文章编号: 1001-0920 (2014) 01-0123-06 DOI: 10.13195/j.kzyjc.2012.1301

基于局部搜索的人工蜂群算法

刘三阳, 张 平, 朱明敏

(西安电子科技大学 数学与统计学院，西安 710071)

摘 要: 针对人工蜂群算法存在收敛速度慢、易早熟等缺点, 提出一种改进的人工蜂群算法. 利用随机动态局部搜索算子对当前的最优蜜源进行局部搜索, 以加快算法的收敛速度; 同时, 采用基于排序的选择概率代替直接依赖适应度的选择概率, 维持种群的多样性, 以避免算法出现早熟收敛. 对标准测试函数的仿真实验结果表明, 所提出的算法具有较快的收敛速度和较高的求解精度.

关键词: 人工蜂群；局部搜索算子；排序选择；函数优化

中图分类号: TP18 **文献标志码:** A

Artificial bee colony algorithm based on local search

LIU San-yang, ZHANG Ping, ZHU Ming-min

(School of Mathematics and Statistics，Xidian University，Xi'an 710071，China．Correspondent：ZHANG Ping，E-mail: pzhangxdedu@163.com)

Abstract: Taking into account the basic artificial bee colony algorithm converges slowly and prematurely, an improved artificial bee colony algorithm based on local search is proposed. The method makes full use of the stochastic dynamic local search to optimize the current best solution to speed up the convergence rate. In order to maintain the population diversity and avoid premature convergence, the selection probability based on ranking is used instead of depending on fitness directly. Through the simulation experiment on a suite of standard functions, the results show that the algorithm has a faster convergence rate and higher solution accuracy.

Key words: artificial bee colony；local search；rank selection；function optimization

0 引 言

人工蜂群 (ABC) 算法是 Karaboga[1]于 2005 年提出的一种群体智能优化算法. ABC 算法模拟蜂群采蜜的智能行为, 结构简单、参数较少、易于实现, 受到了众多学者的关注和研究. ABC 算法最初是为解决函数优化[1]问题提出的, 目前已推广到很多领域, 如文献 [2] 将 ABC 算法用于人工神经网络的训练, 文献 [3] 将 ABC 算法应用于模糊聚类, 文献 [4] 利用 ABC 算法解决无线传感器网络的动态部署问题. 但是, ABC 算法作为一种随机优化算法, 与其他进化算法类似, 也存在收敛速度慢、易早熟等缺点. 近年来, 人们对基本 ABC 算法提出了多种改进策略. 文献 [5] 提出使用混沌序列产生初始解; 文献 [6] 受粒子群算法的启发, 给出了由最优解引导的搜索算子; 文献 [7] 将粒子群算法思想融入人工蜂群算法, 提出了混合算法; 文献 [8] 通过控制扰动频率来加快算法的收敛速度. 这些改进的 ABC 算法都在一定程度上改善了算法的性能, 但很难实现在避免算法早熟的同时加快算法的收敛速度.

为了提高 ABC 算法的收敛速度和避免早熟现象, 本文在基本 ABC 算法中引入了局部搜索算子, 利用随机动态局部搜索算子对目前找到的最优蜜源进行局部搜索, 以加快算法的收敛速度. 同时, 为了维持种群的多样性, 避免算法出现早熟收敛, 采用基于排序的选择概率代替直接依赖适应度的选择概率. 该算法充分利用了 ABC 算法的全局寻优能力以及局部搜索算子的局部快速收敛能力. 通过对标准测试函数的仿真实验, 并与基本 ABC 算法以及几种改进 ABC 算法进行比较, 表明了所提出的算法能显著提高算法的收敛速度和全局寻优能力.

收稿日期: 2012-08-29；**修回日期:** 2012-11-27.

基金项目: 国家自然科学基金项目(60974082)；中央高校基本科研业务费专项资金项目(K5051270002)；西安电子科技大学基本科研业务项目(K5051270013).

作者简介: 刘三阳(1959–), 男, 教授, 博士生导师, 从事智能信息处理、最优化方法等研究；张平(1988–), 女, 硕士生, 从事进化计算、最优化理论与方法的研究.

1 人工蜂群算法

在ABC算法中, 人工蜂群包括引领蜂、跟随蜂和侦查蜂3类. 假设在D维空间中, 种群规模为$2\times$SN (引领蜂个数$=$跟随蜂个数$=$SN), 蜜源与引领蜂一一对应, 即蜜源数目为SN, 第i个蜜源的位置记为$X_i=(x_{i1},x_{i2},\cdots,x_{iD})$. 每个蜜源的位置代表优化问题的一个候选解, 花蜜的数量反映解的质量. 人工蜂群搜索最优蜜源的过程如下.

1) 引领蜂对当前蜜源进行邻域搜索, 产生新蜜源, 贪婪选择较优蜜源;

2) 跟随蜂根据引领蜂分享的信息选择一个蜜源, 进行邻域搜索, 贪婪选择较优蜜源;

3) 引领蜂放弃蜜源, 转变成侦查蜂, 并随机搜索新的蜜源.

搜索过程中, 跟随蜂根据引领蜂分享的信息, 以轮盘赌的方式按下式选择一个蜜源:

$$P_i=\frac{\mathrm{fit}_i}{\sum\limits_{j=1}^{\mathrm{SN}}\mathrm{fit}_j}. \tag{1}$$

其中fit_i是花蜜数量(适应度), 按下式计算:

$$\mathrm{fit}_i=\begin{cases}1/(1+f_i),\ f_i\geqslant 0;\\ 1+\mathrm{abs}(f_i),\ f_i<0.\end{cases} \tag{2}$$

这里f_i是第i个解的目标函数值.

引领蜂和跟随蜂按下式搜索新蜜源:

$$v_{ij}=x_{ij}+r(x_{ij}-x_{kj}). \tag{3}$$

其中: k是随机产生的整数, $k\in 1,2,\cdots,$SN且$k\neq i$, $j\in 1,2,\cdots,D$; $r\in[-1,1]$是一个随机数. 若在一定循环次数(limit)后, 蜜源质量没有提高, 则引领蜂放弃该蜜源, 转变为侦查蜂, 按下式随机产生新蜜源:

$$x_{ij}=l_j+\mathrm{rand}(0,1)(u_j-l_j), \tag{4}$$

其中l_j和u_j分别是变量x_{ij}的下界和上界.

2 基于局部搜索的人工蜂群算法

在基本ABC算法中, 蜂群依靠个体之间的信息共享探索新蜜源, 随机性大, 所以基本ABC算法具有较强的全局搜索能力, 但其对当前找到的最优解没有充分开发, 局部搜索能力差, 当蜂群接近最优蜜源时搜索效率明显下降, 从而导致算法的收敛速度很慢. 同时, 跟随蜂根据概率(1)选择蜜源, 较差蜜源几乎得不到更新, 从而减少了种群多样性. 本文采取两种策略来改善算法性能. 首先, 针对ABC算法探索能力强、开采能力弱的缺点, 在ABC算法中引入局部搜索算子, 该算子对当前最优解进行邻域搜索, 以有效弥补基本ABC算法开采能力弱的缺点; 然后, 采用基于排序的选择概率, 使得选择过程只与蜜源的顺序有关, 而与适应度无直接关系, 从而使较差蜜源被选择的概率增大, 维持了种群的多样性. 下面具体介绍两种策略的实现过程.

2.1 局部搜索算子

动态随机搜索技术[9]是由Hamzaceb和Kutay在基本随机搜索技术的基础上提出的. 它包括一般搜索和局部搜索, 其中局部搜索算子是在当前解的邻域内进行随机搜索, 并且搜索步长逐渐缩小(缩小比例为0.5), 使得算子对当前解进行充分开发. 目前已应用于粒子群算法[10]和遗传算法[11], 并取得了很好的效果. 本文将此搜索算法应用到ABC算法中, 同时利用局部搜索算子开发能力强的特点来弥补ABC算法探索能力强而开发能力弱的缺陷. 在粒子群算法和遗传算法中, 将初始搜索步长设置得较大, 以提高算法的探索能力, 随着步长的变小, 该算法将侧重于局部开采. 本文将局部搜索算子嵌入ABC算法, 主要是基于如下考虑: 1) 针对ABC算法探索能力强而开发能力弱的缺陷, 利用局部搜索算子所具有的强开发能力对当前最优蜜源进行局部搜索, 使其快速收敛到搜索区域的最优蜜源; 2) 基本ABC算法中的搜索算子每次只改变一个变量, 而局部搜索算子可以同时改变多个变量, 这样的搜索方式能够有效加快算法的收敛速度.

具体步骤如下.

Step 1: 参数设置. 设置局部搜索迭代次数为E, 初始搜索步长为α_0, 局部搜索迭代计数器为$\mathrm{epoch}=0, k=0, x_{\mathrm{current}}=x_{\mathrm{best}}$.

Step 2: 置迭代计数器$n=0$.

Step 3: 生成随机向量dx, 且满足$-\alpha_k\leqslant dx\leqslant\alpha_k$.

Step 4: 更新$\mathrm{epoch}, \mathrm{epoch}=\mathrm{epoch}+1$.

Step 5: $f_{\mathrm{new}}=f(x_{\mathrm{current}}+dx)$.

若$f_{\mathrm{new}}<f_{\mathrm{best}}$, 则$f_{\mathrm{best}}=f_{\mathrm{new}}, x_{\mathrm{best}}=x_{\mathrm{current}}+dx, n=n+1$, 转Step 7;

若$f_{\mathrm{new}}<f_{\mathrm{current}}$, 则$f_{\mathrm{current}}=f_{\mathrm{new}}, x_{\mathrm{current}}=x_{\mathrm{current}}+dx, n=n+1$, 转Step 7.

Step 6: $f_{\mathrm{new}}=f(x_{\mathrm{current}}-dx)$.

若$f_{\mathrm{new}}<f_{\mathrm{best}}$, 则$f_{\mathrm{best}}=f_{\mathrm{new}}, x_{\mathrm{best}}=x_{\mathrm{current}}-dx, n=n+1$, 转Step 7;

若$f_{\mathrm{new}}<f_{\mathrm{current}}$, 则$f_{\mathrm{current}}=f_{\mathrm{new}}, x_{\mathrm{current}}=x_{\mathrm{current}}-dx, n=n+1$, 转Step 7.

Step 7: 如果$n<N$, 则转Step 3.

Step 8: $k=k+1$.

Step 9: $\alpha_k=\alpha_{k-1}\times 0.5$.

Step 10: 若$\mathrm{epoch}=E$, 则算法终止, 否则转Step 2.

x_{best}为当前找到的最优蜜源; $f_{\mathrm{best}}, f_{\mathrm{new}}, f_{\mathrm{current}}$为对应蜜源的适应度.

为了充分发挥局部搜索算子的开采能力, 初始搜索步长 α_0 与当前最优解的数量级应尽量保持一致, 本文取 $\alpha_0 = x_{\rm best}$.

2.2　基于排序的选择概率

在基本 ABC 算法中, 跟随蜂对蜜源的选择概率直接依赖蜜源的适应度, 具有高适应度的蜜源被选择的概率较大. 但是, 较差蜜源也可能隐含有用的信息, 特别是算法在进化初期很可能会产生超常个体, 如果按式 (1) 中的概率选择蜜源, 这些超常个体会控制选择过程, 从而影响算法的全局收敛能力. 基于排序的选择概率在文献 [12] 中取得了很好的数值效果而且操作简单、易于实现. 这种选择概率与目标函数值没有直接关系, 只与蜜源的顺序有关, 是一种有效的控制选择压力的方法. 本文采用基于排序的选择概率, 使得较差蜜源被选择的概率增加, 避免了超常个体对选择过程的负面影响. 首先, 根据适应度值对所有蜜源进行排序, 蜜源的适应度越大, 位置越靠前; 然后, 根据下式计算排序后第 k 个蜜源被选择的概率:

$$\begin{aligned} &P_k = \frac{1}{\rm SN} + a(t)\frac{\rm SN+1-2k}{\rm SN(SN+1)},\ k = 1, 2, \cdots, {\rm SN};\\ &a(t) = 0.2 + \frac{3t}{4{\rm Max\_Cycle}},\ t = 1, 2, \cdots, {\rm Max\_Cycle}. \end{aligned} \tag{5}$$

其中: $a(t)$ 是自适应参数, Max_Cycle 是最大迭代次数. 算法进化初期阶段, 为了保持种群的多样性, $a(t)$ 的值比较小; 进化后期阶段, 个体差异性较小, 导致种群多样性降低, 进而导致算法有可能陷入局部最优而跳不出来, 从而出现停滞现象. 为了避免搜索陷入停滞状态, $a(t)$ 的值应比较大.

2.3　基于局部搜索的人工蜂群算法

基于局部搜索的人工蜂群算法的基本思路是: 首先, 利用局部搜索算子对目前找到的最优位置进行局部搜索, 并同时改变多个变量, 以加强算法的开采能力, 加快算法的收敛速度; 其次, 为了避免进化过程中超常个体使算法陷入局部最优, 维持种群多样性, 采用基于排序的选择概率, 使算法具有更强的鲁棒性.

改进算法的具体步骤如下.

Step 1: 随机产生初始解集 $x_{ij}, i = 1, 2, \cdots, {\rm SN}, j = 1, 2, \cdots, D$;

Step 2: 计算解的适应度;

Step 3: 引领蜂按式 (3) 搜索新蜜源 v_i, 计算其适应度;

Step 4: 如果新蜜源 v_i 的适应度比 x_i 大, 则用 v_i 取代 x_i, 否则保留 x_i;

Step 5: 根据式 (5) 计算蜜源的选择概率 P;

Step 6: 跟随蜂根据概率 P 选择蜜源, 按式 (3) 搜索新蜜源 v_i, 计算其适应度;

Step 7: 如果新蜜源 v_i 的适应度比 x_i 大, 则用 v_i 取代 x_i, 否则保留 x_i;

Step 8: 记忆最优蜜源;

Step 9: 如果达到迭代间隔 $t_{\rm step}$, 则利用局部搜索算子对当前最优位置进行局部搜索, 更新最优位置;

Step 10: 判断是否存在要放弃的蜜源, 若存在, 则该处的引领蜂变为侦查蜂;

Step 11: 如果满足终止条件, 则算法终止, 输出结果, 否则转 Step 3.

3　仿真实验及结果分析

为了测试改进人工蜂群 (LRABC) 算法的性能, 对 12 个基准函数[13]进行测试, 表 1 给出了测试函数的名称、维数、搜索空间和理论最优值. 其中: $f_1 \sim f_6$ 为单峰函数, $f_7 \sim f_{12}$ 为多峰函数.

表 1　测试函数的名称、维数、搜索空间和最优值

function	名称	维数	搜索空间	最优值
f_1	Sphere	30	$[-100, 100]$	0
f_2	Schwefel 2.22	30	$[-10, 10]$	0
f_3	Schwefel 2.21	30	$[-100, 100]$	0
f_4	Rosenbrock	30	$[-30, 30]$	0
f_5	Step	30	$[-100, 100]$	0
f_6	Quadric	30	$[-1.28, 1.28]$	0
f_7	Schwefel	30	$[-500, 500]$	$-12\,569.5$
f_8	Rastrigin	30	$[-5.12, 5.12]$	0
f_9	Ackley	30	$[-32, 32]$	0
f_{10}	Griewank	30	$[-600, 600]$	0
f_{11}	Penalized 1	30	$[-50, 50]$	0
f_{12}	Penalized 2	30	$[-50, 50]$	0

将改进算法与基本 ABC 算法[14]进行比较. f_1, $f_5, f_7 \sim f_{12}$ 最大函数评价次数为 100 000, $f_2 \sim f_4, f_6$ 的最大函数评价次数为 200 000. 改进算法和基本 ABC 算法中的参数采用文献 [15] 中的设置方案, 即种群规模为 $50({\rm SN} = 25)$, ${\rm limit} = {\rm SN} \times D$. 改进算法的其他参数设置按目前通用的方法根据实验测试确定. 图 1 给出了 $t_{\rm step} = 20$ 时, E 和 N 的不同组合对函数 f_1, f_2, f_6, f_9 计算结果的影响. 从图 1 可以看出, $E = 50, N = 5$ 时, 求解精度最高. 这是因为: E, N 取值较小时, 不能对当前解进行充分开发; E, N 取值较大时, 增加了局部搜索算子的无效迭代次数.

将 2 种算法独立运行 30 次, 测试结果如表 2 所示. 其中: Best 表示 30 次独立实验的最好值, Mean 表示平均最优值, Worst 表示 30 次独立实验的最差值, Std.Dev 表示标准差. Mean 反映了算法在给定最大评价次数时算法的求解精度, Std.Dev 反映了算法的稳定性.

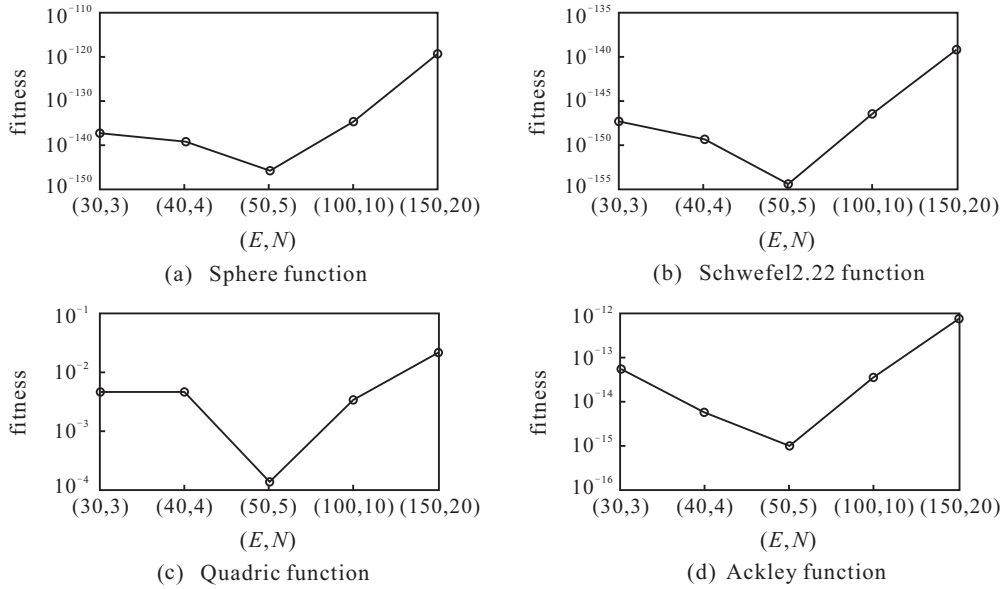

图 1 (E,N) 对计算结果的影响

表 2 2 种算法对 12 个函数的测试结果比较

function	algorithm	Best	Mean	Worst	Std.Dev
f_1	ABC	3.04e-016	4.97e-016	7.20e-016	8.74e-017
	LRABC	2.53e-146	2.94e-146	5.84e-146	3.37e-146
f_2	ABC	9.69e-016	1.27e-015	1.61e-015	1.48e-016
	LRABC	1.84e-155	1.73e-155	2.87e-155	6.61e-155
f_3	ABC	0.177 1	0.474 8	1.254 0	0.215 3
	LRABC	2.85e-160	7.81e-149	6.39e-148	1.95e-148
f_4	ABC	0.061 3	0.120 7	0.185 5	0.091 6
	LRABC	0.074 1	0.131 0	0.189 0	0.081 5
f_5	ABC	0	0	0	0
	LRABC	0	0	0	0
f_6	ABC	0.022 0	0.032 5	0.041 6	0.009 9
	LRABC	1.26e-004	2.16e-004	5.23e-004	2.21e-004
f_7	ABC	−12 498.42	−12 469.54	−12 435.61	9.04e-02
	LRABC	−12 569.48	−12 569.45	−12 569.42	2.78e-13
f_8	ABC	0.995 1	1.064 1	1.152 5	3.521 9
	LRABC	0	0	0	0
f_9	ABC	5.55e-010	7.44e-010	8.67e-010	1.66e-010
	LRABC	8.88e-016	8.91e-016	9.21e-016	0
f_{10}	ABC	1.85e-016	2.36e-016	4.44e-016	2.31e-016
	LRABC	0	0	0	0
f_{11}	ABC	5.34e-016	6.01e-016	7.20e-016	1.02e-016
	LRABC	1.57e-32	2.63e-32	3.74e-32	5.09e-33
f_{12}	ABC	3.19e-016	4.29e-016	4.90e-016	9.55e-017
	LRABC	6.31e-030	7.22e-030	8.04e-030	7.34e-030

由表 2 数据可看出, 在单峰函数测试中, 除函数 f_4 外, LRABC 算法在解的精度和稳定性两方面都优于基本 ABC 算法. 特别地, 对于函数 f_5, 2 种算法都能得到理论最优值, 这是因为函数 f_5 的最优解是一个区域, 而不是一个点. 在多峰函数的测试中, 对于函数 f_8 和 f_{10}, LRABC 算法能得到理论最优值, 而基本 ABC 算法所得到解的精度较低. 对于其他的复杂多峰函数, LRABC 算法也能得到较高精度的解. 为了更直观地反映 LRABC 算法的收敛性能, 图 2 为 2 种算法对部分测试函数的收敛曲线. 从图 2 可以看出, LRABC 算法在迭代初期就表现出良好的性能, 测试函数的收敛曲线下降速度很快, 而且能收敛到高精度的解.

为进一步说明 LRABC 算法具有较高的计算精度, 将 LRABC 算法与 CABC、GABC、RABC 和 IABC 算法进行比较. CABC、GABC、RABC 和 IABC 算法的参数设置分别参见文献 [16, 6, 17, 15]. 表 3 给出了 5 种算法的比较结果, 其中 FEs 表示函数评价次数. 由表 3 的数据对比可以看出, 除函数 f_{11} 外, 对于函数 f_1, f_2, f_3, f_7, f_9, LRABC 算法的收敛精度最高. 特别地, LRABC 算法能够得到函数 f_3 的理论最优值.

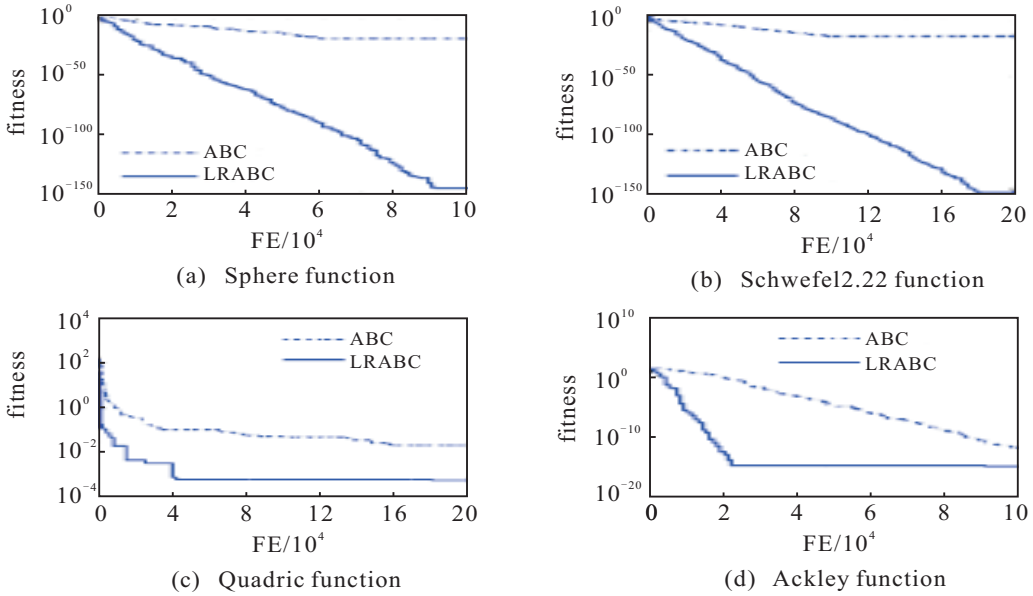

图 2　4 个函数的收敛曲线

表 3　5 种算法对 9 个函数的测试结果比较

function	FEs		CABC[16]	GABC[6]	RABC[17]	IABC[15]	LRABC
f_1	1.5×10^5	Mean	2.3e-40	3.6e-63	9.1e-61	5.34e-178	9.48e-208
		Std.Dev	1.7e-40	5.7e-63	2.1e-60	0	0
f_2	2.0×10^5	Mean	3.5e-30	4.8e-45	3.2e-74	8.82e-127	1.81e-154
		Std.Dev	4.8e-30	1.4e-45	2.0e-73	3.49e-126	2.47e-155
f_3	5.0×10^5	Mean	6.1e-03	3.6e-06	2.8e-02	4.98e-38	0
		Std.Dev	5.7e-03	7.6e-07	1.7e-02	8.59e-38	0
f_5	1.5×10^5	Mean	0	0	0	0	0
		Std.Dev	0	0	0	0	0
f_7	3.0×10^5	Mean	3.8e-02	1.1e-02	3.6e-02	2.42e-03	5.20e-04
		Std.Dev	5.2e-01	5.3e-02	6.8e-03	5.57e-04	1.80e-04
f_8	5.0×10^4	Mean	1.3e-00	1.5e-10	2.3e-02	0	0
		Std.Dev	2.7e-00	2.7e-10	5.1e-01	0	0
f_9	5.0×10^4	Mean	1.0e-05	1.8e-09	9.6e-07	3.87e-14	8.88e-16
		Std.Dev	2.4e-06	7.7e-13	8.3e-07	8.52e-15	7.34e-16
f_{10}	5.0×10^4	Mean	1.2e-04	6.0e-13	8.7e-08	0	0
		Std.Dev	4.6e-04	7.7e-13	2.1e-08	0	0
f_{11}	5.0×10^4	Mean	4.2e-11	8.5e-20	5.4e-16	1.57e-32	6.65e-16
		Std.Dev	5.3e-11	4.1e-20	2.8e-16	2.73e-48	5.16e-16

另外, LRABC 算法在收敛速度方面也有很大的提高. 为了说明这一点, 将 LRABC 算法与 APSO[18]、ABC 和 GABC 算法进行比较. APSO、ABC 和 GABC 算法的参数设置分别参见文献 [18, 15, 6]. 表 4 给出了 4 种算法达到精度要求时所需的函数评价次数. 从表 4 可以看出, LRABC 算法达到精度要求时所需的函数评价次数最少.

表 4　4 种算法的收敛速度比较

function	ABC	GABC	APSO	LRABC	acceptance
f_1	6 401	5 241	7 074	2 001	0.01
f_2	11 201	7 581	7 900	1 501	0.01
f_5	5 461	3 801	4 902	1 001	0
f_6	–	–	78 117	4 952	0.01
f_7	4 621	4 261	5 159	4 061	−10 000
f_8	3 201	2 861	3 531	1 913	50
f_9	14 881	8 581	2 905	2 417	0.01
f_{10}	16 301	7 941	40 736	1 411	0.01
f_{11}	5 429	4 941	7 568	4 475	0.01

注: “–”表示达到最大评价次数时未满足精度要求.

4　结　论

为了加快基本 ABC 算法的收敛速度并避免早熟收敛现象, 本文提出了基于局部搜索的人工蜂群算法. 通过对目前找到的最优蜜源进行局部搜索, 提高了算法的开采能力, 从而加快了算法的收敛速度; 同时, 采用基于排序的选择概率维持种群的多样性, 避免了早熟收敛. 对 12 个标准测试函数的仿真结果表明, 该算法在收敛速度和计算精度方面都有显著的提高. 进一步的研究可将算法推广到其他应用领域, 如参数优化和典型的线性系统逼近工程优化等问题.

参考文献(References)

[1] Karaboga D. An idea based on honey bee swarm for numerical optimization[R]. Kayseri: Engineering Faculty Computer Engineering Department, Ereiyes University, 2005.

[2] Karaboga D, Akay B. Artificial bee colony(abc) algorithm on training artificial neural networks[C]. 2007 IEEE 15th Signal Proc and Communications Applications. Eskisehir: IEEE, 2007: 1-4.

[3] Karaboga D, Ozturk C. Fuzzy clustering with artificial bee colony algorithm[J]. Scientific Research and Essays, 2010, 5(14): 1899-1902.

[4] Ozturk C, Karaboga D, Gorkemli B. Probabilistic dynamic deployment of wireless sensor networksby artificial bee colony algorithm[J]. Sensors, 2011, 11(6): 6056-6065.

[5] 罗钧, 李研. 具有混沌搜索策略的蜂群优化算法[J]. 控制与决策, 2010, 25(12): 1913-1916.
(Luo J, Li Y. Artificial bee colony algorithm with chaotic-search strategy[J]. Control and Decision, 2010, 25(12): 1913-1916.)

[6] Zhu G P, Kwong S. Gbest-guided artificial bee colony algorithm for numerical function optimization[J]. Applied Mathematics and Computation, 2010, 217(7): 3166-3173.

[7] El-Abd M. A hybrid ABC-SPSO algorithm for continuous function optimization[C]. IEEE Symposium on Swarm Intelligence. Paris: IEEE, 2011: 1-6.

[8] Basturk B, Karaboga D. A modified artificial bee colony algorithm for real-parameter optimization[J]. Information Sciences, 2012, 192(1): 120-142.

[9] Hamzacebi C, Kutay F. Continuous functions minimization by dynamic random search technique[J]. Applied Mathematical Modelling, 2007, 31(10): 2189-2198.

[10] 高卫峰, 刘三阳. 一种高效粒子群优化算法[J]. 控制与决策, 2011, 26(8): 1158-1162.
(Gao W F, Liu S Y. An efficient particle swarm optimization[J]. Control and Decision, 2011, 26(8): 1158-1162.)

[11] Hamzacebi C. Improving genetic algorithms' performance by local search for continuous function optimization[J]. Applied Mathematics and Computation, 2008, 196(1): 309-317.

[12] Bao L, Zeng J C. Comparison and analysis of the selection mechanism in the artificial bee colony algorithm[C]. The 9th Int Conf on Hybrid Intelligent Systems. Shenyang: IEEE Press, 2009: 411-416.

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

[14] Karaboga D, Basturk D, On the performance of artificial bee colony(ABC) algorithm[J]. Applied Soft Computing, 2008, 8(1): 687-697.

[15] Gao W F, Liu S Y. Improved artificial bee colony algorithm for global optimization[J]. Information Processing Letters, 2011, 111(17): 871-882.

[16] Alatas B. Chaotic bee colony algorithms for global numerical optimization[J]. Expert Systems with Applications, 2010, 37(8): 5682-5687.

[17] Kang F, Li J J, Ma Z Y. Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions[J]. Information Sciences, 2011, 181(16): 3508-3531.

[18] Zhan Z H, Zhang J, Li Y, et al. Adaptive particle swarm optimization[J]. IEEE Trans on Systems, Man, and Cybernetics, 2009, 39(6): 1362-1381.

(责任编辑: 曹洪式)