带交尾行为的混沌人工萤火虫优化算法

黄 凯 周永权
(广西民族大学数学与计算机科学学院 南宁 530006)

摘 要 针对基本萤火虫优化(GSO)算法在求解全局优化问题存在易陷入局部极小值、收敛速度慢和求解精度不高等缺陷,首先对基本萤火虫优化算法采用混沌搜索技术进行初始化,使算法获得质量较高且分布较均匀的初始解,在此基础上再引入交尾行为,提出了一种带交尾行为的混沌萤火虫优化算法(MCGSO)。该算法在一定程度上防止了基本GSO算法易陷入局部最优,且能够获得精度更高的解甚至可达到理论最优解。最后,通过对8个标准测试函数进行测试,测试结果表明,带交尾行为的混沌萤火虫优化算法比基本萤火虫优化算法有更高的收敛速度和求解精度。

关键词 全局优化,GSO,交尾行为,MCGSO,混沌搜索
中图法分类号 TP183 **文献标识码** A

Chaotic Artificial Glowworm Swarm Optimization Algorithm with Mating Behavior

HUANG Kai ZHOU Yong-quan
(College of Mathematics and Computer Science, Guangxi University for Nationalities, Nanning 530006, China)

Abstract According to basic glowworm optimization (GSO) algorithm in solving global optimization problems easily into the local minimum value, slow convergence speed and higher precision, first of basic glowworm defects by chaotic search technology optimization algorithm initialized, make the algorithm can achieve high quality and are uniformly distributed, the initial solution again on this foundation mating behavior, introduced proposed mating behavior with the chaotic fireflies optimization algorithm (MCGSO). The algorithm to a certain extent prevent basic GSO algorithm easily trapped into local optimal, and can obtain higher accuracy can reach even the solution theory optimal solutions. Finally, based on 8 standard test functions, the test results show that mating behavior with the chaotic glowworm optimization algorithm than the basic glowworm optimization algorithm has higher convergence speed and precision.

Keywords Global optimization problem, Glowworm swarm optimization, Mating behavior, MCGSO, Chaotic search

1 引言

自然科学和工程技术等领域许多问题的求解都可归结为全局优化问题。关于全局优化问题,迄今为止,研究者已提出了很多种优化算法,包括传统的基于梯度的算法和各种启发式算法。传统算法如DFP变尺度算法、黄金分割法、共轭方向法、Powell方向加速法和区间方法等[1-3],这些算法的优化结果对初值选取的依赖性高,并且对函数的性态要求较高。但许多工程实际问题的目标函数性态差,一般很难用基于梯度的传统算法实现求解。近年来,随着计算智能技术的发展,相继提出各种新的仿生群智能算法,如粒子群算法(PSO)、模拟退火算法(SA)、遗传算法(GA)等群智能优化算法。然而由于各种函数的形态与复杂性不同,因此在优化问题上每种算法都有各自的优点与不足。

人工萤火虫算法(Glowworm Swarm Optimization, GSO)是由K. N. Krishnanad和D. Ghose于2005年模拟自然界萤火虫求偶或觅食行为而提出的一种新的群智能仿生算法[4,5],该算法近年来在计算智能领域引起了人们极大的关注,并逐渐成为计算智能领域一个新的研究热点。随着研究的不断深入,该算法已经成功应用于传感器的噪声测试[6]和模拟机器人[7]等。该算法的优势在于捕捉极值域速度快、捕捉效率高、具有较强的通用性等优点。但该算法也存在着对初始解分布的依赖性高、后期收敛速度慢、捕捉到的解精度不高等问题。本文针对基本萤火虫算法的不足,提出了一种带交尾行为的萤火虫优化算法(Chaos Glowworm Swarm Optimization With Mating Behavior, MCGSO)。仿真实验结果表明,改进后的萤火虫算法得到了性能较高、分布较均匀的初始解,有效避免了算法过早陷入局部最优值,交尾行为的引入使算法能捕捉到的解具有更高的精度和收敛速度。

2 基本萤火虫算法(GSO)描述

在基本萤火虫算法中,首先随机地在目标函数的解空间当中初始化一群萤火虫,每一只荧火虫都携带有初始值相同的萤光素,萤火虫的萤光亮度是与其萤光素值成正比例关系的,而萤光素值又与其运动过程中所处位置的适应度值紧紧联系在一起,其对应的萤光素值越高,对其邻域范围内的萤火虫的吸引力就越强。邻域范围在GSO中称之为决策域范围,其大小由其邻域半径(r_d)决定。r_d 的大小在初始最大值 r_s 与0之间变动,r_s 称为可视范围(radial sensor range)。在萤火虫的运动当中,每一只萤火虫以一定的概率向其邻域范围

到稿日期:2011-04-16 返修日期:2011-07-15 本文受广西自然科学基金(0991086)资助。
黄 凯(1985—),男,硕士生,主要研究方向为智能计算;周永权(1962—),男,博士,教授,主要研究方向为神经网络、计算智能及其应用。

内的邻居萤火虫前进。萤火虫 j 要成为萤火虫 i 的邻居萤火虫，必须满足 j 在 i 的邻域决策范围之内并且 j 的萤光素值要高于 i。通过萤火虫群的不断运动，较多的萤火虫最后会聚集在适应度值较高的萤火虫周围。

在 GSO 中，每一次迭代都由两个阶段组成：第一阶段是萤光素更新阶段；第二阶段是萤火虫的运动阶段。

萤光素更新阶段：在此阶段中，每一只萤火虫都按式(1)来更新萤光素：

$$l_i(t)=(1-\rho)l_i(t-1)+\gamma J(x_i(t)) \tag{1}$$

式中，$l_i(t)$为第 t 代第 i 个萤火虫的萤光素值，$\rho\in(0,1)$为控制萤光素值的参数，γ 为评价函数值的参数，$J(x_i(t))$为函数适应度值。

萤火虫运动阶段：在此阶段当中，萤火虫 i 以一定的概率选择邻域范围内的萤火虫 j 并朝其运动，概率选择公式如式(2)所示。萤火虫 i 下一时刻的位置由式(3)确定，在运动阶段最后进行决策域范围的更新，用式(4)进行。

路径概率选择公式：

$$p_{ij}(t)=\frac{l_j(t)-l_i(t)}{\sum\limits_{k\in N_i(t)} l_k(t)-l_i(t)} \tag{2}$$

位置更新公式：

$$x_i(t+1)=x_i(t)+s*\left(\frac{x_j(t)-x_i(t)}{\|x_j(t)-x_i(t)\|}\right) \tag{3}$$

式中，$x_i(t)\in R^m$ 表示萤火虫 i 在 m 维实数空间的位置，$\|\cdot\|$ 表示标准欧基里德距离运算符，$s(>0)$表示移动步长。

决策域范围更新公式：

$$r_d^i(t+1)=\min\{r_s,\max\{0,r_d^i(t)+\beta(n_t-|N_i(t)|)\}\} \tag{4}$$

式中，β是一个比例常数，n_t 是控制邻域范围内邻居萤火虫个数的参数，$|N_i(t)|$是萤火虫 i 邻域内邻居萤火虫的个数。

3 带交尾行为的混沌萤火虫算法(MCGSO)

3.1 混沌搜索策略

混沌[8,9]是自然界广泛存在的一种非线性现象，存在于绝大多数非线性系统中。混沌现象是不含外加随机因素的完全确定性系统所表现出介于规则和随机之间的一种随机行为。混沌现象具有独特的性质：(1) 随机性；(2) 遍历性：它能够在一定范围内按自身规律不重复地遍历所有状态；(3) 规律性。利用混沌变量进行优化搜索无疑具有很大的优越性。正是混沌的遍历性使其成为搜索过程中避免陷入局部最优，提高全局寻优能力的一种机制。

混沌搜索经常用 Logistic 函数：

$$z_{i+1,d}=\mu z_{i,d}(1-z_{i,d}),i=0,1,\cdots,\mu\in(2,4] \tag{5}$$

式中，μ 为控制参数，当 $\mu=4,0\leqslant z_0\leqslant 1$，Logistic 完全处于混沌状态。本文中 μ 取值 4，取式(5)中的 Logistic 映射为混沌信号发生器。

3.2 MCGSO 算法的交尾行为描述

自然界中的萤火虫交尾时，雌萤火虫以萤光素的形式发出求偶信息之后，在其周围会陆续聚集越来越多的雄萤火虫。这些雄萤火虫会在雌萤火虫周围试探性地向雌萤火虫“示好”，最后雌萤火虫会选择那些发光模式和自己相同并且发光持续时间长的雄萤火虫来进行交配。为了模拟萤火虫的这种交尾行为，我们假设在每一代萤火虫中，萤光素值最大的萤火虫作为此代当中的雌萤火虫。假设在每一代交配阶段中，每次同时有 *male_num* 只雄萤火虫进行 *tyrnumber* 次的试探，则算法在每一次试探中都在雌萤火虫的可视范围之内随机生成 *male_num* 只雄萤火虫。然后使雌萤火虫移到函数适应度值最高的雄萤火虫的位置，表示雄萤火虫和雌萤火虫交尾成功。但是交尾成功并不表示交配成功，并不表示此雄萤火虫的基因能成功传播，因为雄萤火虫之间还存在精子竞争。根据式(6)减小雄萤火虫的择偶半径，进行下一次雄萤火虫的试探，直至达到设定的探试次数为止。最后雌萤火虫的位置是所有进行试探的雄萤火虫的历史最优位置，表示此萤火虫和雌萤火虫进行了交尾行为，并且精子竞争成功，基因得到顺利的传播。在算法中，为了模拟雌萤火虫周围雄萤火虫数目的增多，假设此雌萤火虫在选择配偶时的可视半径为 *mate_rs*，其初值为 *rs*，根据式(6)不断地减小，从而模拟周围雄萤火虫密度增加。择偶可视半径更新公式如下：

$$mate\_rs=(1-constrap)mate\_rs \tag{6}$$

式中，*constrap* 表示每一次雌萤火虫择偶范围的收宿率。

3.3 MCGSO 算法的实施步骤

MCGSO 具体实施步骤如下：

Step 1　初始化 $\rho,\gamma,\beta,s,l_0,m,N,D$, *trynumber*, *male_num*, *constrap* 等参数，并初始化最大的迭代次数 $T_{\max}$。

Step 2　混沌初始化。随机产生一个 D 维的初始向量 $z_1=[z_{1,1},z_{1,2},z_{1,d},\cdots,z_{1,D}]$，$z_{1,d}\in[0,1]$，且各分量值之间有微小的差别，用向量 z_1 作为混沌初始迭代向量。根据式(5)，$z_{i+1,d}=\mu z_{i,d}(1-z_{i,d})$，$(d=1,2,\cdots,D;i=1,2,\cdots,N-1)$ 得到 N 个 $z_1,z_2,\cdots,z_N$。利用 $x_{id}=a_d+(b_d-a_d)z_{id}$，$(d=1,2,\cdots,D;i=1,2,\cdots,N)$把 z_i 的各分量载波到优化变量的取值范围。计算 $x_i,i=1,2,\cdots,N$ 目标函数值，从 N 个初始个体中选择前 m 个较好的个体作为初始解。

Step 3　对所有萤火虫按式(1)更新荧光素值。

Step 4　进入运动阶段后，求每只萤火虫的邻域邻居萤火虫集合。

Step 5　用轮盘赌方法选择移动方向的萤火虫 $j(j\in N_i(t))$，并用式(3)进行位置的更新。

Step 6　用式(4)更新决策域半径。

Step 7　对萤光素值最大的萤火虫进行交尾行为。

Step 8　完成一次迭代，判断是否满足结束条件，记录结果，退出迭代。否则转 Step 3，进入下一次迭代。

3.4 MCGSO 算法的伪代码

```
MCGSO ALGORITHM:
    Set number of dimensions=D
    Set number of glowworms=m
    Let s be the step size
    Initialize all other parameters
Let X_i(t) be the location of glowworm i at time t
    x=ChaosInitialize(N,m);
    For i=1 to m do l_i(0)=l_0
    r_d^i(0)=r_0
    Set maximum iteration number =T_max;
    Set t=1;
while (t≤T_max) do
{
    for each glowworm i do: % Luciferin-update phase
    l_i(t)=(1-ρ)l_i(t-1)+γJ(x_i(t));    %See(1)
```

for each glowworm i do; % Movement-phase

{

$N_i(t)=\{j:d_{ij}(t)<r_d^i(t);l_i(t)<l_j(t)\}$;

for each glowworm $j\in N_i(t)$ do:

$p_{ij}(t)=\frac{l_j(t)-l_i(t)}{\sum_{k\in N_i(t)}l_k(t)-l_i(t)}$; %See(2)

$j=select\_glowworm(\vec{p})$;

$x_i(t+1)=x_i(t)+\frac{s(x_j(t)-x_i(t))}{||x_j(t)-x_i(t)||}$; %See(3)

$r_d^i(t+1)=\min\{rs,\max\{0,r_d^i(t)+\beta(n_t-|N_i(t)|)\}$; %See(4)

}

$index=\max(l)$;

$Brightest=x_{index}$;

$Brightest=mate(Brightest,trynumber,male\_num)$;

$x_{index}=Brightest$;

$t=t+1$;

}

3.5 MCGSO 算法的流程

MCGSO 算法的流程图如图 1 所示。

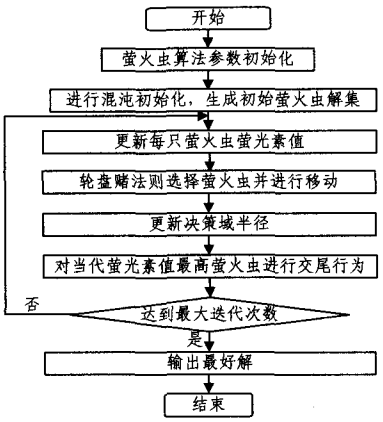

图 1 MCGSO 算法的流程图

4 实验分析

4.1 实验测试函数

为了验证 MCGSO 收敛速度比 GSO 快、求解精度比 GSO 高、算法的稳定性和健壮性比 GSO 好，采用了 8 个标准测试函数对算法进行了对比测试。8 个测试函数如下：

$$F_1(x)=-13+x_1+((5-x_2)\times x_2-2)\times x_2)^2-29+x_1+((x_2+1)\times x_2-14)\times x_2)^2$$

$$F_2(x)=(x_1^2+x_2^2)^{\frac{1}{4}}(\sin^2(50(x_1^2+x_2^2)^{\frac{1}{10}}+1\cdot 0)$$

$$F_3(x)=-\cos x_1\cos x_2 e^{-(x_1-\pi)^2-(x^2-\pi)^2}$$

$$F_4(x)=\sum_i^3 x_i^2$$

$$F_5(x)=100(x_1^2-x_2^2)^2+(1-x_1)^2$$

$$F_6(x)=g(x)h(x)$$

$$g(x)=1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)$$

$$h(x)=30+(2x_1-3x_2)^2(18-32x_1+12x^2+48x_2-36x_1x_2+27x_2^2)$$

$$F_7(x)=4+4\cdot 5x_1-4x_2+x_1^2+2x_2^2-2x_1x_2+x_1^4-2x_1^2x_2$$

$$F_8(x)=4x_1^2-2x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$$

标准测试函数如表 1 所列。

表 1 标准测试函数

函数	搜索范围	最优值
F_1	[−10,10]	0
F_2	[−100,100]	0
F_3	[−100,100]	−1
F_4	[−5.12,5.12]	0
F_5	[−2.048,2.048]	0
F_6	[−2,2]	3
F_7	[−100,100]	−0.5134
F_8	[−5,5]	−1.031628

4.2 实验参数设置

实验参数设置如表 2 所列。

表 2 SVGSO 算法参数取值

ρ	γ	β	n_t	l_0	male_num	trynumber	constrap	r_s	s
0.4	0.6	0.08	5	5	2	10	0.01	3	0.03

在 MCGSO 和 GSO 算法中，对 F_1，F_2，F_4-F_8 的测试萤火虫的个数都取 50。而在对 F_3 的测试中，MCGSO 取 50，而 GSO 取 400。

4.3 测试平台

本实验用 MATLAB2008a 编写的仿真程序，运行在 Windows XP 操作系统，Intel® Core™ 2 Duo CPU E4500 2.20GHz，1G 内存的 PC 机。

4.4 实验仿真结果分析

对这两种算法的检测用两种方式来进行：第一种方式为在给定的精度 $\varepsilon=10^{-5}$ 下，如果满足 $|fun_{bes}-fun^*|<\varepsilon$，则认为此解收敛，否则不收敛。其中，$fun_{best}$ 为求得的最优解，fun^* 为理论最优解。分别测试两种算法达到设定精度所需的进化代数。第二种方式为分别测试两种算法对于每一个测试函数所求得的最优值、最差值、均值，并通过收敛曲线图直观给出两种算法收敛速度和求解精度的对比。

表 3 设定精度下两种算法 20 次实验所需代数对比

函数	算法	20 次实验收敛所需的最少迭代次数	20 次实验收敛所需的最多迭代次数	20 次实验收敛所需的平均迭代次数	获取函数理论最优值的次数	收敛率
F_1	GSO	227	382	3.2525 e+002	0	8/20
	MCGSO	45	245	86.3000	14	20/20
F_2	GSO	1	371	1.7050 e+002	0	6/20
	MCGSO	1	26	9.9000	20	20/20
F_3	GSO	—	—	—	0	0/20
	MCGSO	33	56	46.7368	19	19/20
F_4	GSO	177	376	2.7656 e+002	0	16/20
	MCGSO	34	63	55.4500	0	20/20
F_5	GSO	61	123	89.9000	0	20/20
	MCGSO	36	103	71.7000	2	20/20
F_6	GSO	91	335	193	0	6/20
	MCGSO	66	319	1.8936 e+002	0	14/20
F_7	GSO	—	—	—	0	0/20
	MCGSO	34	55	46.08	11	11/20
F_8	GSO	57	253	1.7867 e+002	0	18/20
	MCGSO	45	134	72.7000	0	20/20

实验仿真选用了 8 个标准测试函数进行了 20 次独立实验并以以上两种方式来测试本文算法的有效性。通过与标准的 GSO 算法进行对比，从表 3 可以看出，MCGSO 算法所需的最少迭代次数、最多迭代次数和平均迭代次数都要比 GSO 少，而收敛次数 MCGSO 比 GSO 要多。从表 4 可以看出，对于其中 F_1、F_2、F_3、F_5 和 F_7，MCGSO 都取得了理论最优解。对于比较复杂的 F_3 和 F_7 函数，GSO 算法求得的解与理论最

优解偏离得较远，而 MCGSO 算法却获得了理论最优解。对于其它的函数，从表 3 也可以看到，MCGSO 的收敛所需的进化代数较少，求解精度比 GSO 有显著提高，因此，MCGSO 比 GSO 捕获精确解的效率和准确性要高得多。

表 4　标准测试函数实验结果对比

函数	算法	最优值	最差值	平均值
F_1	MCGSO	0	1.620676045021684e-006	8.103416721924578e-008
	GSO	3.784415801294448e-007	18.653620462685740	2.659715343839053
F_2	MCGSO	0	0	0
	GSO	1.269986789060944e-007	0.171100827857829	0.016891447040379
F_3	MCGSO	−1	0	−0.950000000000000
	GSO	−0.541897374065207	−4.974928351669068e-194	−0.950000000000000
F_4	MCGSO	2.003869883032956e-036	4.515959267628626e-034	3.932272661044802e-035
	GSO	2.192331910404988e-007	0.081876788454084	0.004097924048164
F_5	MCGSO	0	7.558211495916246e-007	6.577095821146598e-008
	GSO	2.195482410410626e-009	1.874248793623958e-007	5.283213757789870e-008
F_6	MCGSO	2.999999999999922	3.000028953713166	3.000006738856884
	GSO	3.000001214466028	3.000088326426064	3.000028459885972
F_7	MCGSO	−0.513409257283794	1.008421422686688	0.087319564205257
	GSO	2.094931129686337	6.577093964630658e+003	1.684129705725960e+003
F_8	MCGSO	−1.031621943868863	−1.031621889810352	−1.031621941165938
	GSO	-1.031621941057579	0.520833524743552	−0.953998450377353

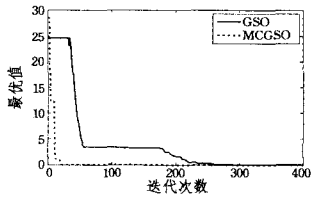

图 2　F_1 函数收敛曲线对比

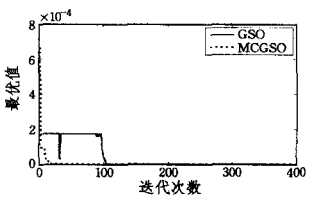

图 3　F_2 函数收敛曲线对比

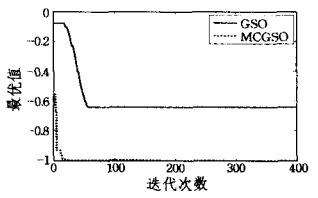

图 4　F_3 函数收敛曲线对比

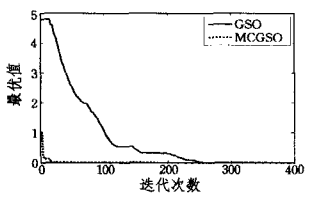

图 5　F_4 函数收敛曲线对比

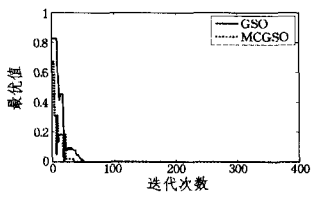

图 6　F_5 函数收敛曲线对比

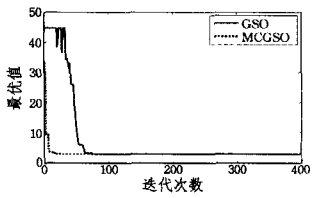

图 7　F_6 函数收敛曲线对比

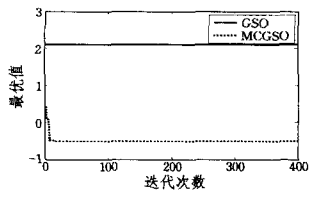

图 8　F_7 函数收敛曲线对比

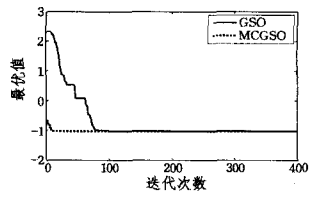

图 9　F_8 函数收敛曲线对比

从上面各收敛曲线图可以看出(见图 2－图 9)，MCGSO 算法比 GSO 算法在函数全局优化方面收敛速度更快、求解精度更高。在 F_3 函数的测试中，MCGSO 只用了 50 只萤火虫，而 GSO 用了 400 只萤火虫，而其效果仍没有 MCGSO 好，并且还过早地陷入了局部最优值。从图 8 可以看到，GSO 算法陷入了局部最优，而 MCGSO 收敛到了全局最优值。从其它曲线图可直观地看出，MCGSO 算法稳定性、健壮性也都要比 GSO 好，而且在萤火虫个数相对较少的情况能有效地防止算法陷入局部最优。

结束语　提出了一种带交尾行为的混沌人工萤火虫算法，通过在算法中加入混沌搜索策略，提高了初始解的质量，起到了有效防止算法陷入局部最优值，提高了捕获到全局最优值的可能性。又在 GSO 算法中引入交配行为，进一步提高了 MCGSO 算法的收敛速度和求解精度。这两种策略的加入，同时增强了算法的稳定性与健壮性。因此，改进后的 MCGSO 算法较 GSO 算法是一种更有效的算法，但是改进后的算法多数情况下是用在全局优化等问题方面，因此，如何把改进后的算法用于局部优化方面，是有待进一步研究的工作。

参 考 文 献

[1] 袁亚湘，孙文瑜. 最优化理论与方法[M]. 北京：科学出版社，2003

[2] 陈开周. 最优化计算方法[M]. 西安：西安电子科技大学出版社，1990

[3] Csendest. Numerical experiences with a new generalized subinterval selection criterion for interval global optimization [J]. Reliable Computing，2003，9(2)：109-125

[4] Krishnanand K N D，Ghose D. Glowworm swarm optimization：new method for optimizing multi-modal functions[J]. Computational Intelligence Studies，2009，1(1)：93-119

[5] Krishnanand K N. Glowworm swarm optimization：a multimodal function optimization paradigm with applications to multiple signal source localization tasks[D]. Indian：Department of Aerospace Engineering，Indian Institute of Science，2007

[6] Krishnanand K N，Ghose D. A glowworm swarm optimization based multi-robot system for signal source localization [M]. Design and Control of Intelligent Robotic Systems，2009：53-74

[7] Krishnanand K N，Ghose D. Chasing multiple mobile signal sources：a glowworm swarm optimization approach [C]// Third Indian International Conference on Artificial Intelligence (IICAI 07). Indian，2007

[8] Liu B，Wan L，Jin Y H，et al. Improved particle swarm optimization combined with chaos [J]. Chaos，Solitons and Fractals，2005，25：1261-1271

[9] 高尚，杨静宇. 群智能算法及其应用[M]. 北京：中国水利水电出版社，2006：63-82

[10] Krishnanand K N，Ghose D. Theoretical foundations for rendezvous of glowworm-inspired agent swarms at multiple locations [J]. Robotics and Autonomous Systems，2008，56(7)：549-569