自适应搜索空间的混沌蜂群算法

暴 励[1,2], 曾建潮[1]
(1. 太原科技大学 复杂系统与计算智能实验室, 太原 030024; 2. 长治医学院, 山西 长治 046000)

摘 要: 针对人工蜂群(ABC)算法的不足,以种群收敛程度为依据,结合混沌优化的思想,提出一种改进的人工蜂群算法—自适应搜索空间的混沌蜂群算法(SA-CABC)。其基本思想是在原搜索区域的基础上,根据每次寻优的结果自适应地调整搜索空间,逐步缩小搜索区域,并利用混沌变量的内在随机性和遍历性跳出局部最优点,最终获得最优解。基于六个标准测试函数的仿真结果表明, 本算法能有效地加快收敛速度,提高最优解的精度,其性能明显优于基本 ABC 算法,尤其适合高维的复杂函数的寻优。

关键词: 人工蜂群算法; 混沌优化; 自适应搜索空间
中图分类号: TP301.6 **文献标志码**: A **文章编号**: 1001-3695(2010)04-1330-05
doi:10.3969/j.issn.1001-3695.2010.04.034

Self-adapting search space chaos-artificial bee colony algorithm

BAO Li[1,2], ZENG Jian-chao[1]
(1. *Complex System & Computational Intelligence Laboratory, Taiyuan University of Science & Technology, Taiyuan* 030024, *China*; 2. *Changzhi Medical College, Changzhi Shanxi* 046000, *China*)

Abstract: To improve the performance of ABC algorithm, this paper proposed an improved ABC algorithm called self-adapting search space chaos artificial bee colony algorithm (SA-CABC). The main idea was to contract appropriately the ranges of search space according to the results of each optimization, and took use of the randomicity and ergodicity properties of the chaos to break away the local optima, and ultimately found the global optima. Experimental simulations show that the improved algorithm not only accelerates the convergence rate and improves its accuracy, but also effectively avoids the premature convergence problem. This improved algorithm is better than the basic ABC, and provides excellent performance in dealing high-dimensional complex problems.

Key words: artificial bee colony(ABC); chaos optimization; self-adapting search space

0 引言

人工蜂群算法(ABC)是由土耳其埃尔吉耶斯大学 Karaboga[1] 在 2005 年提出的一种基于蜜蜂群智能搜索行为的优化算法。目前,关于 ABC 算法研究与应用还处于初级阶段,但由于其控制参数少、易于实现、计算简洁等优点,已被越来越多的学者所关注。Karaboga 等人[2,3] 已经成功地将 ABC 算法应用于函数的无约束数值优化问题。2007 年,Karaboga 等人[4] 提出了用于解决约束数值优化问题的人工蜂群算法,通过对 13 个标准的约束性优化问题的测试,以及与差分进化算法(DE)、微粒群算法(PSO)的比较,实验结果表明改进的蜂群算法可以有效地解决约束性的优化问题。Karaboga 等人[5,6] 还将 ABC 算法应用于人工神经网络的训练。Srinivasa 等人[7] 将 ABC 算法用于解决分布式系统中的网络重构问题。随后,Karaboga[8] 提出了一种基于人工蜂群算法的数字 IIR 滤波器设计方法,并将其性能与微粒群(PSO)等算法进行了比较。Singh[9] 提出了一种解决最小生成树问题的人工蜂群算法。丁海军等人[10] 通过对 ABC 算法参数的改进,提出了一种适合组合优化问题的改进蜂群算法,并将其应用在 TSP 的求解上。康飞等人[11] 针对人工蜂群算法求解反演分析时存在收敛速度慢、长期停滞等缺陷,引入了文化算法的双层进化结构和具有概率突跳特点的模拟退火操作,提出了一种具有知识引导功能的文化退火人工蜂群算法。为解决参数识别和系统优化提供了一种新方法。

ABC 算法结合全局搜索和局部搜索的方法来使蜜蜂在食物源的探索(exploration)和开采(exploitation)两个方面达到最好的平衡,在函数优化方面的性能优于差分进化算法、微粒群算法等[3,4]。但是,目前 ABC 算法作为一种新的随机优化算法,在接近全局最优解时,仍旧存在着搜索速度变慢、过早收敛、个体的多样性减少,甚至陷入局部最优解等难题。为了加快搜索速度,避免早熟收敛,本文结合 ABC 算法的特性,自适应地调整搜索空间,并引入混沌优化的思想,提出了一种改进的 ABC 算法——自适应搜索空间的混沌人工蜂群算法(SA-CABC)。其主要思想是:每次循环一定的代数后在原搜索区域的基础上,重新构造一个较小的搜索区域,并重新评估每个解,然后继续进行搜索,最终获得最优解。而当某个解陷入局部最优时,则利用混沌变量的内在随机性和遍历性使其跳出局部最优点。仿真结果表明,该算法在收敛速度与精度上较基本 ABC 算法均有明显改善。

收稿日期: 2009-06-25; **修回日期**: 2009-09-09

作者简介:暴励(1975-),男,山西长治人,硕士,主要研究方向为智能计算(libao@sina.com);曾建潮(1963-),男,教授,博导,博士,主要研究方向为复杂系统建模、仿真与优化、智能计算等.

1　ABC 算法

1.1　蜜蜂行为描述

蜂群产生群体智能的最小搜索模型包含三个基本组成要素,即食物源、未被雇佣的蜜蜂和被雇佣的蜜蜂;蜜蜂的基本行为有两种,即为食物源招募蜜蜂和放弃某个食物源[1]。

1)食物源　相当于优化问题中解的位置,在 ABC 算法中食物源的价值由适应度值来表示。

2)未被雇佣的蜜蜂　即侦查蜂(scout)和跟随蜂(onlooker),其主要任务是探索和开采食物源。初始时刻,对于非雇佣蜂有如下两种选择:a)成为侦察蜂,随机搜索蜂巢附近的食物源;b)成为跟随蜂,在观察完引领蜂的摇摆舞后,通过舞蹈的剧烈程度、持续时间等,来确定食物源的收益率,并依据收益率来选择到哪个食物源采蜜。此时,非雇用蜂成为被招募者,即跟随蜂。

3)被雇佣的蜜蜂　即已经发现了食物源的蜜蜂,也称引领蜂(employed bee)。引领蜂储存有某一个食物源的相关信息(位置、花蜜数量等),并将这些信息以一定的概率与其他蜜蜂分享。

蜜蜂对食物源的搜索由三步组成:a)引领蜂发现食物源并记录下花蜜的数量;b)跟随蜂依据引领蜂所提供的花蜜信息,来选定到哪个食物源采蜜;c)确定侦查蜂,寻找新的食物源。

1.2　ABC 算法描述

ABC 算法中,人工蜂群主要有引领蜂、跟随蜂和侦察蜂。ABC 算法在求解优化问题时,食物源的位置被抽象成解空间中的点,蜜蜂采蜜(食物源)的过程也就是搜寻最优解的过程。考虑全局优化问题(P),$\min\{f(X):X\in S\subset R^n\}$,则问题(P)的多个可行解的一个结合称为一个种群,种群中每个元素(可行解)称为一个食物源,每个食物源的优劣程度取决于待优化问题所确定的适应度值,解的个数(SN)等于引领蜂或跟随蜂的个数。本文用 d 维向量 $\boldsymbol{X}_i(x_{i1},x_{i2},\cdots,x_{id})^{\mathrm{T}}\in S$ 来表示第 i 个食物源的位置。首先,ABC 算法生成含有 SN 个解(食物源)的初始种群,每个解 $\boldsymbol{X}_i(i=1,2,\cdots,SN)$ 是一个 d 维的向量;然后,蜜蜂对所有的食物源进行循环搜索,循环次数为 $C(C=1,2,\cdots,MCN)$。引领蜂首先对相应的食物源(解)进行一次邻域搜索,如果搜索到的食物源(解)的花蜜数量(适应度)优于以前的,则用新的食物源位置代替旧的食物源位置,否则保持旧的食物源位置不变。所有的引领蜂完成搜索之后,回到舞蹈区把食物源上花蜜数量的信息通过跳摇摆舞传达给跟随蜂。跟随蜂根据得到的信息按照概率选择食物源。花蜜越多的食物源,被选择的概率越大。跟随蜂选中食物源后,也进行一次邻域搜索,并保留较好的解。ABC 算法就是通过如此重复搜索,最终来找到最优解。

引领蜂和跟随蜂依据式(1)进行食物源位置更新:

$$v_{ij}=x_{ij}+r_{ij}(x_{ij}-x_{kj}) \tag{1}$$

其中:$k\in\{1,2,\cdots,SN\}$,$j\in\{1,2,\cdots,d\}$,这两个数都是随机选取的,但是 k 不能等于 i(k 是 i 邻域的一个解);$r_{ij}\in[-1,1]$是一个随机数,它控制 x_{ij} 邻域的生成范围,随着搜索接近最优解,邻域的范围会逐渐减小。

ABC 算法中跟随蜂对食物源的选择是通过观察完引领蜂的摇摆舞来判断食物源的收益率,并依据收益率大小来选择到哪个食物源采蜜。收益率通过适应度值来表示,选择概率 P_i 按照下式确定:

$$P_i=fit_i/\sum_{i=1}^{SN}fit_i \tag{2}$$

其中:fit_i 是第 i 个解的适应度值;SN 是解的个数。

在 ABC 算法中,还有一个控制参数 limit,它用来记录某个解被更新的次数。假定某个解连续经过 limit 次循环之后没有得到改善,表明这个解陷入局部最优,那么这个位置就要被放弃,与这个解相对应的引领蜂也转变为侦察蜂。假设被放弃的解是 x_i 且 $j\in\{1,2,\cdots,d\}$,那么就由侦查蜂通过式(3)随机产生一个新的解来代替 x_i。

$$x_i^j=x_{\min}^j+\mathrm{rand}(0,1)(x_{\max}^j-x_{\min}^j) \tag{3}$$

2　自适应搜索空间的混沌蜂群算法

2.1　动态调整搜索空间

设种群空间的解为 n 个,每个解是 d 维向量。

初始状态,按照式(1)生成空间的解:$X_1=(x_{11},x_{12},\cdots,x_{1d})$,$X_2=(x_{21},x_{22},\cdots,x_{2d})$,$\cdots$,$X_n=(x_{n1},x_{n2},\cdots,x_{nd})$,则

$$Y=(y_1,y_2,\cdots,y_i,\cdots,y_d)=(\max(|x_{11}|,|x_{21}|,\cdots,|x_{n1}|),\max(|x_{12}|,|x_{22}|,\cdots,|x_{n2}|),\cdots,\max(|x_{1i}|,|x_{2i}|,\cdots,|x_{ni}|),\cdots,\max(|x_{1d}|,|x_{2d}|,\cdots,|x_{nd}|))$$

其中:分量 y_i 表示种群在这维坐标上的空间分布,y_i 越大,种群在第 i 维坐标上的空间分布就越大。

在 SA-CABC 算法中,每当搜索进行到一定的迭代次数时,就利用 Y 的最大可能变化区间,按照式(4)重新生成种群个体,即引领蜂根据式(4)产生新解,并计算其适应度值,然后对它们进行评估后继续搜索。经过如此循环迭代,使产生初始种群的区间逐步收缩,从而可加快进化迭代进程,提高了算法的整体运行效率。

$$v_i=2y_i\times\mathrm{rand}(0,1)-y_i \tag{4}$$

按照上述方法,随着搜索空间的缩小,会出现两个问题:a)最优解有可能被排除在缩小后的搜索空间之外,这样问题的最优解将无法被搜索到;b)个体位置的运动范围被大大缩减,降低了算法突破局部最优的能力。如果大部分个体均在相同的局部极值附近运动时,算法容易出现暂时的停滞现象,这样,突破局部极值的限制可能需要经过很长一段时间,也可能无法突破这一限制而陷入局部最优点。为此,本文通过两种方式来改善以上提出的问题:a)把种群中的个体分为两部分,一部分动态调整搜索区域,加快算法的收敛速度,另一部分在原空间内搜索,保证位于空间边缘的解不被忽略,仿真试验证明了该方法的可行性;b)搜索空间经过调整后,不马上进行下一次压缩,而是在每作一次压缩后,就迭代几次,使群体对新的环境有个适应的过程,然后再作下一次压缩。

2.2　混沌搜索

混沌是自然界广泛存在的一种非线性现象,它看似混乱,却有着精致的内在结构,具有随机性、遍历性及规律性等特点,在一定范围内能按其自身的规律不重复地遍历所有状态[12]。

一般混沌是指由确定性方程得到的具有随机性的运动状态,呈现混沌状态的变量称为混沌变量。常用的 Logistic 映射就是一个典型的混沌系统,其方程如下:

$$z_{n+1} = \mu z_n + (1 - z_n);\quad n = 0,1,2,\cdots \tag{5}$$

其中:μ 为控制参数,式(5)可以看做是一个动力学系统。μ 值确定后,由任意初值 $z_0 \in [0,1]$,可以迭代出一个确定的时间序列 $z_0, z_1, z_2, \cdots$。当 μ= 4 时,系统完全处于混沌状态[15]。

由于混沌的遍历性,混沌优化算法易跳出局部最优解,是一种很好的搜索机制。国内外许多学者将混沌搜索和群智能算法这两种机制有机结合,并作了广泛研究。文献[14 ~ 16]在遗传算法(GA)中引入混沌算子,提出了混沌遗传算法,取得了较好的效果。文献[17 ~ 19]把微粒群算法与混沌搜索相结合,提出了混沌粒子群优化算法,很好地提高了搜索性能。

在 ABC 算法中,如果某个解连续经过 limit 次循环后没有得到改善,表明这个解陷入局部最优,则会随机产生新解来替换它。而在本文提出的 SA-CABC 算法中,对搜索停滞的解是利用混沌搜索来给该解重新赋值,使其跳出局部最优。其主要思想是利用混沌运动的遍历性以当前搜索停滞的解为基础产生混沌序列,用产生的混沌序列中的最优位置替代它原来的位置,通过这种处理使得搜索停滞的解继续进化,提高算法的收敛速度和精度。本文假设搜索停滞的解是 $X_k = (x_{k1}, x_{k2}, \cdots, x_{kd})$,$x_{ki} \in [a_i, b_i]$ 则对它进行混沌优化,取式(5)作为混沌迭代公式。其主要步骤如下:

a)将 X_k 映射到 Logistic 方程的定义域[0,1]内,即

$$z_{ki}^{o} = \frac{x_{ki} - a_i}{b_i - a_i};k = 1,2,\cdots,n;i = 1,2,\cdots,d$$

b)用 Logistic 方程进行迭代产生混沌变量序列 z_k^m ($m = 1, 2, \cdots, C_{\max}$)。其中 $C_{\max}$ 是混沌搜索的最大迭代次数。

c)把产生的混沌序列 z_{ki}^m ($m = 1,2,\cdots,C_{\max}$) 通过逆映射 $x_{ki} = a_i + (b_i - a_i) z_{ki}^m$ 返回到原解空间,得 $X'_k = (x'_{k1}, x'_{k2}, \cdots, x'_{kd})$,计算其适应值 $f' = f(X'_k)$,并将其与原来的解比较,保留最好解。

d)若达到最大迭代次代数,则优化过程结束,否则返回步骤 b)。

2.3　选择策略的确定

在 SA-CABC 算法中,跟随蜂选择食物源的选择策略采用锦标赛[20]的方法,因为锦标赛选择只把适应值的相对值作为选择的标准,而且对适应值的正负也没有要求,从而避免了超级个体对算法的影响,在一定程度上可以避免算法过早收敛和停滞现象的发生。

2.4　SA-CABC 算法流程

算法步骤如下:

a)初始化种群的解 x_i ($i = 1 \cdots n$)。

b)计算每个解 x_i 的适应度值。

c)判断是否调整搜索空间,如符合调整的条件,则引领蜂根据式(4)产生新的解空间,否则根据式(1)产生新解 v_i,并且计算其适应度值。

d)如果 v_i 的适应度值好于 x_i,则用 v_i 替换 x_i,否则保留 x_i 不变。

e)根据锦标赛选择策略利用式(2)计算与 x_i 相关的概率值 P_i。

f)跟随蜂根据 P_i 选择食物源(解),并根据式(1)进行邻域搜索产生新解 v_i,计算其适应度值。

g)如果 v_i 的适应度值好于 x_i,则用 v_i 替换 x_i,否则保留 x_i 不变。

h)判断是否有要放弃的解,如果存在,则利用混沌搜索产生一个新解来代替它。

i)记录迄今为止最好的解。

j)判断是否满足循环终止条件,如满足输出最优结果,否则返回 c)。

3　实验结果与分析

3.1　测试函数的参数设置

为了检验改进算法的性能,在仿真实验中选择了六个测试函数[21],并根据函数性质分为具有单个极小点(单模态)和多个局部极小点(多模态)两大类。表 1 显示了这些测试函数的定义、取值范围和理论全局最优解。

表 1　六个标准测试函数

Function	Equation	Range	Min
Schwefel Problem 1.2	$f_1(x) = \sum_{i=1}^{n}(\sum_{j=1}^{i} x_j)^2$	$[-100,100]$	0
Rosenbrock	$f_2(x) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-30,30]$	0
Schwefel Problem 2.26	$f_3(x) = -\sum_{i=1}^{n}(x_i \sin\sqrt{\lvert x_i \rvert}) + 418.982\,9n$	$[-500,500]$	0
Rastrigin	$f_4(x) = \sum_{i=1}^{n}[x_1^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12,5.12]$	0
Ackley	$f_5(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i) + 20 + e)$	$[-32,32]$	0
Griewank	$f_6(x) = \frac{1}{4000}\sum_{i=1}^{n}x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-600,600]$	0

表 1 中,f_1、f_2 是单模态函数。f_1 函数是 Sphere 函数的变形,增加了各维度之间的相互作用,它们主要用来测试算法的寻优精度,考察算法的执行性能;f_2 函数是一个经典的复杂优化问题,取值区间内走势平坦,它的全局最优点位于一个平滑、狭长的抛物线山谷内,要收敛到全局最优点的机会微乎其微,通常用于检验算法收敛速度,考察算法的执行效率;函数 $f_3 \sim f_6$ 是复杂的非线性多模态函数,具有许多局部极值点,一般算法都较难找到全局最优值,因此可用来检验算法的全局搜索性能和避免早熟的能力。在固定的迭代次数下,通过对测试函数的最优值、最差值、平均值和方差的考察来对算法进行评估。

3.2　实验结果分析

为了验证以上分析的有效性,将本文提到的 SA-CABC 算法与基本 ABC 算法进行比较实验。在实验中,种群个数均设为 100,引领蜂和跟随蜂的个数均为 50,测试函数分别取 50 维和 100 维,相应的最大迭代次数分别为 1 000 和 3 000,limit =

30,在搜索过程中每隔 5 代对搜索空间进行一次调整。Logistic 混沌序列中 $\mu=4$,$C_{max}=100$。针对每个测试函数,各算法均随机运行 30 次求其平均值。各函数在不同维数下的最优值、最差值、平均值和方差的比较如表 2、3 所示。为了进一步比较算法的性能,图 1 ~6 给出每个函数 50 维时平均适应值的进化过程曲线。图 7 ~12 给出每个函数 100 维时平均适应值的进化过程曲线。图中的横坐标表示循环代数,纵坐标表示平均适应值。

表 2　各函数平均适应值比较结果(50 维)

函数	算法	均值	方差	最大值	最小值
f_1 Schwefel Problem 1.2	ABC	3.848660e+004	5.199943e+003	4.730240E+004	2.834136E+004
	SA-CABC	7.439490e-001	1.090007e+000	4.325602E+000	1.318367E-003
f_2 Rosenbrock	ABC	4.930503e+001	4.348459e+001	1.414591E+002	2.073254E-001
	SA-CABC	7.957118e+000	5.835000e+000	2.606172E+001	7.253819E-001
f_3 Schwefel Problem 2.26	ABC	1.256950e+003	1.908990e+002	1.562290E+003	8.668952E+002
	SA-CABC	8.204321e-004	1.610429e-004	1.211104E-003	6.573548E-004
f_4 Rastrigin	ABC	5.123967e+000	2.415273e+000	9.872841E+000	1.009375E+000
	SA-CABC	3.197495e-005	3.128849e-005	1.152077E-004	1.520781E-007
f_5 Ackley	ABC	1.562519e-002	1.047046e-002	4.748646E-002	3.860550E-003
	SA-CABC	6.854994e-004	3.917473e-004	1.521886E-003	7.299477E-006
f_6 Griewank	ABC	1.077579e-003	2.958124e-003	1.587020E-002	2.141047E-006
	SA-CABC	4.155794e-006	3.944642e-006	1.380544E-005	1.023970E-008

表 3　各函数平均适应值比较结果(100 维)

函数	算法	均值	方差	最大值	最小值
f_1 Schwefel Problem 1.2	ABC	1.194049e+005	1.121189e+004	1.385360E+005	9.776777E+004
	SA-CABC	5.623618e-001	8.049912e-001	3.323914E+000	3.030532E-004
f_2 Rosenbrock	ABC	2.191075e+001	2.798687e+001	9.209682E+001	6.353412E-002
	SA-CABC	8.389768e+000	1.699884e+001	9.680935E+001	6.289571E-001
f_3 Schwefel Problem 2.26	ABC	1.794497e+003	3.018012e+002	2.372057E+003	9.592072E+002
	SA-CABC	1.272822e-003	6.688051e-008	1.273052E-003	1.272771E-003
f_4 Rastrigin	ABC	3.980298e+000	1.649066e+000	7.031541E+000	1.000407E+000
	SA-CABC	6.526458e-009	5.355012e-009	2.658804E-008	9.233929E-010
f_5 Ackley	ABC	1.997337e-004	8.494032e-005	4.104943E-004	8.801201E-005
	SA-CABC	8.545156e-006	3.784372e-006	1.696285E-005	1.912228E-006
f_6 Griewank	ABC	5.648466e-007	2.553564e-006	1.397109E-005	2.132728E-009
	SA-CABC	1.911345e-010	2.540139e-010	1.095047E-009	8.060552E-012

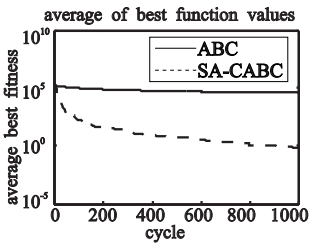

图1　$f_1(x)$ Schwefel Problem 1.2 进化曲线

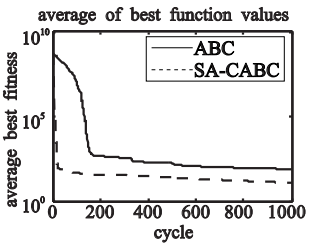

图2　$f_2(x)$ Rosenbrock 进化曲线

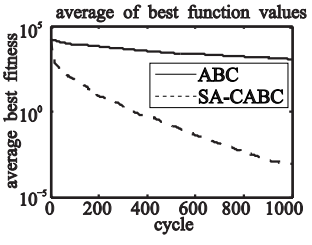

图3　$f_3(x)$ Schwefel Problem 2.26 进化曲线

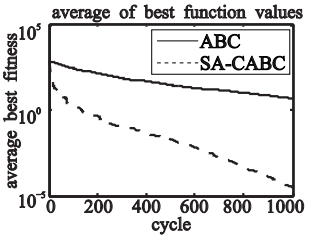

图4　$f_4(x)$ Rastrigin函数进化曲线

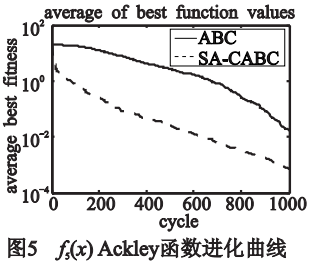

图5　$f_5(x)$ Ackley函数进化曲线

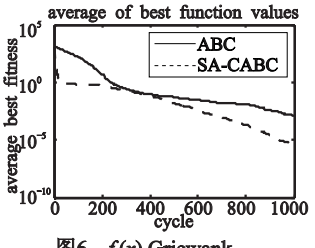

图6　$f_6(x)$ Griewank 函数进化曲线

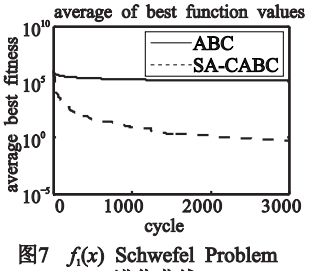

图7　$f_1(x)$ Schwefel Problem 1.2 进化曲线

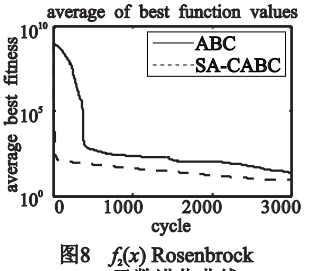

图8　$f_2(x)$ Rosenbrock 函数进化曲线

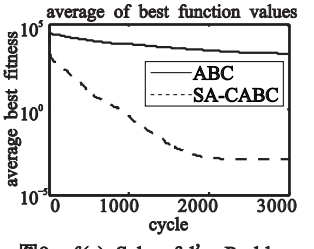

图9　$f_3(x)$ Schwefel's Problem 2.26 进化曲线

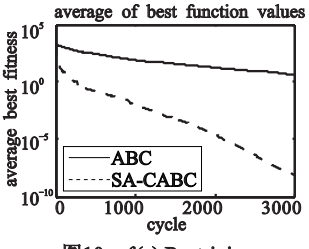

图10　$f_4(x)$ Rastrigin 函数进化曲线

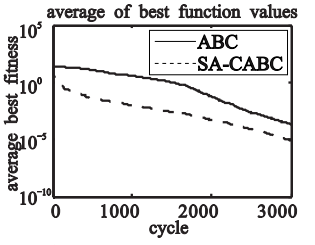

图11　$f_5(x)$ Ackley 函数进化曲线

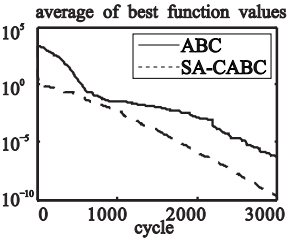

图12　$f_6(x)$ Griewank 函数进化曲线

a)由图 1 和 7 可知,对于复杂的单模态函数 Schwefel Prob-

lem 1.2,ABC 算法的收敛速度非常慢,几乎没有太大的变化,而 SA-CABC 算法在 50 维时收敛速度明显快于 ABC 算法,在 100 维时 SA-ABC 算法在进化后期虽然收敛速度略有减慢,但依然优于 ABC 算法。

b)由图 2 和 8 可知,对于复杂的单模态函数 Rosenbrock,维数取 50 和 100 时,在进化早期由于搜索空间的缩小,使得 SA-CABC 具有较快的收敛速度,但由于该函数的复杂性,在进化后期收敛速度明显减慢,但依然优于 ABC 算法。由于 Rosenbrock 函数自身的特性,它很容易陷入局部最优而搜索停滞,但 SA-ABC 算法引入了混沌优化的思想,使其尽可能地跳出局部最优,提高了算法的收敛精度。从表 2、3 中的数据显示,SA-CABC 的稳定性也好于 ABC 算法。

c)多模态函数 Schwefel Problem 2.26 的取值范围为[500,500],它在点(420.968 7,…,420.968 7)处取得理论最优值 0,该函数的最优值位于搜索空间的边界附近。由图 3 和 9 可知,在固定的迭代次数下,ABC 算法的收敛非常缓慢,SA-CABC 算法的收敛速度明显快于 ABC 算法,但在 100 维时,SA-CABC 算法在进化后期也有搜索停滞的迹象。由此可以看出,SA-CABC 算法对于最优解位于搜索空间边缘的情况,它也能较好地找到最优解。

d)多模态函数 Rstrigin、Ackley、Griewank 均是复杂的非线性全局优化问题,主要用来测试算法的全局搜索性能。从图 4~6和图 10~12 可以看出,SA-CABC 算法的收敛速度和精度均高于 ABC 算法,几乎线性递减地收敛到最优解,尤其对于 Rastrigin 函数,效果更加明显。由此可见,SA-CABC 算法具有良好的全局搜索性能和较快的搜索速度。

总之,通过以上分析可知,无论是单模态还是多模态函数,无论是在 50 维还是更高的 100 维,SA-CABC 比基本 ABC 在收敛精度和收敛速度上均有非常显著的提高。因此,SA-CABC 算法的性能明显优于基本 ABC 算法,并且随着维数的增加,SA-CABC 的性能也保持了较好的稳定性。

4 结束语

本文针对 ABC 算法的特性,自适应地调整搜索空间,并结合混沌优化的思想,提出了一种自适应搜索空间的混沌蜂群(SA-CABC)算法。通过对较复杂的全局优化问题的仿真结果表明,该算法收敛性能明显优于基本的 ABC 算法,改进的 ABC 算法收敛速度更快、精度更高、运行更为稳定,其全局搜索与跳出局部最优的能力均强于基本 ABC 算法。尤其对于求解搜索空间巨大的高维全局优化问题,优势更加突出,这是因为它动态缩小搜索空间,加速收敛的同时,又保证了种群的多样性,所以大大提高了算法的搜索效率。

参考文献:

[1] KARABOGA D. An idea based on honey bee swarm for numerical optimization, Technical Report-TR06[R]. Kayseri:Erciyes University, Engineering Faculty,Computer Engineering Department,2005.

[2] KARABOGA D, BASTURK B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm[J]. Journal of Global Optimization, 2007,39(3): 459-471.

[3] KARABOGA D, BASTURK B. On the performance of artificial bee colony (ABC) algorithm[J]. Applied Soft Computing, 2008,8(1):687-697.

[4] KARABOGA D, BASTURK B. Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems[C]// Proc of Advances in Soft Computing: Foundations of Fuzzy Logic and Soft Computing. Berlin: Springer-Verlag,2007: 789-798.

[5] KARABOGA D, AKAY B B. Artificial bee colony algorithm on training artificial neural networks[C]//Proc of the 15th IEEE Signal Processing and Communications Applications Conference. 2007:1-4.

[6] KARABOGA D, AKAY B B, OZTURK C. Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks[C]//Proc of Modeling Decisions for Artificial Intelligence Conference. Berlin: Springer-Verlag,2007:318-319.

[7] SRINIVASA R R, NARASIMHAM S V L, RAMALINGARAJU M. Optimization of distribution network configuration for loss reduction using artificial bee colony algorithm[J]. International Journal of Electrical Power and Energy Systems Engineering, 2008,1(2): 709-715.

[8] KARABOGA N. A new design method based on artificial bee colony algorithm for digital IIR filters[J]. Journal of the Franklin Institute, 2009,346(4):328-348.

[9] SINGH A. An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem[J]. Applied Soft Computing, 2009,9(2):625-631.

[10] 丁海军,李峰磊. 蜂群算法在 TSP 问题上的应用及参数改进[J]. 中国科技信息,2008(3):241-243.

[11] 康飞,李俊杰,许青,等. 改进人工蜂群算法及其在反演分析中的应用[J]. 水电能源科学,2009,27(1):126-129.

[12] 李兵,蒋慰孙. 混沌优化方法及其应用[J]. 控制理论与应用,1997,14(4):613-615.

[13] 张春慨,徐立云,邵惠鹤. 改进混沌优化及其在非线性约束优化问题中的应用[J]. 上海交通大学学报,2000, 34(5):593-596.

[14] YUAN X, YUAN Y, ZHANG Y. A hybrid chaotic genetic algorithm for short-term hydro system scheduling[J]. Mathematics and Computers in Simulation, 2002,59(4):319-327.

[15] 袁晓辉,袁艳斌,王乘. 一种新型的自适应混沌遗传算法[J]. 电子学报, 2006, 34(4): 708-712

[16] 陈炳瑞,杨成祥,冯夏庭,等. 自适应混沌遗传混合算法及其参数敏感性分析[J]. 东北大学学报:自然科学版,2006,27(6):689-693.

[17] LIU B, WANG L, JIN Y H, *et al*. Improved particle swarm optimization combined with chaos [J]. Chaos Solitons &Fractals,2005, 25(21): 1261-1271.

[18] 高尚, 杨静宇. 混沌粒子群优化算法研究[J]. 模式识别与人工智能, 2006, 19(2): 266-270.

[19] 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.

[20] BAO Li, ZENG Jian-chao. Comparison and analysis of the selection mechanism in artificial bee colony algorithm[C]//Proc of the 9th International Conference on Hybrid Intelligent Systems. Washington DC: IEEE Computer Society, 2009:411-416.

[21] YAO Xin, LIU Yong, LIN Guang-ming. Evolutionary programming made faster[J]. IEEE Trans on Evolutionary Computation, 1999,3(2):82-102.