优的贡献,导致其收敛速度和深度不够理想.社会性的群体寻优应是在完全有序和混沌之间的平衡[7],而 APSO- Ⅰ未解决有序与无序之间的平衡.

APSO- Ⅱ的特点是将有序(标准 PSO)和无序(自适应寻优)进行适当分离,以发挥各自的优势.高理性解能够从低理性的环境中涌现.在自适应寻优阶段,通过在最优粒子邻域空间探寻更优的解.一旦这个解被发掘,便利用标准 PSO 的快速性,使大量粒子蜂拥而至,然后陷入局部最优.在周而复始过程中,快速性和开拓能力得到了均衡.

在本文中自适应概率 δ 选择为 0.1.如果将 δ 选为 1,则表示在最优粒子所有的维上添加随机量.进行对比实验,实验参数为 $D = 30$, $N = 60$, $N_t = 4000$, $w = [0.95,0.35]$, $c_1 = [2.5,0.5]$, $c_2 = [0.5,2.5]$, $\beta = 1.0e-7$, $\delta = 1$,最大速度 0.5,收敛因子 0.01,循环次数 200.实验结果见表 4.

表 4 测试结果(APSO- Ⅱ, $\delta = 1$)

函数	最优解	平均解	最劣解	最快收敛代数	平均收敛代数	收敛概率 %
f_1	3.87e-52	3.58e-03	3.58e-01	1034	1218	99
f_2	3.46e + 00	6.93e + 01	5.37e + 02	-	-	0
f_3	1.49e + 01	4.55e + 01	8.36e + 01	-	-	0
f_4	0	2.15e-02	1.33e + 00	848	2794	42

表 4 表明,在 $\delta = 1$ 时,APSO- Ⅱ算法的性能变得很不理想:对于 Rosenbrock 和 Rastrigrin 函数均无法收敛;与 $\delta = 0.1$ 相比,对于 Sphere 和 Griewank 函数平均收敛代数增大,收敛深度也变差.这是否意味着无序要适度呢?过大的自适应概率在增加种群多样性的同时,也将导致种群发生混乱,使种群不能进行精确的局部搜索,延缓了算法的收敛速度.

搜索系数 β 对算法的性能也产生很大的影响.β 太大,在有限时间内很难发现较优的解;β 太小,使得在最优粒子位置处的搜索能力有限.实验结果表明,对于 4 个 Benchmark 函数而言,$\beta = 1.0e-7$ 是较佳的.

Ratnaweera[3] 建议加速度线性变化,即 c_1 先大后小,而 c_2 则先小后大.目的是在搜索初期尽可能利用自己的学习知识,而在搜索后期则以社会知识为主.表 3 中的 APSO- Ⅱ算法选取 $c_1 = [2.5,0.5]$, $c_2 = [0.5,2.5]$.文献[4]等推荐 c_1 和 c_2 采用固定参数.为此进行对比实验.实验参数为 $D = 30$, $N = 60$, $N_t = 4000$, $w = [0.95,0.35]$, $c_1 = c_2 = 1.49445$, $\beta = 1.0e-7$, $\delta = 0.1$,最大速度 0.5,收敛因子 0.01,循环次数 200.实验结果见表 5.

表 5 测试结果(APSO- Ⅱ, $c_1 = c_2 = 1.49445$)

函数	最优解	平均解	最劣解	最快收敛代数	平均收敛代数	收敛概率 %
f_1	0	1.82e-17	1.82e-15	24	108	100
f_2	1.08e-13	5.10e-01	2.08e + 01	22	437	92
f_3	0	5.57e-02	3.98e + 00	18	214	97
f_4	0	3.92e-03	1.37e-01	24	242	96

表 5 表明,采用固定加速度时,算法收敛概率和收敛速度的性能,与采用线性加速度时相差不大.而对于 Rosenbrock 和 Rastrigrin 测试函数,采用固定加速度时的平均解和最劣解要差些.实验结果倾向于采用线性变化加速度.

5 结 论

APSO-I 是在有序的决策中始终引入随机的、不可预测的决定.APSO-I 使得在秩序和随机行为共同支配的群体行为中的稳定性得到提高.但是 APSO-I 算法的收敛深度和速度性能较差.

APSO-II 是将有序(标准 PSO)和无序(自适应寻优)进行适当分离,充分发挥各自的优势. 在自适应寻优阶段,通过在最优粒子邻域空间探寻更优化的解. 一旦新的优化解被发掘,便利用标准 PSO 快速寻优. 通过对 4 个 Benchmark 函数的测试表明, APSO-II 在收敛快速性和收敛深度上均优于 DPSO, HPSO, AEPSO 以及 APSO- Ⅰ,从而证明了算法的有效性.

参考文献(References)

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

[2] Xie X F, Zhang W J, Yang Z L. A dissipative particle swarm optimization [C]. Proc of the Congress on Evolutionary Computation. Honolulu, 2002: 1456-1461.

[3] Ratnaweera A, Halgamuge S K, Watson H C. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients [J]. IEEE Trans on Evolutionary Computation, 2004, 8(3): 240-255.

[4] 赫然, 王永吉, 王青, 等. 一种改进的自适应逃逸微粒群算法及实验分析[J]. 软件学报, 2005, 16(12): 2036-2044.
(He R, Wang Y J, Wang Q, et al. An improved particle swarm optimization based on self-adaptive escape velocity[J]. J of Software 2005, 16(12): 2036-2044.)

[5] 孟红记, 郑鹏, 梅国辉, 等. 基于混沌序列的粒子群优化算法[J]. 控制与决策, 2006, 21(3): 263-266.
(Meng H J, Zheng P, Mei G H, et al. Particle swarm optimization algorithm based on chaotic series [J]. Control and Decision, 2006, 21(3): 263-266.)

[6] 陈如清, 俞金寿. 混沌粒子群混合优化算法的研究与应用[J]. 系统仿真学报, 2008, 20(3): 685-688.
(Chen R Q, Yu J S. Study and application of chaos particle swarm optimization-based hybrid optimization algorithm[J]. J of System Simulation, 2008, 20(3): 685-688.)

[7] Millonas M M. Swarms, phase transitions, and collective intelligence [M]. Massachusetts: Addison Wesley Reading, 1994: 417-445.

[8] 罗辞勇, 陈民铀. 适应性粒子群优化算法[J]. 控制与决策, 2008, 23(10): 1135-1138.
(Luo C Y, Chen M Y. Adaptive particle swarm optimization[J]. Control and Decision, 2008, 23(10): 1135-1138.)

[9] Shi Y, Eberhart R. A modified particle swarm optimizer [C]. IEEE World Congress on Computation Intelligence. Anchorage, 1998: 69-73.

[10] 谢晓锋, 张文俊, 杨之廉. 微粒群算法综述[J]. 控制与决策, 2003, 18(2): 129-134.
(Xie X F, Zhang W J, Yang Z L. Overview of particle swarm optimization [J]. Control and Decision, 2003, 18(2): 129-134.)

(上接第 858 页)

[5] Yang S Y, Jiao L C. The quantum evolutionary programming [C]. Proc of the 5th Int Conf on Computational Intelligence and Multimedia Applications. Xi'an: IEEE, 2003: 362-367.

[6] 陈辉, 张家树, 张超. 实数编码混沌量子遗传算法[J]. 控制与决策, 2005, 20(11): 1300-1303.
(Chen H, Zhang J S, Zhang C. Real-coded chaotic quantum-inspired genetic algorithm [J]. Control and Desision, 2005, 20(11): 1300-1303.)

[7] 张葛祥, 李娜, 金炜东, 等. 一种新量子遗传算法及其应用[J]. 电子学报, 2004, 32(3): 476-479.
(Zhang G X, Li N, Jin W D, et al. A novel quantum genetic algorithm and its application [J]. Acta Electronica Sinic, 2004, 32(3): 476-479.)

[8] Cruz A V A, Vellasco M B R, Pacheco M A C. Quantum-inspired evolutionary algorithm for numerical optimization [C]. IEEE Congress on Evolutionary Computation. Vancouver, 2006: 2630-2637.

[9] 盛骤, 谢式千, 潘承毅. 概率论与数理统计[M]. 北京: 高等教育出版社, 2002.
(Sheng Z, Xie S Q, Pang C Y. Probability and statistics [M]. Beijing: China Higher Education Press, 2002.)

[10] 江瑞, 罗予频, 胡东成, 等. 一种协调勘探和开采的遗传算法:收敛性及性能分析[J]. 计算机学报, 2001, 24(12): 1233-1241.
(Jiang R, Luo Y P, Hu D C, et al. A genetic algorithm by coordinating exploration and exploitation: Convergence properties and performance analyse [J]. Chinese J of Computers, 2001, 24(12): 1233-1241.)

[11] Yao X, Liu Y, Lin G. Evolutionary programming made faster[J]. IEEE Trans on Evolutionary Computation, 1999, 3(2): 82-102.

[12] Tu Z, Lu Y. A robust stochastic genetic algorithm (STGA) for global numerical optimization [J]. IEEE Trans on Evolutionary Computation, 2004, 8(5): 456-470.