

来自大自然的寻优策略:像蝙蝠一样感知*

刘长平^{1,2}, 叶春明¹, 刘满成³

(1. 上海理工大学 管理学院, 上海 200093; 2. 淮阴工学院, 江苏 淮安 223001; 3. 中国人民大学 商学院, 北京 100872)

摘要: 蝙蝠算法是受自然界中的蝙蝠通过回声定位进行搜寻、捕食猎物行为的启发,并将多智能体系统与进化机制相结合发展而来的优化方法。作为一种新颖的仿生群体智能优化算法,分析了蝙蝠算法的仿生原理、优化机理及特点,对算法优化过程进行了定义。通过标准算例对蝙蝠算法在连续空间和离散空间的优化性能进行了仿真测试,结果表明该算法在函数优化和组合优化方面应用的可行性和有效性,具有良好的应用前景。

关键词: 蝙蝠算法; 优化机理; 函数优化; 组合优化; 仿真测试

中图分类号: TP301.6 **文献标志码:** A **文章编号:** 1001-3695(2013)05-1320-03

doi:10.3969/j.issn.1001-3695.2013.05.010

Optimization strategy from nature: perceive as bat

LIU Chang-ping^{1,2}, YE Chun-ming¹, LIU Man-cheng³

(1. College of Management, University of Shanghai for Science & Technology, Shanghai 200093, China; 2. Huaiyin Institute of Technology, Huaian Jiangsu 223001, China; 3. School of Business, Renmin University of China, Beijing 100872, China)

Abstract: Inspired by the echolocation behavior of bats and combined multi-agent system with evolution mechanism, bat algorithm (BA) was developed as a novel bionic swarm intelligence optimization method. This paper analyzed the bionic principle and trait of BA as well as defined the mechanism of optimization by formulation. It tested the BA by benchmarks of continuous space optimization and discrete space optimization, simulation results show that the new bio-inspired algorithm has better feasibility and validities for function optimization and combinatorial optimization.

Key words: bat algorithm (BA); optimization mechanism; function optimization; combinatorial optimization; simulation test

0 引言

工程技术、经济管理等领域中的多数决策问题都可归结为优化问题,其求解质量决定了决策方案的优劣。传统的优化方法如线性规划、动态规划等,或者对优化问题有条件限制(如要求目标函数连续、可导),或者由于优化问题的属性(如NP问题)只能求解小规模问题,对于大规模、强约束、非线性等优化问题难以求解或效果不佳。受自然规律和生物群体智能行为的启发演变而来的智能优化方法,如遗传算法、蚁群算法、粒子群算法等,在上述传统优化技术禁区则表现出不俗性能。近年来学者们又陆续提出了一些新颖的仿生群智能算法,如鱼群算法^[1]、蜂群算法^[2,3]、萤火虫算法^[4]等,在科学计算和工程技术领域中显示出独特的特点和应用效果,受到学术界和工程界的日益关注。

蝙蝠能够在完全黑暗的环境中躲避头发丝粗细的障碍物且能捕食猎物,这得益于蝙蝠所拥有的令人惊异的回声定位能力。蝙蝠通过探测发出的超声波回波的时间延迟,利用回波到达双耳的时间差、回波音强的变化建立起周围环境的三维场

景,不仅能探测猎物的距离,还能识别其体型特征、方位和角度。蝙蝠在复杂环境中精确定位、捕食的情形与算法在高维复杂空间寻优的状况类似,这为模拟前者的生物学机理进行优化带来了启发。蝙蝠算法是一种源于大自然中蝙蝠通过超声波搜索、捕食猎物的生物学特性发展而来的随机型全局优化算法,最早由 Yang^[5,6]提出。本文分析了算法的仿生原理、优化机理和特点,对算法在连续空间和离散空间的优化性能进行了仿真测试,验证了算法在函数优化和组合优化方面的可行性和有效性。

1 蝙蝠算法的优化机理和特点

1.1 算法的生物学机理及数学模拟

依靠回声定位进行捕食的蝙蝠,在搜寻猎物时通常每秒发出大约10~20个、音强可达110 dB的超声波脉冲,脉冲音强在搜寻猎物时通常为最大,在飞向猎物时逐渐减小,同时脉冲频率逐渐增加,达到每秒发射约200个脉冲。脉冲音强大有助于超声波传播更远的距离,脉冲频率高有助于精确掌握猎物不断变化的空间位置。蝙蝠发出的回声定位声波一般由单谐波

收稿日期: 2012-08-30; **修回日期:** 2012-10-14 **基金项目:** 国家自然科学基金资助项目(71271138); 国家教育部人文社会科学规划基金资助项目(10YJA630187); 上海市教委科研创新基金资助项目(12ZS133); 国家教育部高校博士学科点专项科研基金资助项目(20093120110008); 上海市重点学科建设项目(S30504)

作者简介: 刘长平(1974-),男,河南洛阳人,讲师,博士研究生,主要研究方向为管理信息系统、系统工程(lcp_mail@163.com); 叶春明(1964-),男,安徽宣城人,教授,博导,博士,主要研究方向为系统工程、工业工程; 刘满成(1974-),男,安徽六安人,副教授,博士研究生,主要研究方向为管理信息系统。

或多谐波宽频带的调频信号组成,频率通常在 25 ~ 100 kHz。每个谐波频率由高到低,下降较快,多谐波、宽频带的调频声有利于确定复杂环境和猎物的精细结构。

如果把自然界中的蝙蝠个体模拟为分布在搜索空间中的多智能体,用待优化问题的目标函数值来衡量个体所处位置的优劣,将蝙蝠搜索目标和飞行移动过程类比为用好的可行解取代较差可行解的更新过程,则蝙蝠利用超声波来探测、定位、捕食猎物等一系列行为可以用下述方程^[7]来模拟。蝙蝠搜寻猎物时的飞行速度和位置更新公式为

$$v_{id}^t = v_{id}^{t-1} + (x_{id}^t - x_d^*) \times f_i \quad (1)$$

$$x_{id}^t = x_{id}^{t-1} + v_{id}^t \quad (2)$$

其中: v_{id}^{t-1} 、 v_{id}^t 分别表示蝙蝠*i*在*t*-1和*t*时刻第*d*维的飞行速度; x_{id}^t 表示蝙蝠*i*在*t*时刻的空间位置; x_d^* 表示在当前搜索过程中群体最佳蝙蝠所处位置;下标*d*表示搜索空间维度; f_i 为蝙蝠*i*搜寻猎物时使用的脉冲频率, $f_i \in [f_{\min}, f_{\max}]$, $[f_{\min}, f_{\max}]$ 为搜索脉冲频率范围。

根据蝙蝠的生物学机理可知,蝙蝠在搜寻猎物时,刚开始发射超声波脉冲的音强较大而频率较低,有助于在更广泛的空间搜索,一旦发现了猎物,就逐渐减小脉冲音强同时增加脉冲发射次数,以利于精确掌握猎物的空间位置,用式(3)(4)来模拟蝙蝠这种搜索特性。

$$r^{t+1} = r^0 [1 - \exp(-\gamma \times t)] \quad (3)$$

$$A^{t+1} = \alpha \times A^t \quad (4)$$

其中: r^0 表示最大脉冲频率; r^{t+1} 表示蝙蝠在*t*+1时刻的脉冲频率; γ 表示脉冲频率增加系数,是一个大于零的常数; A^t 表示*t*时刻蝙蝠发射脉冲的音强; α 是脉冲音强衰减系数,为[0,1]上的常数。

1.2 蝙蝠算法的特点

蝙蝠算法中,脉冲频率增加系数 γ 、脉冲音强衰减系数 α 对算法性能有重要影响。以下对这两个参数进行简要分析。

蝙蝠算法中,个体当前位置的更新方式由判断式 $\text{rand} > r_i$ 来决定, rand 是[0,1]上的随机数, r_i 是蝙蝠个体*i*当前的搜索脉冲频率,根据式(3)计算。若判断式为真,第*i*只蝙蝠当前位置由群体中当前最佳位置的邻域产生;反之,则根据式(2)计算产生。从定义中不难看出, $\lim_{t \rightarrow +\infty} r = r^0$,当*t*一定时, γ 影响脉冲频率*r*的变化。根据式(3), γ 取值越大, $r \rightarrow r^0$ 越快,蝙蝠个体当前位置更多由式(2)计算产生,算法进行全局搜索但收敛速度较慢;反之,算法大多在当前全局最优位置的邻域搜索,收敛速度较快但易被局部极值吸引。

脉冲音强衰减系数 α 决定脉冲音强的衰减速度,而脉冲音强*A*决定了蝙蝠个体接受更新位置的概率。由定义可知, $\lim_{t \rightarrow +\infty} A = 0$,当*t*一定时, α 取值大, $A \rightarrow 0$ 过程缓慢,算法能够充分搜索解空间,所得到的性能会比较满意但计算时间较长;反之,算法收敛快但易早熟。这里 α 所起作用与模拟退火算法中温度衰减系数类似。通过合理调整 γ 、 α ,可使算法得以兼顾全局搜索和局部搜索能力,有效平衡算法收敛速度和搜索精度。

2 算法流程

综上所述,蝙蝠算法流程如下:

a) 初始化算法基本参数。设置蝙蝠数目*m*,搜索脉冲频率

范围 $[f_{\min}, f_{\max}]$,最大脉冲频率 r^0 ,最大脉冲音强*A*,音强衰减系数 α ,频率增加系数 γ ,最大迭代次数 MaxT 或搜索精度 ε 。

b) 随机初始化蝙蝠的位置 $x_i (i = 1, \dots, m)$,找出当前群体中处于最佳位置 x^* 的个体。

c) 初始化搜索脉冲频率 f_i ,根据式(1)计算蝙蝠的飞行速度 v_i ,根据式(2)更新蝙蝠的空间位置 x_i 。

d) 生成随机数 rand1。如果 $\text{rand1} > r_i$,则对处在最佳位置的蝙蝠进行随机扰动,用扰动后的位置代替当前蝙蝠个体*i*的位置;否则,按照式(2)更新蝙蝠个体*i*的空间位置。

e) 生成随机数 rand2。如果 $\text{rand2} > A_i$,并且蝙蝠当前所处位置得到改善,则移动至更新后的位置。

f) 若更新位置后蝙蝠*i*优于当前最佳蝙蝠位置,根据式(3)(4)更新脉冲频率*r*和脉冲音强*A*。

g) 对蝙蝠群体进行评估,找出当前最佳蝙蝠以及所处空间位置。

h) 当满足搜索精度或达到最大搜索次数则转入 i); 否则转入 c),进行下一次搜索。

i) 输出全局极值点和最优个体值。

rand1 与 rand2 是[0,1]上均匀分布的随机数。算法的时间复杂度为 $O(m^2 \cdot \text{MaxT})$ 。其中:*m*为蝙蝠数目,MaxT 为最大迭代次数。

3 仿真实验

本文分别从函数优化和组合优化两方面来验证蝙蝠算法的优化性能,选取相应的标准测试问题进行仿真测试。

3.1 函数优化问题仿真测试

为了验证蝙蝠算法在连续空间的优化性能,选取了六个标准测试函数来进行仿真测试,并与基本粒子群算法和遗传算法进行对比。

蝙蝠算法中涉及的各种参数取值目前尚无确切的理论依据,本文所设置的参数值是根据反复实验获得的经验值来确定的。蝙蝠算法中,搜索脉冲频率范围为 $[-1, 1]$,最大脉冲频率 $r^0 = 0.5$,最大脉冲音强 $A = 0.25$,脉冲音强衰减系数 $\alpha = 0.95$,脉冲频率增加系数 $\gamma = 0.05$ 。粒子群算法中,采用文献[8]中所提出的线性减少的惯性权重 $W_{\max} = 0.9$, $W_{\min} = 0.4$,学习因子 $C_1 = C_2 = 1.4962$ 。遗传算法中,染色体采用二进制编码和精英保留策略,精英个体保留比例为5%,交叉概率 $p_c = 0.95$,变异概率 $p_m = 0.05$ 。上述三种算法最大迭代次数均为 MaxT = 300,群体数目根据搜索空间的维数分别取 $m = 20, 50, 80$,每种算法独立运行20次,结果如表1所示。其中,GA表示标准遗传算法,PSO表示基本粒子群算法,BA表示蝙蝠算法。

表1 标准测试函数仿真测试结果

函数	维数	种群数	平均最优值/(寻优率)		
			GA	PSO	BA
$f_1(x)$	2	20	-0.9799(0%)	-0.9650(41%)	-0.9922(83%)
$f_2(x)$	2	20	-182.0900(0%)	-167.2564(47.7%)	-181.1519(93.3%)
$f_3(x)$	5	20	35.1359(0%)	29.8385(0%)	1.4686(74.3%)
	10	50	48.5893(0%)	41.4450(0%)	37.9843(36.3%)
$f_4(x)$	5	20	4.8659(0%)	8.2144(0%)	3.3081(34.3%)
	10	50	40.2196(0%)	19.1414(0%)	10.1534(37.3%)
$f_5(x)$	5	20	412.4478(0%)	246.6579(0%)	131.0404(29.7%)
	10	50	786.0570(0%)	358.7313(0%)	249.1851(10%)
$f_6(x)$	10	50	352.0130(0%)	112.3518(0%)	32.6190(0%)
	20	80	892.8142(0%)	340.2081(0%)	70.7742(0%)

测试函数如表2所示。测试表明,对于函数 f_1 、 f_2 ,BA与PSO算法均收敛到了最优解,但BA的寻优精度和寻优率均高于PSO算法,寻优性态如图1、2所示。函数 f_3 、 f_4 、 f_5 均是高维、多峰且有大量局部极值的多模态函数,GA、PSO两种算法均未搜索到最优解,BA在不同维度的搜索空间不仅找到了最优解,而且寻优精度和收敛速度也高于其他两种算法,如图3~5所示。函数 f_6 虽然是连续单峰函数,但取值区间走势平坦,只为算法提供少量信息,极难收敛到全局最优值,测试中三种

算法均未搜索到最优解,反映出这三种算法对于高维函数的优化效果还不理想,但从平均寻优值看出BA寻优精度远高于其他两种算法,考虑到采用的均是基本算法,优化结果是可以接受的,如图6所示(图6纵轴采用的是对数坐标)。综合来看,在相同维数和群体数条件下,BA在寻优精度、寻优率和收敛速度方面均优于GA、PSO算法,在多数情况下以较小的种群规模和较少的迭代次数就可以捕获函数最优解,表现出较强的全局寻优能力和很高的搜索精度。

表2 标准测试函数

函数名	函数表达式	搜索空间	理论最优解
Schaffer F6	$f_1(x) = \frac{\sin^2 \sqrt{x_1^2 + x_2^2} - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2} - 0.5$	$[-100, 100]$	$f_1(0, 0) = -1$
Shubert	$f_2(x) = \sum_{i=1}^5 i \times \cos((i+1) \times x_1 + i) \times \sum_{j=1}^5 j \times \cos((j+1) \times x_2 + j)$	$[-10, 10]$	$f_2(x^*) = -186.7309$
Rastrigin	$f_3(x) = \sum_{i=1}^D [x_i^2 - 10 \cos(2\pi x_i) + 10]$	$[-5.12, 5.12]$	$f_3(0, \dots, 0) = 0$
Griewank	$f_4(x) = \frac{1}{4000} \sum_{i=1}^D (x_i^2) - \prod_{i=1}^D \cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-600, 600]$	$f_4(0, \dots, 0) = 0$
Schwefel	$f_5(x) = 418.9829 \times D - \sum_{i=1}^D x_i \sin(x_i ^{1/2})$	$[-500, 500]$	$f_5(420.96, \dots, 420.96) = 0$
Rosenbrock	$f_6(x) = \sum_{i=1}^{D-1} [(x_i - 1)^2 + 100(x_i^2 - x_{i+1})^2]$	$[-2.048, 2.048]$	$f_6(1, \dots, 1) = 0$

注: x^* 表示在搜索空间内最优位置不唯一

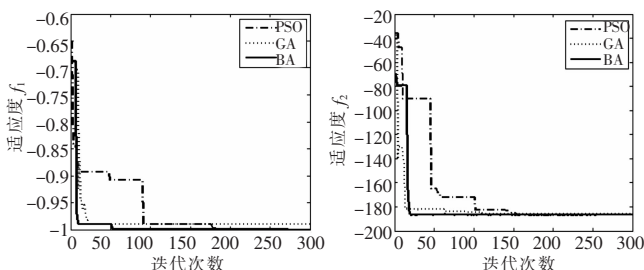


图1 Schaffer F6函数测试寻优曲线($m=20$)

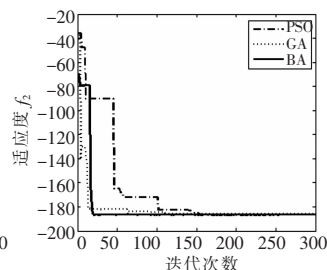


图2 Shubert函数测试寻优曲线($m=20$)

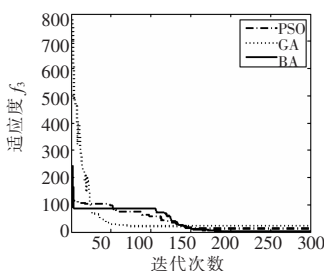


图3 Rastrigin函数测试寻优曲线($D=10, m=50$)

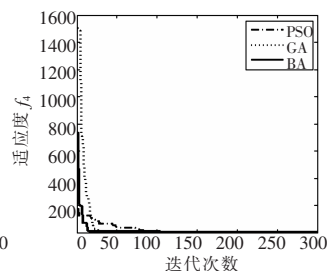


图4 Griewank函数测试寻优曲线($D=10, m=50$)

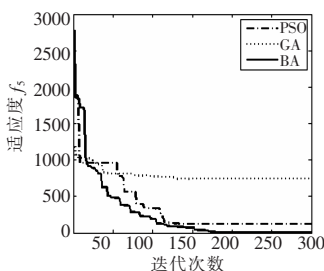


图5 Schwefel函数测试寻优曲线($D=10, m=50$)

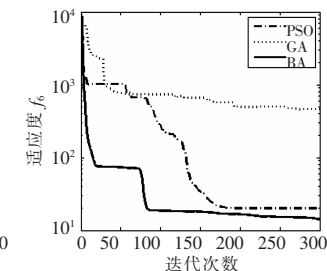


图6 Rosenbrock函数测试寻优曲线($D=20, m=80$)

3.2 组合优化问题仿真测试

组合优化问题就是在给定的约束条件下找出离散事件的最优变量组合问题,如工件加工排序问题、车辆路径问题、背包问题等均是典型的组合优化问题。这些问题有很强的工程背景和实际应用价值,但在计算理论中属于NP难题,最优化求解非常困难。

本文选取了生产调度问题中的置换流水线调度问题(permutation flow-shop scheduling problem, PFSP)作为蝙蝠算法的测试用例,采用由Carlier设计的Car类问题^[9]作为测试数据,并与基本粒子群算法进行对比。与函数优化中可以直接用个体位置代表问题的解不同,应用蝙蝠算法求解PFSP首先需要采用合理的编码方式来表示调度问题的解。针对PFSP的特点,本文采用基于最小位置值规则的随机键编码方式^[10],将蝙蝠个体的一个连续位置向量转换为机器上一个工件的加工顺序,从而计算个体所对应的调度解的目标值。通过这种转换,既能保证调度解的可行性,又无须修改蝙蝠算法的进化操作。算法中群体数目 $m=40$,最大搜索次数 $\text{MaxT}=200$,其余参数设置同3.1节。每种算法独立运行20次,测试结果如表3所示。表3中 P 代表问题类型; n, m 分别表示工件数与机器数; C^* 表示对应调度问题的最小化完工时间;SR(success rate)是寻优成功率,代表20次独立测试中找到 C^* 值的比率;BRE(best relative error)、ARE(average relative error)、WRE(worst relative error)分别表示最优、平均、最差相对误差。

表3 组合优化问题测试结果

P	n, m	C^*	BA				PSO			
			BRE	ARE	WRE	SR/%	BRE	ARE	WRE	SR/%
Car1	11, 5	7 038	0	0	0	100	0	0.07	1.53	95
Car2	13, 4	7 166	0	0.25	2.93	90	0	2.13	4.82	40
Car3	12, 5	7 312	0	0.71	3.16	50	0.87	2.74	4.18	0
Car4	14, 4	8 003	0	0	0	100	0	1.48	5.24	45
Car5	10, 6	7 720	0	1.02	1.65	15	0	1.65	4.27	10
Car6	8, 9	8 505	0	0.80	2.79	35	0	1.57	3.39	15
Car7	7, 7	6 590	0	0.24	0.80	80	0	0.42	2.55	75
Car8	8, 8	8 366	0	0.32	1.96	75	0	0.76	3.84	60

从表3数据可以看出,对于上述Car类测试问题,BA均能找到已知下界值,而且BA寻优成功率高于对应的PSO算法,反映出BA具有良好的全局收敛性能。从ARE和WRE数据来看,BA所求得调度方案的平均质量远高于PSO算法。由于在寻优过程中并没有对蝙蝠算法采用任何局部搜索策略,显示出BA在离散空间也具有优良的进化机制,使得BA在组合优化领域也具有有良好的应用前景。(下转第1356页)

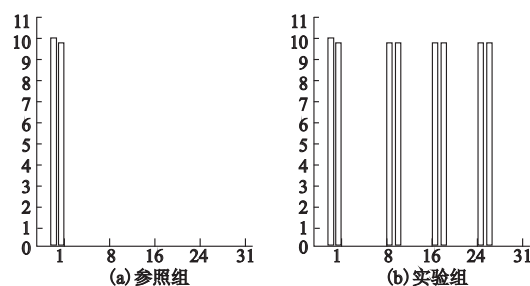


图6 I组闪存擦除情况

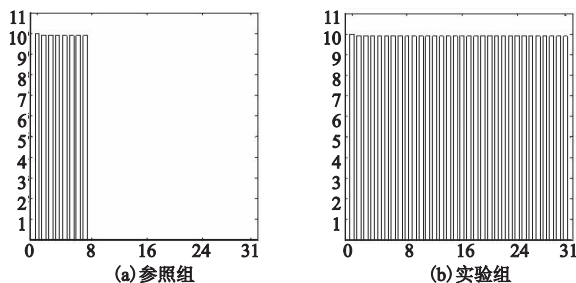


图7 II组闪存擦除情况

main 能被等概率擦写。同时,两种情况下的 PBI 比较见表 1。

表1 两种情况下 PBI 比较

	I 参照组	I 实验组	II 参照组	II 实验组
PBI	12.5%	25%	25%	99.9%

显然,无论在哪种情况下,在满足态势信息快速存取的前提下该算法使得参与擦写操作的物理块更多、闪存的磨损均衡效果更好一些。

4 结束语

应用于战场环境下的移动终端要求态势信息能够快速存取,而有限的存储空间需要不断地将过时的信息擦除。结合该具体应用背景,本文设计了一种闪存磨损均衡算法。该算法基于分组循环擦除的思想,在满足态势信息能够快速存取、擦除最过时信息的前提下使得更多的空间循环参与擦写。仿真实

验结果表明该算法是有效的。下一步的工作是在某型号终端上验证并完善该算法。

参考文献:

- [1] CHANG Li-pin, HUANG Lu-chun. A low-cost wear-leveling algorithm for block-mapping solid-state disks[C]//Proc of Conference on Languages, Compilers and Tools for Embedded Systems. New York: ACM Press, 2011: 31-40.
- [2] GAL E, TOLEDO S. Algorithms and data structures for flash memories[J]. ACM Computing Surveys, 2005, 37(2): 138-163.
- [3] 黄德才, 邢春波, 吕莲. 闪存磨损均衡算法综述[J]. 浙江工业大学学报, 2009, 37(1): 73-78.
- [4] LOFGREN K, NORMAN R. Wear leveling techniques for flash E²PROM systems; USA, US007353325B2[P]. 2008-04-01.
- [5] CHANG Li-pin. On efficient wear leveling for large-scale flash-memory storage systems[C]//Proc of ACM Symposium on Applied Computing. [S. l.]: ACM Press, 2007: 1126-1130.
- [6] JUNG D, CHAE Y, JO H, et al. A group-based wear-leveling algorithm for large-capacity flash memory storage systems[C]//Proc of the 2007 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems. New York: ACM Press, 2007: 160-164.
- [7] 邢春波, 杨良怀, 龚卫华, 等. 一种有效的混合式闪存磨损均衡算法[J]. 小型微型计算机系统, 2009, 30(9): 1903-1906.
- [8] CHANG Y, HSIEH J, KUO T. Endurance enhancement of flash memory storage systems: an efficient static wear leveling design[C]//Proc of the 44th Annual Conference on Design Automation. New York: ACM Press, 2007: 212-217.
- [9] LEE C C. System and method for managing blocks in flash memory; US, 0204187[P]. 2005-09-15.
- [10] 张骏, 樊晓桢, 刘松鹤. 一种 flash 存储器静态负载平衡策略[J]. 计算机应用, 2006, 26(5): 1205-1207.
- [11] 邢春波. 闪存磨损均衡算法研究[D]. 杭州: 浙江工业大学, 2009.
- [12] 孙文静, 李明强, 舒继武. Flash 存储技术[J]. 计算机研究与发展, 2010, 47(4): 716-726.

(上接第 1322 页)

4 结束语

本文对蝙蝠算法进行了理论分析,该算法吸收了自然界中蝙蝠通过超声波搜索、捕食猎物的行为特性,利用进化的方式来实现智能体的行为以达到优化的目的。算法实现简单,具有本质并行性。通过选取的标准算例进行仿真测试,表明了蝙蝠算法在连续空间和离散空间优化的可行性和有效性,具有良好的应用前景。由于蝙蝠算法优化理论和应用研究还处于初始阶段,许多问题还有待于人们不断地探索和解决,如算法中涉及的参数设置的理论依据、算法的收敛性分析以及与其他群智能优化算法的有机融合等,这些都是进一步要做的工作。

参考文献:

- [1] 李晓磊, 邵之江, 钱积新. 一种基于动物自治体的寻优模式: 鱼群算法[J]. 系统工程理论与实践, 2002, 22(11): 32-38.
- [2] KARABOGA D. An idea based on honey bee swarm for numerical optimization, Technical Report-TR06[R]. Kayseri: Erciyes University, 2005.
- [3] KARABOGA D, AKAY B. A survey: algorithms simulating bee swarm

intelligence[J]. Artificial Intelligence Review, 2009, 31(1-4): 61-85.

- [4] KRISHNANAND K N, GHOSE D. Detection of multiple source locations using a glowworm metaphor with applications to collective robotics[C]//Proc of IEEE Swarm Intelligence Symposium. Piscataway: IEEE Computer Society, 2005: 84-91.
- [5] YANG X S. nATURE Inspired metaheuristic algorithms[M]. Frome, UK: Luniver Press, 2008: 83-96.
- [6] YANG Xin-she. Nature-inspired metaheuristic algorithms[M]. 2nd ed. Frome, UK: Luniver Press, 2010: 97-104.
- [7] YANG Xin-she. A new metaheuristic bat-inspired algorithm[M]//GONZÁLEZ J R, PELTA D A. Nature Inspired Cooperative Strategies for Optimization. Berlin: Springer, 2010: 65-74.
- [8] SHI Yu-hui, EBERHART R. A modified particle swarm optimizer[C]//Proc of IEEE International Conference on Evolutionary Computation. Anchorage: IEEE Press, 1998: 69-73.
- [9] 王凌, 刘波. 微粒群优化与调度算法[M]. 北京: 清华大学出版社, 2008: 125-126.
- [10] TASGETREN M F, SEVKLI M, LIANG Y C, et al. Particle swarm optimization algorithm for permutation flowshop sequencing problem[C]//Lecture Notes in Computer Science, vol 3172. 2004: 382-389.