基于改进小生境粒子群算法的多模函数优化

史哲文, 白雪石, 郭 禾
(大连理工大学 软件学院 嵌入式系统工程系, 辽宁 大连 116620)

摘 要: 在基于粒子群算法的多模优化问题中,针对现存小生境方法需要特定参数的缺陷,提出了一种不需要参数的小生境算法。该算法通过粒子适应度在种群适应度中所占比例以及粒子之间的欧式距离两方面因素确定粒子的局部最优解,并通过每轮迭代中每个局部最优解粒子和以它作为局部最优解的普通粒子的欧式距离的平均值确定出该小生境的半径。在几个广泛的测试函数上的实验结果表明,该算法在收敛速度和成功率方面比需要小生境参数的算法(FERPSO、SPSO)更优秀。

关键词: 小生境; 粒子群优化; 多模函数; 适应度; 欧氏距离
中图分类号: TP18 **文献标志码**: A **文章编号**: 1001-3695(2012)02-0465-04
doi:10.3969/j.issn.1001-3695.2012.02.016

Multimodal function optimization based on modified niching particle swarm optimization

SHI Zhe-wen, BAI Xue-shi, GUO He
(*Dept. of Embedded System Engineering, School of Software Technology, Dalian University of Technology, Dalian Liaoning* 116620, *China*)

Abstract: With respect to the multimodal problem depending on particles' swarm optimizer, this paper proposed a niching algorithm on the basis of the demerit that many existing niching algorithms need parameters. This algorithm located particles' local optima depending on two aspects, the ratio of a particle's fitness and swarm's fitness and the Euclidean-distance among them. Alternatively, the radius of each niche was based on the distance between the local optima and other common swarms that regard that swarm as local optima. Experimental results for several wide test functions show that the algorithm proposed performs better than those algorithms(FERPSO, SPSO) which need niching parameters on converging speed and success rate.
Key words: niche; particle swarm optimizer(PSO); multimodal function; fitness; Euclidean-distance

0 引言

粒子群优化(PSO)算法作为一种随机优化算法,已有许多研究者将其应用于解决实际优化问题,研究表明该算法在解决复杂优化问题中表现出很好的鲁棒性[1]。现在粒子群算法不仅用于搜索有单个极值点的优化问题,而且更多地用来解决多模优化问题[25]。随着越来越多搜索策略的提出,基于粒子群算法的多模优化问题已经成为研究热点。

这种解决多模优化问题的搜索策略就称为小生境技术。有研究者将粒子群算法引入到小生境技术, NichePSO[3]、SPSO[4,5]等都是有代表性的算法。其中 NichePSO 可以保证收敛于局部最优解的粒子建立子种群并继续向全局最优解移动,缺点是算法没有考虑小生境个数与子种群个数的关系,并且对于参数的变化比较敏感;SPSO 对于低维的测试函数具有较高的成功率,维数增大后算法难以收敛,而且种群半径需要提前设定。

针对现存的基于粒子群算法的小生境技术,大部分需要设定一些基于先验知识的参数问题,本文提出一种基于粒子群优化的不需要参数的小生境算法,确定小生境的策略是先找到以同一个粒子作为局部极值点(lbest)的普通粒子,这些与 lbest 粒子的欧氏距离的平均值就是这个小生境的半径,与 lbest 粒子的欧氏距离小于或者等于这个半径的粒子就是这个小生境的粒子,其他的粒子排除在这个小生境之外。粒子运动的原则不仅取决于自己的历史最佳位置,还取决于周边适应度较好且距离较近的粒子位置,这样经过多次迭代,可以保证粒子向全局极值点不断靠近,最终收敛。

实验结果证明,本文提出的算法在与现存的需要小生境参数的算法比较中表现出一定的优势,一些测试函数的成功率、收敛速度要好于 FERPSO[6] 和 SPSO。

1 粒子群算法

粒子群优化算法是一种模拟鸟群或者鱼群的社会行为的优化算法[7]。在一个社会化的种群中,每个个体的行为不仅会受到自身经验和以往知识的影响,还会受到种群中其他个体行为的约束。在粒子群优化算法中,每个粒子拥有自身的位置和速度信息,这些信息会根据粒子自身以往经验以及搜索空间中其他粒子的信息调整。在随机产生的粒子群中,每个粒子都是一个求解问题的潜在解,求解的问题是由具体的目标函数来确定,粒子的速度都会根据自己的历史最佳位置和种群的最佳位置调整,多次迭代调整,最后粒子就会接近最优解。算法开

收稿日期: 2011-07-23; **修回日期**: 2011-08-30
作者简介: 史哲文(1982-),女,河南南阳人,讲师,硕士,主要研究方向为智能计算、社会网络分析(zhewen.shi@gmail.com);白雪石(1986-),男(回族),内蒙古包头人,硕士,主要研究方向为图像处理、智能计算;郭禾(1955-),男,山东人,博导,硕士,主要研究方向为分布式计算、计算机视觉、软件工程.

始后,随机分布每个粒子,根据目标函数计算粒子的适应度,每轮迭代中粒子都会依据自身的历史最佳位置 pbest 和种群的最佳位置 gbest 来调整速度,具体公式[8]如下:

$$v_i \leftarrow \chi\left(v_i + R_1\left[0, \frac{\varphi_{\max}}{2}\right] \otimes (p_i - x_i) + R_2\left[0, \frac{\varphi_{\max}}{2}\right] \otimes (p_g - x_i)\right) \tag{1}$$

$$x_i \leftarrow x_i + v_i \tag{2}$$

其中:R_1 和 R_2 是两个独立产生随机数的函数,随机数的范围是$[0, \Psi_{\max}/2]$;$\Psi_{\max}$是一个正常数;符号⊗表示点乘,收缩因子 χ对粒子的移动幅度有抑制作用,可以用来阻止粒子在搜索空间中探索得太远。通常 χ 设置为 0.729 8,根据式(3)计算得到,其中 $\varphi = 4.1$。

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \tag{3}$$

在粒子群算法中,种群中粒子的交互对于粒子个体的行为有重要的影响。粒子的位置不仅受到自身历史最佳位置(探索型信息)的影响,还受到周边粒子的最佳位置(存储型信息)的影响。对于探索型信息 p_i,取决于粒子目前为止到达的最佳位置;对于存储型信息 p_g,取决于种群中所有粒子目前为止到达的最佳位置。

2　基于适应度和距离的小生境算法

在基本粒子群算法中,只需要找到一个全局极值点,所以粒子的存储型信息设定为种群中所有粒子的最佳位置即可。需要强调的是,多模优化问题的目标不是找到一个全局极值点,而是找到所有的全局极值点,这就要求算法能够探索并存储多个局部极值点。

在粒子群算法的基本式(1)(2)中,粒子的历史最佳位置 p_i 就是粒子的个体最优解,种群中所有粒子的历史最佳位置 p_g 就是粒子的全局最优解。多模优化问题中的多个全局最优解存在于多个小生境中,现存的很多算法如 SPSO、NichePSO 都需要一定的参数来确定小生境。本文算法着重于不需要参数来确定小生境,使得粒子能够向多个全局最优解移动。

2.1　搜索策略

在多模优化问题中,粒子的个体最优解可以用来存储粒子自身的最佳位置,而粒子的局部最优解用来探索搜索空间中多个适应度较好的粒子,并且向这些粒子移动,进而不断进行修正,最终收敛到全局最优解。本文提出的算法改变了原始粒子群算法的存储型信息的搜索策略,使得算法可以存储多个局部极值点。

本文提出的算法正是基于这样的思路,确定粒子的局部最优解的原则就是通过计算粒子的参考因子 *RFER* 找到适应度在种群中占有比例较大而且与这个粒子欧氏距离较小的粒子,公式描述如下:

$$RFER_{(i,j)} = \alpha \times \frac{\dfrac{f(p_i)}{f(p_s)}}{\| P_j - P_i \|} \tag{4}$$

其中:$f(p_i)$表示粒子 i 的适应度;$f(p_s)$表示种群中所有粒子的适应度和;$\alpha = \dfrac{\| s \|}{f(P_w)/f(P_s)}$是尺度因子,保证适应度和欧氏距离的作用平衡。$\| s \|$ 是搜索空间的大小,由 $\sqrt{\sum_{k=1}^{\mathrm{Dim}} (x_k^u - x_k^l)^2}$ 确定(这里 x_k^u 和 x_k^l 分别是搜索空间第 k 维的上下界)。$f(P_w)$是当前种群中适应度最差的粒子。$RFER_{(i,j)}$ 表示粒子 i 与 j 的参考因子。

粒子根据与其他粒子计算得到的参考因子 RFER 的最大值就可以确定自己的局部最优解 lbest,使得粒子向适应度较好且离该粒子又较近的粒子移动。式(1)重写如下,p_g 用 p_n 代替。

$$v_i \leftarrow \chi\left(v_i + R_1\left[0, \frac{\varphi_{\max}}{2}\right] \otimes (p_i - x_i) + R_2\left[0, \frac{\varphi_{\max}}{2}\right] \otimes (p_n - x_i)\right) \tag{5}$$

2.2　确定小生境策略

在 2.1 节的搜索策略中,每个粒子都会向离自己最近且适应度又较好的粒子移动,这样可以使得粒子探索多个局部最优解并将其存储起来,进而建立小生境。建立小生境的原则是:如果一个粒子作为其他粒子的局部最优解粒子(lbest),这些粒子与 lbest 粒子的欧氏距离的平均值作为一个潜在小生境的半径 radius,与 lbest 粒子的欧氏距离小于半径的粒子就与这个 lbest 粒子组成一个小生境。另一方面,为了防止产生过多的小生境而导致局部收敛,还需要建立删除小生境的原则:如果有的小生境粒子数小于 2,并且这个粒子还是 lbest 粒子自己,那么就删除这个小生境;如果在下一轮迭代中这个粒子与其他粒子的参考因子 *RFER* 值仍然是这个粒子自己最大,那么这个粒子不再以自己作为局部最优解,取而代之的是参考因子 *RFER* 值第二大的粒子。

3　实验设置

3.1　测试函数

为了测试本文提出算法的性能,笔者使用了九个测试函数,如表 1 所示。

表 1　测试函数

名称	测试函数	定义域	极值点个数
Equal Maxima[9]	$f_1(x) = \sin^6(5\pi x)$	$0 \leqslant x \leqslant 1$	5
Decreasing Maxima[9]	$f_2(x) = \exp\left[-2\log(2) \times \left(\frac{x-0.1}{0.8}\right)^2\right] \times \sin^6(5\pi x)$	$0 \leqslant x \leqslant 1$	1
Uneven Maxima[9]	$f_3(x) = \sin^6(5\pi(x^{3/4} - 0.05))$	$0 \leqslant x \leqslant 1$	5
Uneven Decreasing Maxima[9]	$f_4(x) = \exp\left(-2\log(2) \times \left(\frac{x-0.08}{0.854}\right)^2\right) \times \sin^6(5\pi(x^{3/4} - 0.05))$	$0 \leqslant x \leqslant 1$	1
Two-Peak Trap[10]	$f_5(x) = \begin{cases} \frac{160}{15}(15 - x) & 0 \leqslant x < 15 \\ \frac{200}{5}(x - 15) & 15 \leqslant x \leqslant 20 \end{cases}$	$0 \leqslant x \leqslant 20$	1
Central Two-Peak Trap[10]	$f_6(x) = \begin{cases} \frac{160}{10}x & 0 \leqslant x < 10 \\ \frac{160}{5}(15 - x) & 10 \leqslant x < 15 \\ \frac{200}{5}(x - 15) & 15 \leqslant x \leqslant 20 \end{cases}$	$0 \leqslant x \leqslant 20$	1
Five-Uneven-Peak Trap[11]	$f_7(x) = \begin{cases} 80(2.5 - x) & 0 \leqslant x < 2.5 \\ 64(x - 2.5) & 2.5 \leqslant x < 5.0 \\ 64(7.5 - x) & 5.0 \leqslant x < 7.5 \\ 28(x - 7.5) & 7.5 \leqslant x < 12.5 \\ 28(17.5 - x) & 12.5 \leqslant x < 17.5 \\ 32(x - 17.5) & 17.5 \leqslant x < 22.5 \\ 32(27.5 - x) & 22.5 \leqslant x < 27.5 \\ 80(x - 27.5) & 27.5 \leqslant x \leqslant 30 \end{cases}$	$0 \leqslant x \leqslant 30$	2
Himmelblau's function[9]	$f_8(x,y) = 200 - (x^2 + y - 11)^2 - (x + y^2 - 7)^2$	$-6 \leqslant x, y \leqslant 6$	4
Inverted Vincent function[12]	$f_9(x) = \frac{1}{n}\sum_{i=1}^{n} \sin(10 \times \log(x_i))$	$0.25 \leqslant x_i \leqslant 10$	6^n

测试函数分别具有不同的特点:

a)第一组测试函数 f_1、f_2、f_3、f_4 是一维多模函数。其中:f_1 有 5 个均匀的全局极值点;f_2 与 f_1 相似,不同点是 5 个极值点在高度上依次降低,因此 f_2 只有 1 个全局极值点;f_3 也与 f_1 相似,它的 5 个全局极值点是不均匀分布的;f_4 与 f_3 比较相近,不同点是 5 个极值点在高度上依次降低,因此 f_4 也只有 1 个全局极值点。

b)第二组测试函数 f_5、f_6、f_7 也是一维函数,它们的局部极值点会对全局极值点的识别产生一定影响。例如对于 f_5,当 $x=0$时,函数会收敛到局部极值点,而在初始化的种群中有大约 3/4 的粒子初值介于 015,这些粒子很有可能会向局部极值点而不是全局极值点移动,同样 f_6 对算法的性能也会产生相似的检验,而 f_7 具有 2 个全局极值点和多个局部极值点,使得算法对于全局极值点探测的难度进一步加大。

c)第三组测试函数是二维多模函数,f_8 具有 4 个全局极值点,没有局部极值点。

d)第四组测试函数是复杂测试函数,f_9 的一维形式具有 6 个全局极值点,二维形式具有 36 个极值点,没有局部极值点。

3.2　参数说明

对于测试函数 $f_1$$f_8$,实验中使用 100 个粒子就可以找到所有的全局极值点。实验中比较的三种算法(RFERPSO、FERPSO、SPSO)的迭代次数限定在 500 次,如果算法没有在 500 次以内找到全部的全局极值点,就认为算法失败。为了比较算法的性能,需要为每个测试函数设定一个误差值 ε $(0\leqslant\varepsilon\leqslant1)$和小生境半径 r。误差值 ε 用来判定是否找到了一个全局极值点,如果计算得到的极值点和实际极值点的距离小于这个误差值,那么就认为找到了这个全局极值点,否则认为没有找到。小生境半径 r 用来判定两个全局极值点是否在同一个小生境内,如果两个全局极值点的欧氏距离大于 r 值,那么说明它们不在同一个小生境内;否则在同一个小生境内,所以 r 值的设定不能大于两个最近的全局极值点之间的距离[13]。

算法成功的条件是在规定迭代次数中,找到所有的全局极值点。判断是否找到所有的全局极值点的方法是:如果一个粒子的适应度与全局极值点的距离小于等于误差值 ε,而且它与已经找到的全局极值点的距离大于小生境的半径 r,就认为找到了一个新的极值点。每个粒子都这样判断,直到找到了所有的全局极值点。

对于复杂测试函数 $f_9(2-D)$,实验中使用了 200 个粒子,而 f_9 $(1-D)$的一维形式使用 100 个粒子,迭代次数均限定在 1 000 次。

3.3　评价因素

算法的衡量标准有以下两个:

a)成功率,即 n 次实验中,算法找到所有的全局极值点的次数与 n 的比值。本文的实验中每种算法每个测试函数进行了 50 次实验。

b)收敛速度,即算法找到所有全局极值点所需要的平均迭代次数,由式(6)确定。比较的收敛速度是运行成功的实验计算出的精度平均值。如式(6)所示:N_p 表示全局极值点个数;iter_i 表示第 i 个全局极值点需要的迭代次数;ave_iter 表示收敛速度。

$$\text{ave\_iter}=\frac{1}{N_p}\sum_{i=1}^{N_p}\text{iter}_i \tag{6}$$

4　实验结果

4.1　成功率

第一组测试函数,对于 f_3,本文算法的成功率没有达到 100%,其他三个函数的成功率都达到 100%。第二组测试函数,对于 f_5 和 f_6,本文算法的成功率接近 100%,都高于 FERPSO 的成功率,与 SPSO 接近;对于 f_7,本文算法成功率略低于 FERPSO,要高于 SPSO 的成功率。只有第三组测试函数 f_8,本文算法和 FERPSO 的成功率都达到 100%,远远高于 SPSO 的成功率。第四组复杂测试函数,对于 f_9 的一维形式,本文算法成功率均高于 FERPSO 和 SPSO,如表 2 所示。

表 2　测试函数 $f_1$$f_9$ 的成功率

函数	ε	r	RFERPSO/%	FERPSO/%	SPSO/%
f_1	0.01	0.01	100	98	100
f_2	0.01	0.01	100	100	100
f_3	0.01	0.01	98	100	98
f_4	0.01	0.01	100	100	100
f_5	0.1	0.5	98	88	100
f_6	0.1	0.5	100	88	100
f_7	5	0.5	90	94	74
f_8	0.001	0.5	100	100	24
$f_9(1D)$	0.01	0.2	98	92	96

4.2　收敛速度

第一组测试函数,本文算法的收敛速度均高于另外两种算法,而且明显高于 SPSO 的收敛速度,尤其是对于 f_3,本文算法的收敛速度高于 FERPSO 近 2 倍,高于 SPSO 近 10 倍;第二组测试函数,本文算法的收敛速度仍然明显高于另外两种算法;第三组测试函数,本文算法的收敛速度低于 FERPSO 的收敛速度,高于 SPSO 的收敛速度,如表 3 所示。

表 3　测试函数 $f_1$$f_9$ 的收敛速度

函数	ε	r	RFERPSO	FERPSO	SPSO
f_1	0.01	0.01	9.7	13.35	21.36
f_2	0.01	0.01	1.72	1.78	11.28
f_3	0.01	0.01	4.6	10.62	48.86
f_4	0.01	0.01	3.28	3.54	38.62
f_5	0.1	0.5	26.76	37.5	44.86
f_6	0.1	0.5	38.02	44.86	103.58
f_7	5	0.5	17.36	50.68	186.65
f_8	0.001	0.5	112.9	64.64	180.33
$f_9(1D)$	0.01	0.2	10.8	15.92	26.3

4.3　粒子个数的影响

为了测试粒子个数对成功率的影响,本文选择了测试函数 f_1、f_5、f_8 在粒子数为 30、50 和 100 时比较算法的成功率,实验结果如图 1 所示。其中,横坐标表示粒子个数,纵坐标表示成功率。

对于高维复杂函数 f_9(2D),本文测试了粒子个数对本文算法找到的全局最优解个数的影响,实验结果如图 2 所示。其中横坐标表示粒子个数,纵坐标表示本文算法找到的全局极值点的个数和在复杂函数上找到的全局最优解的个数。

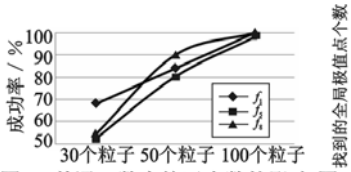

图1　普通函数中粒子个数的影响

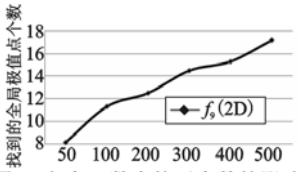

图2　复杂函数中粒子个数的影响

从图中可以看出,随着粒子数的增加,本文算法能够找到的全局极值点的个数也在增加。

5　结束语

本文针对现存的基于粒子群优化的小生境算法需要参数的缺陷,提出一种不需要参数的小生境算法。该算法不仅可以存储多个局部极值点,而且还能使得粒子向适应度较好且离自身较近的粒子移动。实验结果表明,对于九个测试函数,该算法的成功率、收敛速度与 FERPSO 和 SPSO 相比具有更好的性能,并且在复杂函数的实验中也表现出了很好的效果。

参考文献:

[1] KENNEDY J, EBERHART R. Swarm intelligence[M]. San Mateo CA: Morgan Kaufmann,2006:187-219.

[2] 刘宇,吕明伟,李维佳,等. 基于物种的自适应多模态粒子群优化算法[J]. 山东大学学报,2011,46(5):91-96.

[3] BRRITS A E R, Van den BERGH F. A niching particle swarm optimizer[C]//Proc of the 4th Asia-Pacific Conference. 2002: 692-696.

[4] LI Xiao-dong. Adaptively choosing neighborhood bests using species in particle swarm optimizer for multimodal function optimization [C]//Proc of Genetic and Evolutionary Computation Conference. Sea: Kalyanmoy Deb, 2004: 105-116.

[5] PARROTT D, LI Xiao-dong. Locating and tracking multiple dynamic optima by a particle swarm model using speciation[J]. IEEE Trans on Evolutionary Computation,2006,10(4): 440-458.

[6] LI Xiao-dong. Multimodal function optimization based on fitness-Euclidean distance ratio[C]//Proc of Genetic Evolutionary Computation Conference. 2007: 78-85.

[7] KENNEDY J, BARNHART R. Particle swarm optimization[C]// Proc of IEEE International Conference on Neural Networks. 1995: 1942-1948.

[8] CLERC M, KENNEDY J. The particle swarm explosion, stability, and convergence in a multidimensional complex space[J]. IEEE Trans on Evolutionary Computation,2002,6(1):58-73.

[9] DEB K. Genetic algorithms in multimodal function optimization, Technical Report 89002[R]. Tuscaloosa: University of Alabama, 1989.

[10] ACKLEY D. An empirical study of bit vector function optimization [M]// McNEILL A R. Optima for Animals. London: Edward Arnold,1987:170-204.

[11] LI Jian-ping, BALAZS M E, PARKS G T, *et al*. A species serving genetic algorithm for multimodal function optimization[J]. Evolutionary Computation,2002,10(3):207-234.

[12] SHIR O, BACK T. Niche radius adaptation in the CMS-ES niching algorithm[C]//Proc of the 9th International Conference on Parallel Problems Solving Nature. [S.l.]: Springer, 2006: 142-151.

[13] LI Xiao-dong. Niching without niching parameters: particle swarm optimization using a ring topology[J]. IEEE Trans on Evolutionary Computation,2010,14(1):150-169.

(上接第 464 页)

表 2　两种算法的优化结果比较

函数名	项目	文献[5]算法	本文算法	理论最优解
Shuffer F6	最优解	0	0	0
	最坏解	9.71E-3	9.66E-5	
	平均值	4.14E-4	2.22E-6	
	标准差	1.54E-3	1.36E-5	
Camel	最优解	-1.031628	-1.031628	-1.031628
	最坏解	-1.031492	-1.031628	
	平均值	-1.031616	-1.031628	
	标准差	3.01E-5	1.61E-10	
Rosenbrock	最优解	1.38E-18	0	0
	最坏解	3.57E-6	5.62E-11	
	平均值	2.07E-7	2.93E-12	
	标准差	5.80E-7	8.82E-12	
Shuffer F7	最优解	1.64E-10	0	0
	最坏解	2.45E-5	2.71E-20	
	平均值	6.66E-7	1.17E-20	
	标准差	3.46E-6	8.22E-21	
Shubert	最优解	-186.7309	-186.7309	-186.7309
	最坏解	-186.7205	-186.7307	
	平均值	-186.7299	-186.7308	
	标准差	1.83E-3	3.30E-5	
DeJong F1	最优解	2.71E-13	0	0
	最坏解	8.11E-7	2.44E-19	
	平均值	3.25E-8	2.95E-20	
	标准差	1.46E-7	4.38E-20	
Rosenbrock 2	最优解	1.47E-4	1.53E-6	0
	最坏解	3.75E-1	9.31E-2	
	平均值	1.22E-1	5.47E-3	
	标准差	9.79E-2	1.61E-2	
Schwefel's	最优解	20.05	6.07	0
	最坏解	29.07	15.63	
	平均值	24.77	10.04	
	标准差	2.33	2.11	

4　结束语

本文提出的复形粒子群算法,通过复形法充分利用了优秀个体的信息,使局部搜索能力不断得到加强,避免了不成熟收敛,最终能以更高的精度逼近全局最优解。八个测试函数的实验结果表明,本文算法在处理复杂函数优化问题时性能优于文献算法,具有较好的应用价值。

参考文献:

[1] KENNEDY J, EBERHERT R. Particle swarm optimization[C]// Proc of IEEE International Conference on Neural Networks. 1995: 1942-1948.

[2] SHI Yu-hui, EBERHART R C. Fuzzy adaptive particle swarm optimization[C]//Proc of Congress on Evolutionary Computation. 2009: 101-106.

[3] BERGH F, ENGELBRECHT A P. A cooperative approach to particle swarm optimization[J]. IEEE Trans on Evolutionary Computation,2008,8(3):225-239.

[4] 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, 2009, 10(3):281-295.

[5] CLERC M. The swarm and the queen: toward a deterministic and adaptive particle swarm optimization[J]. Evolutionary Computation,2010,10(3):1951-1957.

[6] 郭琦,洪炳熔,吴勃英. 基于复形法求解多智能体避障问题的研究[J]. 计算机工程与应用,2010,37(17):1-7.