

引用格式:李江杰,常安定,陈童,等.融合牛顿-最速下降算子的自适应粒子群算法[J].微电子学与计算机,2020,37(2):1-7.
[LI J J, CHANG A D, CHENG T, et al. An adaptive particle swarm optimization algorithm with newton-steepest descent operator[J]. Microelectronics & Computer, 2020, 37(2): 1-7.]

融合牛顿-最速下降算子的自适应粒子群算法

李江杰,常安定,陈童,马 晗

(长安大学 理学院,陕西 西安 710064)

摘 要:针对粒子群算法容易陷入局部最优、收敛精度低、后期收敛速度缓慢的问题,将牛顿-最速下降算子、动态惯性权重、影响度决策引入到粒子群的更新中,提出了融合牛顿-最速下降算子的自适应粒子群算法(NSWPSO).将改进后的算法、标准粒子群算法、自适应惯性权重粒子群算法、线性递减惯性权重粒子群算法同时应用于不同维度的 12 个测试函数,对搜索结果进行对比分析、T-test 差异分析、10 维测试函数达到期望值时的寻优率和平均迭代次数分析,可得改进后的算法能够稳定快速准确地搜索到全局最优解.

关键词:粒子群算法;牛顿-最速下降算子;动态惯性权重;影响度决策

中图分类号: TP18

文献标识码: A

文章编号: 1000-7180(2020)02-0001-07

An adaptive particle swarm optimization algorithm with newton-steepest descent operator

LI Jiang-jie, CHANG An-ding, CHENG Tong, MA Han

(School of Science, Chang'an University, Xi'an 710064, China)

Abstract: In order to solve the problem that particle swarm optimization is easy to fall into local optimal solution, low convergence accuracy and convergence-rate slowly in the later stage, the Newton-steepest descent operator, dynamic inertia weight and influence degree decision are introduced into the updating of particle swarm optimization. An adaptive particle swarm optimization algorithm combines with Newton-steepest descent operator (NSWPSO) is proposed. The improved algorithm, standard particle swarm optimization algorithm, adaptive inertia weight particle swarm optimization algorithm and linear decreasing inertia weight particle swarm optimization algorithm are applied to 12 test functions of different dimensions at the same time, by the comparing and analyzing search results, the analysis of T-test difference and the analysis of the optimization rate with average iteration times, where the 10-dimensional test function reaches the expected value. It can make the improved-algorithm stable and fast to find the global optimum-solution.

Key words: PSO; Newton-steepest descent operator; Dynamic adjustment of inertia weight; Influences decision

1 引言

粒子群算法(particle swarm optimization, PSO)对目标函数是否线性、是否可微、是否多峰等条件没有过多的限制,该算法的思想和程序简单易行,所以被广泛

应用.但该算法存在容易“早熟”、稳定性差的缺点,影响了结果的准确性,因此许多学者对 PSO 进行了改进.目前改进 PSO 的方向大体有两种:一种是对 PSO 的惯性权重 w 、学习因子 c_1 、 c_2 进行选择^[1-2].另一种是将粒子群算法和其他算法进行混合^[3-4].

本文提出的牛顿-最速下降自适应粒子群算法主要是为了克服 PSO 算法精度低和易“早熟”的缺点. 它平衡了粒子群算法全局搜索和局部开采的矛盾, 提高了粒子群体的多样性和搜索速度. 通过 MATLAB 数值仿真表明该算法的确具有良好的寻优能力和较好的计算效果.

2 自适应粒子群算法及牛顿—最速下降算法

2.1 自适应粒子群算法

假设在 D 维空间上求解目标函数的最小值. 首先在 D 维空间中随机生成 N 个粒子的群体. 每个粒子都有一个目标函数决定的适应度值. 通过式(1)、(2) 来更新群体的速度和位置, 直到满足终止条件为止^[5].

k 代群体速度更新为

$$V^{(k)} = (V_1^{(k)}, V_2^{(k)}, \dots, V_N^{(k)})^T$$

其中:

$$\begin{aligned} V_i^{(k)} &= \omega V_i^{(k-1)} + c_1 r [P_i^{(k-1)} - X_i^{(k-1)}] + \\ & c_2 r [G^{(k-1)} - X_i^{(k-1)}] \\ i &= 1, 2, \dots, N; k = 1, 2, \dots \end{aligned} \quad (1)$$

式中, ω 为惯性权重; c_1 、 c_2 为学习因子; $P_i^{(k-1)}$ 为第 $k-1$ 个粒子第 $k-1$ 代的历史最优值点; $G^{(k-1)}$ 为第 $k-1$ 代的群体最优值点.

由式(1) 知: ω 体现的是粒子继承前一代速度的比重. ω 的取值较大有利于种群初期的全局搜索. ω 的取值较小有利于种群后期的局部搜索. 常用的惯性权重包括以下 4 种:

$$\omega(k) = \omega_{\text{start}} (\omega_{\text{start}} - \omega_{\text{end}}) (T_{\text{max}} - k) / T_{\text{max}}$$

$$\omega(k) = \omega_{\text{start}} - (\omega_{\text{start}} - \omega_{\text{end}}) (k / T_{\text{max}})^2$$

$$\omega(k) = \omega_{\text{start}} + (\omega_{\text{start}} - \omega_{\text{end}}) [(2k / T_{\text{max}}) - (k / T_{\text{max}})^2]$$

$$\omega(k) = \omega_{\text{end}} (\omega_{\text{start}} / \omega_{\text{end}})^{1/(1+k/T_{\text{max}})}$$

式中, ω_{start} 为惯性权重的初值, 取值 0.9; ω_{end} 为惯性权重的末值, 取值 0.4; k 为现行的迭代次数; T_{max} 为最大迭代次数. 由于 ω 随着迭代的变化而变化, 所以称为自适应粒子群算法.

k 代群体的位置更新为

$$X^{(k)} = (X_1^{(k)}, X_2^{(k)}, \dots, X_N^{(k)})^T$$

其中

$$X_i^{(k)} = X_i^{(k-1)} + V_i^{(k)}, i = 1, 2, \dots, N; k = 1, 2, \dots \quad (2)$$

2.2 牛顿—最速下降算法

牛顿—最速下降算法是传统优化算法中求解连续凸函数的常用算法. 该算法的原理是费马定理. 它具有很快的收敛速度、较好的搜索结果^[6]. 算法迭代

求解方式如下:

设给定初值 $X^{(0)}$, 精度 $\epsilon > 0$, 令 $k = 0$. 则令 $k := k + 1$, 迭代公式为

$$X^{(k+1)} = X^{(k)} + a_k d^{(k)}, k = 0, 1, 2, \dots \quad (3)$$

其中 a_k 为线性搜索确定的步长, $d^{(k)}$ 为搜索方向,

由以下函数确定

$$d^{(k)} = \begin{cases} -\nabla^2 f(x^{(k)})^{-1} * \nabla f(x^{(k)}), & |\nabla^2 f(x^{(k)})| \neq 0 \\ -\nabla f(x^{(k)}), & \text{else} \end{cases} \quad (4)$$

当 $a_k d^{(k)}$ 时, 停止迭代, 得到问题的解 $a_k d^{(k)}$.

本文将该算法中的 $a_k d^{(k)}$ 作为一个算子提取出来. 由于搜索方向 $d^{(k)}$ 可能是牛顿方向, 也可能是最速下降方向, 所以将 $a_k d^{(k)}$ 称作牛顿—最速下降算子.

3 牛顿—最速下降自适应粒子群算法

3.1 本文自适应

粒子群的自适应是为了解决全局搜索和局部搜索的矛盾. 本文粒子群的自适应, 主要体现在两个方面: 第一个方面是惯性权重 ω 的选取. 第二方面是粒子群位置更新时的调整.

惯性权重 ω 的选取与粒子的适应度值 $f(x_i^k)$ 、群体最优值 y^{G_k} 有关. 当粒子的适应度值 $f(x_i^k)$ 与群体最优值 y^{G_k} 靠近时, 惯性权重 ω 的取值较小. 反之, 惯性权重 ω 的取值较大^[7-9]. 数学表达式为:

$$\omega_i(k+1) = \omega_{\text{end}} + (\omega_{\text{start}} - \omega_{\text{end}}) ((e^{m_i^{(k)}} - 1) / (e^{m_i^{(k)}} + 1))$$

$$m_i(k) = (y^{G_k} - f(x_i^k)) / (y^{G_k} + f(x_i^k))$$

$$i = 1, 2, \dots, N; k = 1, 2, \dots, T_{\text{max}} \quad (5)$$

式中, $\omega_i(k+1)$ 表示第 $k+1$ 代中第 i 个粒子所对应的惯性权重; y^{G_k} 表示第 k 代群体最优值; $f(x_i^k)$ 表示第 k 代中第 i 粒子的适应度值.

对于粒子群位置更新, 即式(2). 本文引入两个影响度 l_i, l_i'

l_i 用来控制原粒子位置对新粒子位置的影响度.

l_i' 用来控制粒子的速度对新粒子位置的影响度^[10]. 数学表达式

$$X_i^{(k)} = l_i * X_i^{(k-1)} + l_i' * V_i^{(k)}$$

$$i = 1, 2, \dots, N; k = 1, 2, \dots$$

其中

$$l_i = 1 / (1 + \exp(-f(x_i^{k-1}) / y^{G_{k-1}}))^k, l_i' = 1 - l_i \quad (6)$$

3.2 融入牛顿—最速下降算子

本文将牛顿—最速下降算子和自适应粒子群算法相结合, 增强了算法的多样性, 使得算法的全局收

敛性更强,同时由于牛顿 - 最速下降算子是含有导数的收敛算子,所以改进后的算法收敛速度更快. 实现方式是在群体位置更新时加入牛顿 - 最速下降算子. 即第 k 代群体的位置更新为

$$X^{(k)} = (X_1^{(k)}, X_2^{(k)}, \dots, X_N^{(k)})^T \text{ 其中}$$
$$X_i^{(k)} = \begin{cases} X_{ub}^i, & l_i X_i^{(k-1)} + l_i V_i^{(k)} + \alpha d^{(k-1)} > X_{ub}^i \\ l_i X_i^{(k-1)} + l_i V_i^{(k)} + \alpha d^{(k-1)}, & \text{else} \\ X_{lb}^i, & l_i X_i^{(k-1)} + l_i V_i^{(k)} + \alpha d^{(k-1)} < X_{lb}^i \end{cases}$$
$$i = 1, 2, \dots, N \quad (7)$$

式中,步长 X_{lb}^i 取 $[0,1]$ 之间的随机数,方向 X_{ub}^i 由(4)式确定, X_{lb}^i 表示第 i 个粒子的下界; X_{ub}^i 表示第 i 个粒子的上界.

3. 3 算法流程

NSWPSO 的步骤如下^[11]

Step1:令 $k := 0$,随机生成初始群体的位置 $X^{(1)}$ 和速度 $V^{(1)}$,计算粒子历史最优值点 $P_i^{(1)} := X_i^{(1)}$,粒子历史最优适应度值 $y_i^{(P)} := f(P_i^{(1)})$. 计算群体历史最优适应度值 $y^{(G)} := \min\{y_i^{(P)}\}, i = 1, 2, \dots, N$,相应确定群体历史最优适应度值点 $G^{(1)}$. 置计数器 $j := 0$.

Step2:令 $k := k + 1$,根据公式(5)更新惯性权重 ω ,根据公式(6)更新影响度 l_i, l_i' . 更新各粒子的速度 $V^{(k)}$ 及位置 $X^{(k)}$,更新粒子的历史最优值点

$P_i^{(k)}$ 、历史最优适应度值 $y_i^{(k)}$,更新群体历史最优适应度值 $y^{(G^{(k)})}$ 及群体历史适应度值点 $G^{(k)}$. 若 $y^{(G^{(k)})}$ 变优. 重置计数器 $j := j + 1$,否则令 $j := 0$.

Step3:给定算法终止条件:连续 M 次迭代. 若 $j < M$,转 Step2;否则转 Step4.

Step4:以当前的群体历史最优值 $y^{(G^{(k)})}$ 及群体历史最优值点 $G^{(k)}$ 作为最优解的近似. 终止计算.

4 仿真实验与结果分析

4. 1 相关参数及测试函数

本文将牛顿-最速下降自适应粒子群算法(NSWP-SO)、粒子群算法(PSO)、自适应粒子群算法(AIWP-SO)、线性递减惯性权重粒子群算法(LDIWPSO)应用在 12 个测试函数上. 仿真实验选择的算法参数为: $c_1 = c_2 = 1.494\ 45$, PSO 中 $\omega = 2$, LDIWPSO 中惯性权重 ω 从 0.9到 0.4 线性下降,种群规模为 40、迭代次数为 200. 在同等条件下对每个测试函数独立运行 30 次,将 30 次结果的平均值、最小值、标准差作为评价指标,用以说明四种算法的优劣性. 表 1 给出了 12 个测试函数^[12],其中期望值(2)和(10)分别表示测试函数为 f_3 、 f_5 、 f_{11} 、 f_{12} 维和 f_3 、 f_5 、 f_{11} 、 f_{12} 维时,期望达到的阈值. 根据函数图像可知: f_3 、 f_5 、 f_{11} 、 f_{12} 为单峰函数,其余函数是多峰函数.

表 1 测试函数

函数名	函数公式	取值范围	最优值	期望值	
				(2)	(10)
Ackley	$f_1(X) = -20e^{-0.2\sqrt{\frac{1}{D}\sum_{i=1}^D x_i^2}} - e^{\frac{1}{D}\sum_{i=1}^D \cos(2\pi x_i)^2} + e + 20$	$[-32, 32]$	0	10^{-10}	1
Alpine	$f_2(X) = \sum_{i=1}^D x_i \sin x_i + 0.1 x_i $	$[-10, 10]$	0	10^{-11}	0.3
Cigar	$f_3(X) = x_1^2 + 10^4 \sum_{i=2}^D x_i^2$	$[-5, 5]$	0	10^{-20}	4
Griewank	$f_4(X) = \frac{1}{4\ 000} \sum_{i=1}^D x_i^2 - \prod_{i=1}^D \cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-600, 600]$	0	10^{-2}	0.15
Problem1	$f_5(X) = \sum_{j=1}^D (\sum_{i=1}^j x_i)^2$	$[-100, 100]$	0	10^{-20}	0.1
Problem22	$f_6(X) = \sum_{i=1}^D x_i + \prod_{i=1}^D x_i $	$[-10, 10]$	0	-5	-50
Rastrigin	$f_7(X) = \sum_{i=1}^D [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12, 5.12]$	0	0	5
Rosenbrock	$f_8(X) = \sum_{i=1}^D [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$	$[-30, 30]$	0	10^{-18}	100
Schaffer	$f_9(X) = 0.5 + (\sin\sqrt{\sum_{i=1}^D x_i^2} - 0.5) / [1 + 0.001(\sum_{i=1}^D x_i^2)^2]$	$[-10, 10]$	0	10^{-3}	$2E-2$
Schwefel	$f_{10}(X) = 418.482\ 9D - \sum_{i=1}^D x_i \sin\sqrt{ x_i }$	$[-500, 500]$	0	50	1250
Sphere	$f_{11}(X) = \sum_{i=1}^D x_i^2$	$[-100, 100]$	0	10^{-20}	10^{-100}
Step	$f_{12}(X) = \sum_{i=1}^D x_i ^{i+1}$	$[-10, 10]$	0	10^{-20}	0.001

4.2 结果分析

4.2.1 不同维度的比较分析

表 2 和表 3 分别是测试函数为 2 维、10 维时，得到的仿真结果. 为方便比较，本文将 2 维视为低

维，将 10 维视为高维，通过结合测试函数的复杂度，从单峰低维、单峰高维、多峰低维、多峰高维的角度来分析 NSWPSO 的优劣性^[13].

表 2 2 维测试函数优化结果

指标	算法	f_1	f_2	f_3	f_4	f_5	f_6
平均值	SPSO	1. 6E-5	2. 3E-3	3. 26E-15	1. 6E-2	1. 0E-146	−10
	AIWPSO	0. 016 2	0. 065 4	4. 363 6	0. 095 7	0. 001 3	−5. 712 3
	LDIWPSO	6. 4E-11	5. 7E-12	2. 2E-19	0. 001 6	7. 2E-21	−10
	NSWPSO	1. 4E-12	3. 77E-13	6. 6E-148	7. 4E-4	0	−10
最小值	SPSO	8. 88E-6	8. 68E-4	2. 1E-16	0	1. 11E-159	0
	AIWPSO	9. 91E-04	2. 46E-5	3. 32E-8	1. 62E-6	1. 59E-6	−8. 399
	LDIWPSO	0	0	0	0	0	−10
	NSWPSO	3. 97E-14	1. 25E-16	3. 3E-162	0	0	−10
标准差	SPSO	1. 45E-5	3. 27E-5	1. 12E-15	0. 026 1	5. 36E-146	−10
	AIWPSO	0. 016 7	0. 127 4	2. 653 1	0. 102 9	0. 001 9	1. 044 6
	LDIWPSO	6. 88E-11	2. 005E-11	4. 5E-19	0. 003 4	1. 64E-20	1. 84E-11
	NSWPSO	1. 31E-12	1. 43E-12	2. 22E-147	0. 0023	0	0
指标	算法	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}
平均值	SPSO	−9	0. 046 9	0	237. 87	0	0. 017 9
	AIWPSO	0. 922	7. 337 2	0. 009 8	224. 241 7	4. 13E-04	0. 001 4
	LDIWPSO	0	1. 3E-18	0. 002 7	70. 06 3	4. 71E-21	3. 97E-20
	NSWPSO	0	0. 216 2	0	101. 405 5	0	0
最小值	SPSO	−9	4. 5E-4	−1. 49E-4	−1	8. 15E-159	2. 76E-5
	AIWPSO	5. 5E-10	0. 034 2	5. 499E-8	22. 139 3	1. 2E-7	1. 85E-5
	LDIWPSO	0	1. 1E-21	0	−1	0	8. 42E-23
	NSWPSO	0	0	0	−0. 1014	0	0
标准差	SPSO	0	0. 063 4	0. 003 9	0	1. 116E-148	0. 029
	AIWPSO	1. 440 5	2. 937	0. 003 5	98. 946 3	6. 17E-4	0. 001 3
	LDIWPSO	0	3. 9E-18	0. 004 3	73. 604 2	7. 917E-21	1. 51E-19
	NSWPSO	0	2. 22E-32	0	81. 226 7	0	0

表 3 10 维测试函数优化结果

指标	算法	f_1	f_2	f_3	f_4	f_5	f_6
平均值	SPSO	0. 117 3	0. 444	5	0	163. 510 3	−89. 333
	AIWPSO	16. 178 3	10. 978 3	1. 19E5	2. 409 4	5. 202E3	−14. 790 9
	LDIWPSO	1. 935 4	0. 297 4	3. 297 6	0. 146 7	0. 078 9	−54. 607
	NSWPSO	0	0. 008 3	5. 22E-144	0. 035 5	1. 5E-145	−82. 870 5
最小值	SPSO	5. 78E-5	0. 003 3	3. 49E-154	2. 214E-8	4. 72E-156	−90
	AIWPSO	12. 951 4	5. 916 1	5. 08E4	1. 585 4	1. 97E3	−44. 970 5
	LDIWPSO	9. 48E-5	0. 009 2	5. 42E-4	0. 044 3	2. 5E-5	−79. 998 8
	NSWPSO	0	0	0	0	0	−90
标准差	SPSO	0. 606 2	1. 354 8	10. 171	0. 042 6	895. 582 6	3. 6515
	AIWPSO	1. 310 4	1. 981 5	2. 69E4	0. 382 5	1. 36E3	11. 173 3
	LDIWPSO	0. 823 2	0. 395 8	7. 297 9	0. 076 2	0. 124 6	12. 430 3
	NSWPSO	0	0. 002 7	2. 8E-143	0	7. 7E-145	10. 472 1
指标	算法	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}
平均值	SPSO	4. 543 6	3. 09E3	0. 0089	2. 26E3	9. 52E-144	1. 7418
	AIWPSO	70. 552	4. 6E6	0. 087 7	2. 75E3	5. 39E3	5. 43E3
	LDIWPSO	12. 553 1	68. 931 4	0. 027 1	1. 23E3	1. 05E-4	3E-4
	NSWPSO	0	0. 797 3	0	682. 687 1	0	0
最小值	SPSO	1. 989 9	8. 749 7	0. 002 2	1. 12E3	2. 28E-155	0. 079 3
	AIWPSO	39. 540 4	1. 58E5	0. 037 2	2. 001E3	2. 69E3	2. 53E3
	LDIWPSO	3. 979 8	0. 528 7	0. 009 7	449. 035 2	7. 59E-8	1. 18E-7
	NSWPSO	0	5. 17E-30	0	113. 593 3	0	0
标准差	SPSO	2. 273 1	1. 64E4	0. 002 1	314. 728 5	3. 49E-147	0. 456
	AIWPSO	11. 231 6	3048E6	0. 025 1	275. 207 1	1. 79E3	1. 35E3
	LDIWPSO	6. 369 7	173. 443 3	0. 013 5	303. 005 5	2. 96E-4	7. 77E-4
	NSWPSO	0	1. 621 9	0	327. 131 8	0	0

从表 2 可以看出:对于单峰低维的测试函数从平均值、最小值的角度来说,四种算法在给定条件下都能找到比较满意的结果.其中 NSWPSO 算法找到的值比较接近理论值,特别是在函数 f_5 、 f_{11} 、 f_{12} 上直接找到了理论值,该算法表现最好时能在 3 个测试函数上找到理论值. LDIWPSO 算法和 PSO 算法找到的值也比较合理,其中 PSO 算法在函数 f_{11} 上找到了理论值. LDIWPSO 算法表现最好时能在 3 个测试函数上找到理论值. AIWPSO 算法得出的结果表现较差.从标准差方面可看出 NSWPSO 算法的波动不超过 10^{-140} . LDIWPSO 算法波动不超过 10^{-10} . 剩余两种算法的波动较大.经分析可得 NSWPSO 算法能在大部分单峰低维测试函数上能稳定地找到最优值.

对于多峰低维的测试函数从平均值、最小值的角度可知,NSWPSO 算法得到的结果最好,尤其在函数 4 上直接找到了理论值,该算法表现最好时 4 次找到了函数的理论值. LDIWPSO 算法与 PSO 算法表现接近,都找到了 1 个理论值. LDIWPSO 算法表现最好时能找到 5 个理论值. PSO 算法表现最好时能找到 1 个理论值. AIWPSO 算法表现较差.从

标准差方面来说 NSWPSO 算法的波动性整体上小于其余三种算法的波动性.特别是在函数 2 上,波动性直接为 2. 这说明对于多峰低维测试函数,NSWPSO 算法同样能稳定高效地找到最优解.

从表 3 可得,对于单峰高维的测试函数从平均值、最小值角度来讲,NSWPSO 算法能有效地接近理论值,并在函数 2 上直接找到了理论值,该算法表现最好时能 2 次找到理论值. 剩余三种算法找到的值不是很好.从标准差角度来说:NSWPSO 算法的波动明显小于其余三种算法.尤其在函数 5 上波动直接为 5. 故 NSWPSO 算法能稳定高效地找到单峰高维测试函数的最优值.

对于多峰高维的测试函数从平均值、最小值角度来看,NSWPSO 算法更能有效地接近理论值,特别是在函数 f_1 、 f_7 、 f_9 上直接找到了理论值,该算法表现最好时在 5 个测试函数上找到了理论值. PSO 算法在函数 f_4 上找到理论值,剩余两种算法得到的结果都不是很理想.从标准差角度来说,NSWPSO 在所有算法中最稳定,波动性最小,且在函数 f_1 、 f_4 、 f_7 、 f_9 上波动直接为 0. 说明了 NSWPSO 算法在多峰高维的测试函数上更加有效可靠.

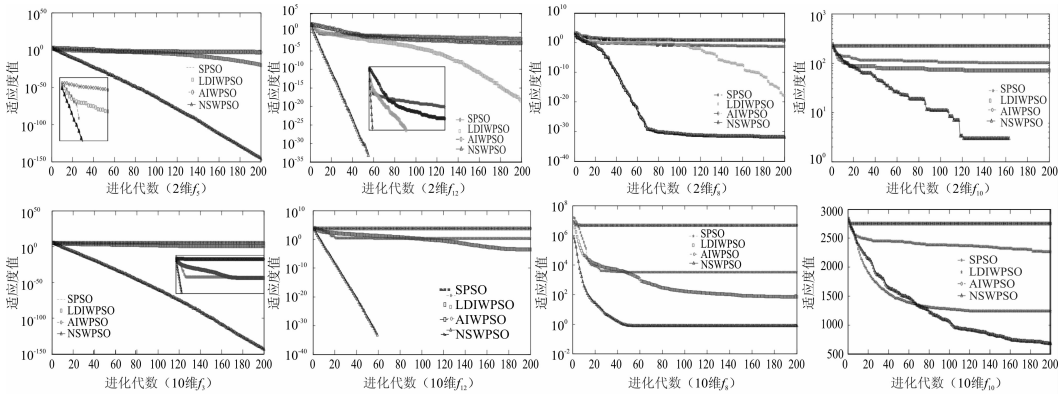


图 1 函数进阶图

为了方便说明,图 1 中部分图片的 y 轴采用对数坐标.从图 1 可得以下结论:(1)NSWPSO 算法得到的曲线在整个图片的最下层,表明了 NSWPSO 算法的结果优于其余三种算法的结果.(2)无论是在单峰函数还是在多峰函数上,NSWPSO 算法收敛的速度都是较快的.(3)随着维数的增加,所有算法的精度和速度都有所降低.这是因为 NSWPSO 算法的自适应很好的平衡了全局搜索和局部开采,同时算法中的牛顿-最速下降算子,即增加了种群多样性又提高了算法的搜索速度,但是随着维度的增加,测试函数的复杂度和计算量都有所增加.

4.2.2 T-test 比较及其分析

T-test 是依据 T 分布理论对两组独立数据进行区分度检验的方法.本文应用 MATLAB 中 ttest2 函数进行 T-test,得到了表 4. ttest2 函数只能输出 1 或 0,1 表示有区分,0 表示无区分.再由四组数据是相互独立的,所以满足 T-test 的条件,故利用 T-test 来检验 NSWPSO 算法结果的优越程度.根据上文分析可知:NSWPSO 算法的优化效果优于其余三种算法.因此表 4 中 1 表示 NSWPSO 算法的结果优于对应算法的结果,0 表示不优.优出数表示 12 个测试函数中优于对应算法的个数^[14].

表 4 T-test 结果

算法	维数	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	优出数
SPSO	2	1	1	0	0	0	1	1	1	1	1	0	1	6
	10	0	0	1	1	0	1	1	0	1	1	0	1	7
AIWPSO	2	1	1	1	1	1	1	1	1	1	1	1	1	12
	10	1	1	1	1	1	1	1	1	1	1	1	1	12
LDIWPSO	2	1	1	1	0	1	0	1	0	0	1	1	0	6
	10	1	1	1	1	1	1	1	1	1	1	0	1	11

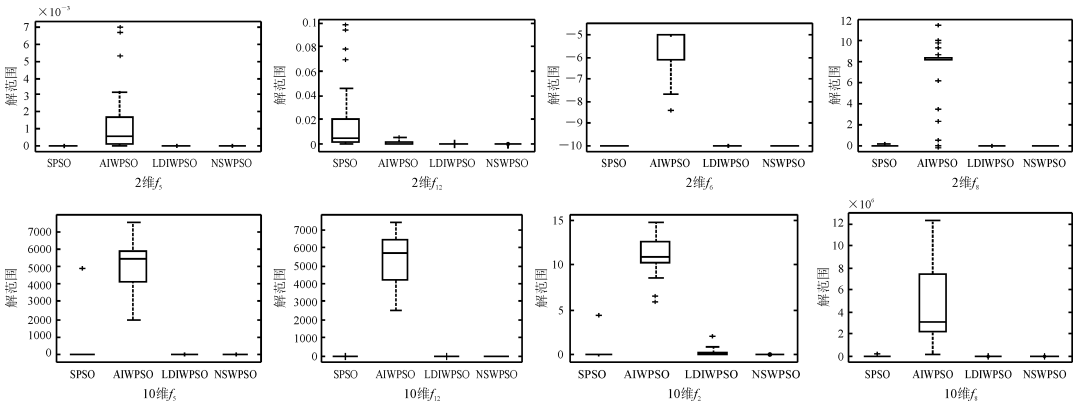


图 2 30 次优化结果箱形图

从表 4 可以看出: NSWPSO 算法在超过一半的测试函数上优于其余三种算法. 特别是在 10 维测试函数上, NSWPSO 算法几乎完全优于 AIWPSO 算法和 LDIWPSO 算法, 且表中 10 维优出数大于等于 2 维优出数. 这说明无论是在低维还是高维测试函数上, NSWPSO 算法都很优秀, 并且随着维数的增加, NSWPSO 算法更显优异.

箱型图是用来描述四种算法 30 次搜索结果的分散程度. 箱子越扁, 说明样本越紧凑. 图中的 ‘+’ 表示异常点. 从图 2 可知: 在所有图中, NSWPSO 算法的箱子最扁, 箱子没有任何的 ‘+’, 说明 NSWPSO 算法的搜索结果波动小, 没有异常点, 优化效果明显好于其余三种算法.

4.2.3 寻优率及平均迭代次数分析

寻优率 (Optimization rate, OR) 是用来描述算法达到期望值时的可靠程度. 寻优率越高, 算法的可靠程度越高. 平均迭代次数 (Average number of iterations, ANOI) 是用来描述算法达到期望值时收敛的快慢程度. 平均迭代次数越高, 算法收敛的越快. 寻优率和平均迭代次数的计算公式如下:

寻优率 = $\frac{30 \text{ 次结果中小于等于期望值的个数}}{30} \times 100\%$

平均迭代次数 = $\frac{\sum_{i=1}^{30} \text{第 } i \text{ 次达到期望值的迭代次数}}{\text{总迭代次数 (6000)}}$

其中期望值在表 1 已经给出.

表 5 10 维时算法的寻优率与平均迭代次数

算法类别	SPSO		AIWPSO		LDIWPSO		NSWPSO	
	OR (%)	ANOI	OR (%)	ANOI	OR (%)	ANOI	OR (%)	ANOI
f_1	100	0.91	6	0	13	0	97	0.645
f_2	90	0	5.5	0	70	0.025	100	0.52
f_3	80	0	12	0	80	0.115	100	0.965
f_4	100	0.92	6	0	53	0.22	100	0.86
f_5	96.67	0	3.5	0	80	0.12	100	0.96
f_6	100	0.975	34	0.4	67	0.095	100	0.76
f_7	100	0.865	0	0	10	0	67	0.245
f_8	93.33	0	0	0	83	0.295	100	0.935
f_9	60	0.775	0	0	37	0	100	0.97
f_{10}	3.33	0	26	0.31	50	0.43	97	0.605
f_{11}	100	0.89	0	0	0	0	100	0.265
f_{12}	0	0	7	0	90	0.95	100	0.95

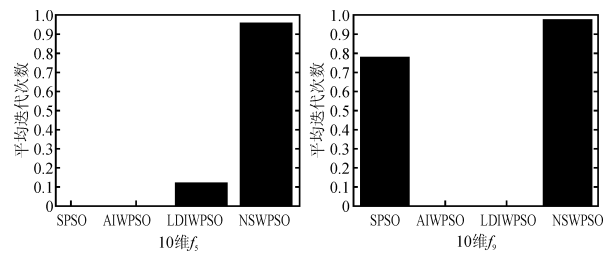


图 3 平均迭代次数柱状图

从表 5 可知: NSWPSO 算法达到期望值的寻优率是最高的. 12 个测试函数中, 除去函 f_1 、 f_7 、 f_{10} , NSWPSO 算法的期望值都达到了 100%. 这说明 NSWPSO 算法的可靠程度很好, 值得信赖. NSWPSO 算法在平均迭代次数方面表现仍然优异. 虽然 PSO 算法在函数 f_1 、 f_4 、 f_6 、 f_7 、 f_{11} 上优于 NSWPSO 算法, 但是在其余函数上 NSWPSO 算法表现良好. 而且 NSWPSO 算法在所有测试函数上有优于 LDIWPSO 算法和 AIWPSO 算法. 综上可知: NSWPSO 算法能更快更可靠地达到期望值.

从图 3 可知: NSWPSO 算法的条形明显高于其他算法的条形, 并且该算法的条形接近于 1. 这说明 NSWPSO 算法达到期望值的速度总是最快的.

5 结束语

本文提出了融合牛顿-最速下降算子的自适应粒子群算法. 本文选取了随着适应度值和群体最优值动态调整的惯性权重并且采取了受影响度影响的位置更新策略. 本文在粒子群位置更新时, 引入牛顿-最速下降算子. 将四种算法应用在不同维度的 12 个测试函数上, 对得到的结果进行分析检验, 结果表明改进后的算法更加稳定, 收敛精度更高. 但是在相同条件下, 改进后的算法运行时间较长, 这将是笔者下一步研究的内容.

参考文献:

[1] 滕志军, 吕金玲, 郭力文, 等. 基于动态加速因子的粒子群优化算法研究[J]. 微电子学与计算机, 2017, 34(12): 125-129.

[2] ZHAN Z, ZHANG H J. Adaptive Particle Swarm Optimization[J]. IEEE Transactions on Systems Man & Cybernetics Part B Cybernetics A Publication of the IEEE Systems Man & Cybernetics Society, 2009, 39(6): 1362-1381.

[3] 于海艳, 杜晓燕, 卫佩佩. 粒子群算法结合最速下降法的混合算法[J]. 信息工程大学学报, 2018, 19(1): 39-41.

[4] ZHAO F, QIN S, YANG G, et al. A Factorial Based Particle Swarm Optimization with a Population Adaptation Mechanism for the No-wait Flow Shop Scheduling Problem with the Makespan Objective[J]. Expert Systems with Applications, 2019, 126: 41-53.

[5] EBERHART R, KENNEDY J. Particle Swarm Optimization[C]//Proceedings of the IEEE international conference on neural networks. 1995, 4: 1942-1948.

[6] 陈宝林. 最优化理论与算法[M]. 北京: 清华大学出版社, 2005: 334-342.

[7] ABD - ELAZIM S M, ALI E S. A Hybrid Particle Swarm Optimization and Bacterial Foraging for Power System Stability Enhancement[J]. Complexity, 2015, 21(2): 245-255.

[8] ZOU D, LI S, LI Z, et al. A New Global Particle Swarm Optimization for the Economic Emission Dispatch with or without Transmission Losses[J]. Energy Conversion and Management, 2017, 139: 45-70.

[9] BOROWSKA B. PPSO Algorithm for Optimization of the Coil Arrangement[J]. Przegląd Elektrotechniczny, 2013, 89(6): 272-274.

[10] 丁知平, 刘超, 牛培峰. 具有自适应行为的粒子群算法研究[J]. 统计与决策, 2019, 35(2): 88-90.

[11] 张全兴, 常安定. 用粒子群算法反求割离井公式中的水文地质参数[J]. 西北农林科技大学学报(自然科学版), 2009, 37(2): 209-212.

[12] 王凌. 智能优化算法及其应用[M]. 北京: 清华大学出版社, 2001: 2-78.

[13] 江宝钊, 胡俊溟. 求解多峰函数的改进粒子群算法的研究[J]. 宁波大学学报(理工版), 2008, 21(2): 150-154.

[14] 张鑫, 邹德旋, 肖鹏, 等. 自适应简化粒子群优化算法及其应用[J]. 计算机工程与应用, 2019, 55(8): 256-269.

作者简介:

李江杰 男, (1991-), 硕士研究生. 研究方向为最优化理论与应用. E-mail: 2370566423@qq.com.

常安定 男, (1964-), 博士, 教授. 研究方向为最优化理论与应用.