浙江理工大学学报(自然科学版),第 37 卷,第 6 期,2017 年 11 月
Journal of Zhejiang Sci-Tech University (Natural Sciences)
Vol. 37, No. 6, Nov. 2017

DOI:10.3969/j.issn.1673-3851.2017.11.015

多模态函数聚类后再创种群的并行搜索佳点集萤火虫算法

方　贤[1],铁治欣[1],李敬明[2],高　雄[1]

(1.浙江理工大学信息学院,杭州 310018;2.合肥工业大学管理学院,合肥 230009)

摘　要:萤火虫算法在求解多模态函数时,随着峰值个数的增加,往往需要更大的种群规模才能得到较为理想的结果,而且初始种群是否均匀分布对结果也有很大影响。针对萤火虫算法的这些不足,提出了一种多模态函数的聚类后再创种群的并行搜索佳点集萤火虫算法。该算法首先以数论佳点集的思想将萤火虫均匀分布于搜索空间中,在粗糙搜索完成后,通过密度聚类算法进行捕峰操作,重新构造等同于峰值点数的各个平行空间;然后在各空间中继续加入少量佳点集生成的萤火虫并行精细搜索,最终可获得各个平行空间的局部最优解以及整个空间的全局最优解。与其他算法在 12 个典型多模态函数中的测试结果进行对比,该算法总体上缩小了种群规模,加快了收敛速度,搜索精度更高,时间成本更低,稳定性能更好。

关键词:萤火虫算法;多模态函数;佳点集;密度聚类算法;并行搜索

中图分类号:TP391.9　　**文献标志码**:A　　**文章编号**:1673-3851 (2017) 06-0843-08

0　引　言

在工程技术、科学计算、经济管理等领域中,绝大多数实际问题可以通过构建模型归结为函数优化问题。其中,一些函数由于自身维数高、峰值点多、震荡严重等因素造成性态差。采用传统的算法,如 DFP 变尺度算法、Powell 方向加速法等,往往难以或无法找到全局最优解。近年来随着智能计算学科的发展,一些仿生算法应运而生,如粒子群算法(particle swarm optimization, PSO)、遗传算法(genetic algorithm, GA)、蚁群算法(ant clony optimization, ACO)、蜂群算法(artificial bee colony algorithm, ABC)、鱼群算法(artificial fish swarm algorithm, AFSA)等。这些基本群智能算法可以较好地满足全局搜索需求,但在算法性能上仍有很大程度的提升空间。

多模态函数优化问题(multi-modal function optimization problems),又称多峰值函数优化问题,在函数优化问题中最为常见。它指的是具有多个峰值点的函数,因存在较多的峰值点,很容易使得算法寻优陷入局部最优而无法找出全局最优。将一些基本算法的改进措施应用于多模态函数的全局寻优上可以取得更好的效果,例如:Kennedy[1]在基本 PSO 算法基础上提出一种基于聚类技术改进的 PSO 算法,使用 K-means 算法将主群分割为 K 个子群,通过各子群中粒子的全局最优位置与历史最优位置的质心交换来更新主群的相关参数。Shelokar 等[2]将 ACO 算法与 PSO 算法结合,采用简单的信息素将蚁群与粒子群相关联,当粒子更新时,与之关联的蚂蚁也会更新。Agrawal 等[3]将 Fletcher-Reeves 共轭梯度的思想用于搜索 PSO 算法中的粒子访问区域,可以使粒子移动到更具寻优潜力的区域。Fan 等[4]提出一种最小消除逃逸函数方法,该方法主要由最小消除函数和最小逃逸函数两个子函数实现功

收稿日期:2017－06－14　　网络出版日期:2017－10－10

基金项目:国家自然科学基金项目(61170110);浙江省公益技术应用研究项目(2014C31G2060072);安徽省教育厅自然科学研究重点项目(KJ2016A308)

作者简介:方　贤(1994－),男,安徽滁州人,硕士研究生,主要从事智能计算、生物信息学方面的研究。

通信作者:铁治欣,E-mail:tiezx@zstu.edu.cn

能,前者用于减少局部最优解的个数,后者在此基础上将当前最小解作为唯一的全局最大值。Zhou等[5]对基本ABC算法进行了多方面改进,首先通过动态调整蜂群的规模较好地适应不同的目标函数,其次加入了平衡搜索机制用于加快算法收敛速度,此外通过半径估算及最佳精英策略的嵌入分别增强了不规则分布的蜜蜂搜索最优解能力,消除无关的局部最优解能力。

很多实际问题不仅要求寻找全局最优解,同时还需要找到众多有意义的局部最优解,以便工程师进行多方面考量与决策。因此,对于多模态函数优化问题,在搜索到其全局最优解的同时获得局部最优解既具有挑战难度,更具有实际价值。萤火虫群优化(glowworm swarm optimization,GSO)算法具有操作简单易实现、算法流程清晰、全局与局部性能协调、参数设置较少、鲁棒性强等优点,尤其在可视化方面表现优异,能直观地显现出萤火虫在多模态函数极值点附近的聚簇状况,并快速获取局部最优解以及整个空间的全局最优解。GSO以其卓越的特性而广泛运用于多模态函数优化中,如:Zhang等[6]提出了自适应步长的V-GSO算法和自探索行为的E-GSO算法,并列举3个多模态函数验证GSO算法可以通过动态线性或非线性递减步长两种途径提升性能,此外为每只萤火虫设置阈值和适应度值,通过两者关系来判断是否存在邻居进而考虑是否采用螺旋搜索可改善GSO的自适应性。Zhou等[7]提出了一种混合的HGSO算法,该算法将AFSA算法中鱼群的觅食行为引入到GSO算法中,采用双种群的协同进化机制,同时加入模拟退火算法的局部搜索策略以克服早熟收敛缺陷。虽然以上各种改进的策略在收敛速度和计算精度上较传统GSO有较大的提高,但GSO算法在多模态函数优化中仍存在以下两方面问题未得到妥善地处理[8-9]:一方面,萤火虫的初始种群随机分布,算法稳定性能无法得到提升,不适合的分布还会导致算法收敛速度慢、搜索精度低;另一方面,随着多模态函数峰值点个数的增加,较少的萤火虫种群往往难以捕捉到所有的峰值,即便可以成功捕获所有峰值,也很难将搜索到的各个峰值点的误差降到合理的范围内。为克服以上不足,本文提出一种针对多模态函数的聚类后再创种群的并行搜索佳点集萤火虫算法(parallel search good-point set GSO,PGSGSO),并通过在12个基准测试函数上的仿真实验验证了所提算法的可行性与有效性。

1 基本GSO算法

基本GSO算法最初是由印度学者Krishnanand和Ghose[10-11]于2005年提出的一种新颖的群智能优化算法。此后,随着国内外广大学者们的研究深入[12-16],该算法逐渐展露出其独特的多模态寻优性能,展现出良好的研究与应用前景。通过观察自然界中一种常见的现象可以发现:夜晚萤火虫利用荧光亮度与外界交互,求偶或觅食。在感知范围内,每只萤火虫凭借自身的亮度吸引其它亮度更弱的萤火虫向它聚集,同时也受到亮度更强的萤火虫的吸引。GSO算法受这种自然机理的启发设计。它的数学描述为:若干只萤火虫个体 $i(i=1,2,\cdots,n)$ 被随机分布于目标函数 f 定义的范围空间 $\boldsymbol{D}$ 中,每只萤火虫代表了优化问题的一个潜在解,这些萤火虫各自拥有所在位置对应于目标函数的评价值,称为适应度值。愈亮的萤火虫其适应度值愈优,吸引其它萤火虫向自己移动的能力也愈强。同时,它们还拥有相同的感知半径与决策半径,处于感知范围内的亮度更弱的萤火虫,称为邻居。迭代过程中,决策半径与荧光素值动态更新。其中决策半径受邻居数目的影响,两者成反比例关系;荧光素值受挥发系数及增强系数的影响。迭代完成后,大多数萤火虫会聚集到相对更优的适应度值的位置,获得寻优结果。GSO算法主要涉及以下公式:

a) 荧光素更新公式为

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

式中:ρ 为荧光素值挥发系数,γ 为荧光素值增强系数,$l_i(t)$ 为第 t 次迭代时萤火虫 i 的荧光素值,$x_i(t)$ 为第 t 次迭代时萤火虫 i 的位置,$J(x_i(t))$ 为第 t 次迭代时的目标函数值。

b) 位置更新公式为

$$x_i(t+1)=x_i(t)+s\left(\frac{x_j(t)-x_i(t)}{\|x_j(t)-x_i(t)\|}\right) \tag{2}$$

式中:s 为萤火虫位移的步长。

c) 邻居集合公式为

$$N_i(t)=\{j:\|x_j(t)-x_i(t)\|<r_d^i(t);l_i(t)<l_j(t)\} \tag{3}$$

式中:$N_i(t)$ 为第 i 只萤火虫在第 t 次迭代时的邻居数目,它受到决策半径 r_d^i 的约束。

d) 路径概率选择公式为

$$p_{ij}(t)=\frac{l_j(t)-l_i(t)}{\sum\limits_{k\in N_i(t)}l_k(t)-l_i(t)} \tag{4}$$

式中:$p_{ij}(t)$ 为第 i 只萤火虫转向邻居中其他萤火虫

的转移概率。

e) 决策域更新公式为

$$r_d^i(t+1)=\min\{r_s,\max\{0,r_d^i(t)+\beta(n_t-|N_t(t)|)\}\} \tag{5}$$

式中:r_s 为邻域感知半径($0<r_d^i<r_s$),β 为感知半径变化系数,n_t 为邻居个数阈值。

2　聚类后再创种群的并行搜索佳点集萤火虫算法

2.1　佳点集理论

1978 年底,华罗庚等[17]详细阐述了佳点集思想及其证明过程。佳点集的有关描述如下:令 s 为一正整数,$\boldsymbol{G}_s$ 表示 s 维欧氏空间中的单位立方体,即《x》$=(x_1,x_2,\cdots,x_s)\in\boldsymbol{G}_s$,其中 $0\leqslant x_i\leqslant 1,i=1,2,\cdots,s$。再令 n 为正整数,表示空间中生成的 n 个点,点集合可表示为 $\boldsymbol{P}_n(i)=\{(x_1^{(n)}(i),x_2^{(n)}(i),\cdots,x_s^{(n)}(i)),1\leqslant i\leqslant n\}$。对任意的《$r$》$=(r_1,r_2,\cdots,r_s)\in\boldsymbol{G}_s$,令 $N_n(《r》)=N_n(r_1,r_2,\cdots,r_s)$ 表示点集合 $\boldsymbol{P}_n(i)$ 中满足不等式 $0\leqslant x_k^{(n)}(i)<r_sk=1,2,\cdots,s$ 的点个数,如果 $\Phi(n)=\mathrm{Sup}\left|\dfrac{N_n(r)}{n}-|《r》|\right|$,则称点集合 $\boldsymbol{P}_n(i)$ 具有偏差 $\Phi(n)$。若此时形如 $\boldsymbol{P}_n(i)=\{(r_1*i,r_2*i,\cdots,r_i*i),i=1,2,\cdots,n\}$ 的点集合偏差满足 $\Phi(n)=C(《r》,\varepsilon)n^{(-1+\varepsilon)}$,其中 $C(《r》,\varepsilon)$ 是只与《r》,ε(ε 是任意小的正数)有关的常数,则称 $\boldsymbol{P}_n(i)$ 为佳点集,《r》为佳点集合中的佳点。

张钹等[18]更进一步地研究得出以下结论:

a) 在近似积分的运用中,佳点集所产生的误差的阶只依赖于样本个数,并不随着样本空间维数的递增而变大。这为高维空间的研究提供了一个潜在的理论基础。

b) 对于任意未知分布的对象,随机产生 n 个空间中的点所得出的偏差通常为 $\Phi(n)=O(n^{-1/2}(\log\log n)^{1/2})$,使用佳点集分布所得到的偏差仅为 $\Phi(n)=O(n^{-1+\varepsilon})$。相比较随机方法,佳点集的偏差降到了平方根级别。

2.2　密度聚类算法

聚类算法是数据挖掘领域重要的技术,其思路是按照事物内在属性将对象聚集成若干类,要求同一类间相似性尽可能高,而不同类间相似性尽可能低。它是一种无导师指导的学习方法,主要应用在模式识别、推荐系统、图像处理等方向。根据具体实现方法的不同可将聚类算法分为五种:划分法聚类、层次法聚类、基于模型的聚类、网格法聚类以及密度聚类。其中,密度聚类算法区别于另外几种方法很重要的一点,在于该方法是以数据在空间中分布的稠密程度为原则进行聚类,因此无需预先设置簇的数量,尤其适用于搜寻未知多模态函数的峰值数。

DBSCAN(density-based spatial clustering of applications with noise)算法是密度聚类算法中最为典型的一种,其基本思想是:从数据集中的任意对象出发,查找该对象在半径 Eps 条件下密度可达的所有对象,若所发现的对象不少于最小密度阈值 $MinPts$,则称该对象为核心对象,若所发现的对象少于最小密度阈值 $MinPts$,则称该对象为边界点,此时边界点会被暂时标记为噪声点。采用广度优先搜索循环执行,最终由一个核心对象与其密度可达的所有对象构成一个聚类。

2.3　改进的萤火虫算法

本文所提的 PGSGSO 算法从以下两个方面对基本 GSO 算法进行改进。首先,在基本 GSO 算法的初始种群构造方面,使用数论佳点集均匀化的思想设计萤火虫个体的均匀分布,使得尽可能少的萤火虫能全面地表征整个空间所有潜在解的分布。换言之,如果产生随机初始种群,就很难遍历解空间中的各种状况,只有将整个解空间中最能表征潜在解分布的若干个体作为初始解集合,才能更为有效地解决实际问题。因此,佳点集均匀化设计可以很好地解决这个问题。其次,在针对多模态函数捕峰操作方面,考虑到仅用少量的萤火虫难以准确地捕捉到复杂函数的各个峰值点,而使用大量的萤火虫无疑会增加算法的复杂度,加大时间开销。本文借助密度聚类函数在 GSO 算法迭代完成后,对散落在空间中的所有萤火虫进行聚类,然后分别将每个类中的萤火虫的坐标以各个坐标轴对应的最大最小值为边界,构建出等同于峰值点数的各个平行空间。通过在各个平行空间中继续加入少量的佳点集策略生成的萤火虫进行精细搜索,可以达到并行搜索的效果,以此权衡效率与成本之间的矛盾。

PGSGSO 算法的基本步骤如下:

a) 初始化基本 GSO 算法相关参数,包括:搜索空间维数 d,种群规模 n,荧光素值 l_0,荧光素值挥发系数 ρ,荧光素值增强系数 γ,感知半径 r_s,感知半径变化系数 β,邻居个数阈值 n_t,移动步长 s 等,设置迭代计数器初值为 $t_0=1$,终值为 $t_{\max}$。

b) 利用佳点集方法在搜索空间中设计萤火虫种群的均匀分布,空间中的坐标对应为萤火虫种群的初始位置,选取典型多模态函数作为评价适应度

值函数。

c) 按照式(1)更新所有萤火虫荧光素值。

d) 萤火虫个体按照式(2)进行位置更新。其过程先按照式(3)计算邻居集合,再按照式(4)计算移动概率,以轮盘赌法作为选择移动对象的方法。

e) 按照式(5)对萤火虫动态决策半径进行更新。

f) 判断是否达到最大迭代次数,若是,转向步骤 g,否则,令 $t=t+1$ 并转向步骤 c。

g) 使用密度聚类算法将搜索空间中萤火虫个体聚集在等同于函数峰值数的若干区域内,各区域空间萤火虫坐标值排序后选择最大最小坐标固定平行空间界限。继续投入少量佳点集生成的萤火虫于各平行空间并行搜索。

h) 重复执行步骤 c—e 直至算法达到此阶段最大迭代次数。

i) 输出结果,算法终止。

3 实验分析

3.1 测试函数

为了验证本文所提的 PGSGSO 算法在实际运用中的有效性,采用 Matlab R2013b 平台编写代码,实验环境为:Intel 3.10 GHz CPU、64 位 Windows XP 操作系统。选取了 12 个标准测试函数 f_1—f_{12} 对算法进行仿真分析,它们均是定义在二维空间中的多模态函数,其中:f_1—f_5 的极小峰值点个数小于等于 50,相对简单;f_6—f_{12} 均是极小峰值点个数远大于 50 的复杂多模态函数。f_1、f_9 选自文献[19],f_2 选自文献[3],f_3、f_7 选自文献[20],f_4 选自文献[21],f_5 选自文献[2],f_6、f_{10}、f_{11} 选自文献[22],f_8 选自文献[23],f_{12} 选自文献[4]。12 个测试函数如下:

$$\min f_1(x)=\sum_{i=1}^{2}(|x_i|-5)^2,$$

$$\begin{aligned}\min f_2(x)=&(1+(x_1+x_2+1)^2(19-14(x_1+x_2)+\\&3(x_1^2+x_2^2)+6x_1x_2))\times\\&(30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+\\&48x_2+36x_1x_2+27x_2^2)),\end{aligned}$$

$$\min f_3(x)=4x_1^2-2.1x_1^4+\frac{x_1^6}{3}+x_1x_2-4x_2^2+4x_2^4,$$

$$\min f_4(x)=x_1^2+x_2^2+25(\sin^2x_1+\sin^2x_2),$$

$$\min f_5(x)=x_1^2+x_2^2-\cos(18x_1)-\cos(18x_2),$$

$$\begin{aligned}\min f_6(x)=&-20\exp\left(-0.2\sqrt{\sum_{i=1}^{2}\frac{x_i^2}{n}}\right)-\\&\exp\left(\sum_{i}^{2}\cos\left(\frac{2\pi x_i}{n}\right)\right)+20+e,\end{aligned}$$

$$\min f_7(x)=\prod_{i=1}^{2}\sum_{j=1}^{5}j\times\cos[(i+1)x_i+j],$$

$$\min f_8(x)=\prod_{i=1}^{2}\sum_{j=1}^{5}j\times\cos[(i-1)x_i+j],$$

$$\min f_9(x)=0.5+\frac{\sin^2\sqrt{x_1^2+x_2^2}-0.5}{[1.0+0.0001(x_1^2+x_2^2)]},$$

$$\min f_{10}(x)=\sum_{i=1}^{2}[x_i^2-10\cos(2\pi x_i)+10],$$

$$\min f_{11}(x)=\frac{1}{4000}\sum_{i=1}^{2}x_i^2-\prod_{i=1}^{2}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1,$$

$$\min f_{12}(x)=-\sum_{i=1}^{2}[x_i\sin(\sqrt{|x_i|})].$$

表 1 实验使用的 benchmark 测试函数详细信息

测试函数	函数名称	维数 d	极小峰值个数	搜索范围	全局最小值
f_1	Becker and Lago	2	4	$[-10,10]$	0
f_2	Goldstein-Price	2	4	$[-2,2]$	3
f_3	Six hump camel back	2	6	$[-5,5]$	-1.0316
f_4	Eggcrate	2	14	$[-2\pi,2\pi]$	0
f_5	Rastrigin	2	50	$[-1,1]$	-2
f_6	Ackley's Function	2	>50	$[-32,32]$	0
f_7	Shubert	2	>50	$[-10,10]$	-186.7309
f_8	Hansen	2	>50	$[-10,10]$	-176.5417
f_9	Schaffers F6	2	>50	$[-10,10]$	0
f_{10}	Generalized Rastrigin's Function	2	>50	$[-5.12,5.12]$	0
f_{11}	Generalized Griewank's Function	2	>50	$[-600,600]$	0
f_{12}	Generalized Schwefel's Problem 2.26	2	>50	$[-500,500]$	$-418.9829\times d$

3.2　参数设置

实验参数设置如表 2 所示。对 f_1—f_5 函数,初始萤火虫的个数 n 设为 50;对 f_6—f_{12} 函数,初始萤火虫的个数 n 设为 200。

表 2　PGSGSO 算法实验参数

d	l_0	ρ	γ	γ_s	β	n_t	s	$t_{\max}$
2	5	0.4	0.6	3	0.08	5	0.03	50

3.3　实验结果讨论

以 $f_1(x)$为例说明 PGSGSO 算法的实行过程。首先在定义域空间[−10,10]中均匀生成 50 只萤火虫,如图 1(a)所示。在 20 次粗糙搜索完成后得到图 1(b)所示结果,从图 1(b)可以清晰地看出萤火虫聚集成 4 个簇,即象征着搜索空间中的 4 个峰值点。在事先对极小峰值个数未知的情况下,采用 DBSCAN 算法进行聚类分析得到如图 1(c)所示结果。图(c)中不同颜色的萤火虫积聚在一起代表不同的类,可以验证此时成功捕获出 4 个极小峰值,与实际峰值个数一致。然后对 4 个类划分平行空间,经过先前 20 次迭代萤火虫已经集中分布于各个峰值点附近,这时候需要在各个空间内精确搜索,为此设计在这些空间中增添 20 只萤火虫,如图 1(d)所示。

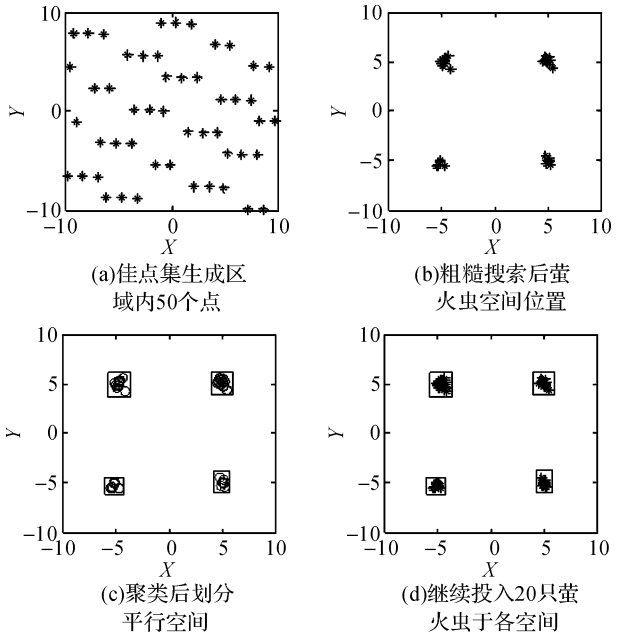

图 1　PGSGSO 对 $f_1(x)$的实行过程

PGSGSO 算法的过程实质上包括两个阶段:前期的全局粗糙搜索和后期的局部精细搜索。为合理地协调以上两个阶段,定义平衡因子 φ,表示前期的粗糙搜索在整个搜索过程所占据的权重($0<\varphi<100\%$)。考虑到 φ 的大小对于实验结果的重要性,在保持其他参数不变时,设置不同大小的 φ 独立进行 20 次实验,结果如图 2 所示。本文选用 $\varphi=40\%$ 作为最终实验参数。

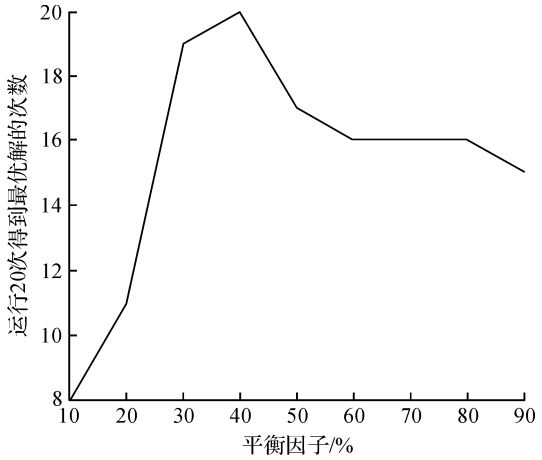

图 2　不同平衡因子与最优解次数曲线图

此外,在后期并行搜索中的过程中,由于划分空间导致的空间数目增加而空间范围缩小,步长与决策半径等变量需要进行调整。采用等比例原则处理上述问题:将空间的缩小以同等比例地应用于这些变量。另外,限制搜索范围,即各个空间之间互不干涉,保证并行搜索的可行性。

评价 PGSGSO 算法的改进工作应尽可能落实到全面而客观,表 3 从捕峰能力、对全局最优解所达到的精度、算法稳定性能、消耗时长等多方面综合考虑。用 PGSGSO 算法、HGSO 算法[7]、GSO-Powell 算法[19]对 12 个标准测试函数各进行 20 次实验,图 3 是各种算法对测试函数的其中一次寻优曲线。在捕峰能力的评价中,规定若平均捕峰个数与实际个数相差在 2%范围内,则称捕峰“成功”,成功率为 $(N_m/N_c)\times100\%$,其中 N_m 代表平均捕峰个数,N_c 代表实际峰值个数。

表 3　PGSGSO 与其他算法性能对比

测试函数	算法	平均最优值	方差	平均耗时/s	捕峰成功率/%
f_1	PGSGSO	6.29273×10^{-7}	7.3482×10^{-34}	12.39	100
	HGSO	3.96593×10^{-6}	3.5829×10^{-19}	32.54	100
	GSO-Powell	6.83576×10^{-7}	1.8909×10^{-15}	54.23	100

表 3 续

测试函数	算法	平均最优值	方差	平均耗时/s	捕峰成功率/%
	PGSGSO	3.000000258	3.4895×10^{-32}	13.52	100
f_2	HGSO	3.000005868	2.8627×10^{-7}	33.12	100
	GSO-Powell	3.000007843	5.1906×10^{-4}	53.42	100
	PGSGSO	−1.031628535	6.7426×10^{-23}	13.11	100
f_3	HGSO	−1.031628534	6.3556×10^{-7}	36.54	99.69
	GSO-Powell	−1.031628531	7.2425×10^{-13}	58.34	100
	PGSGSO	7.09023×10^{-6}	8.3885×10^{-12}	16.28	100
f_4	HGSO	3.93758×10^{-6}	5.2853×10^{-4}	38.35	99.72
	GSO-Powell	8.47588×10^{-6}	3.8938×10^{-3}	63.28	99.16
	PGSGSO	−1.999999039	1.4993×10^{-14}	22.19	100
f_5	HGSO	−1.999998347	8.5831×10^{-5}	56.34	98.88
	GSO-Powell	−1.999994896	2.4362×10^{-2}	63.52	98.92
	PGSGSO	3.25346×10^{-5}	7.4493×10^{-11}	38.57	100
f_6	HGSO	3.28755×10^{-5}	4.6673×10^{-2}	55.21	92.21
	GSO-Powell	4.35665×10^{-5}	9.5683×10^{-6}	77.35	98.79
	PGSGSO	−186.7309342	7.8482×10^{-12}	37.23	99.69
f_7	HGSO	−186.7308847	2.9275×10^{-6}	58.14	98.48
	GSO-Powell	−186.7305368	5.5362×10^{-4}	74.53	98.34
	PGSGSO	−176.5417942	3.8371×10^{-10}	39.11	99.74
f_8	HGSO	−176.5416837	3.5145×10^{-3}	46.41	98.32
	GSO-Powell	−176.5402596	6.2358×10^{-3}	79.27	98.47
	PGSGSO	8.28689×10^{-5}	2.3923×10^{-9}	39.35	99.43
f_9	HGSO	9.47835×10^{-4}	9.3433×10^{-5}	53.32	89.78
	GSO-Powell	1.22544×10^{-4}	8.5402×10^{-2}	78.42	98.83
	PGSGSO	5.92375×10^{-6}	6.4352×10^{-12}	39.23	99.55
f_{10}	HGSO	8.28658×10^{-5}	2.9057×10^{-5}	47.35	81.63
	GSO-Powell	7.68675×10^{-4}	5.3672×10^{-3}	78.43	79.64
	PGSGSO	8.22819×10^{-5}	5.3332×10^{-8}	42.52	97.28
f_{11}	HGSO	3.34376×10^{-4}	3.0289×10^{-3}	54.46	78.23
	GSO-Powell	7.05574×10^{-3}	7.7735×10^{-2}	72.23	86.28
	PGSGSO	−837.9658827	6.4425×10^{-8}	37.39	98.52
f_{12}	HGSO	−837.9638146	3.2975×10^{-2}	56.27	72.45
	GSO-Powell	−837.3975732	3.9728×10^{-3}	76.59	88.86

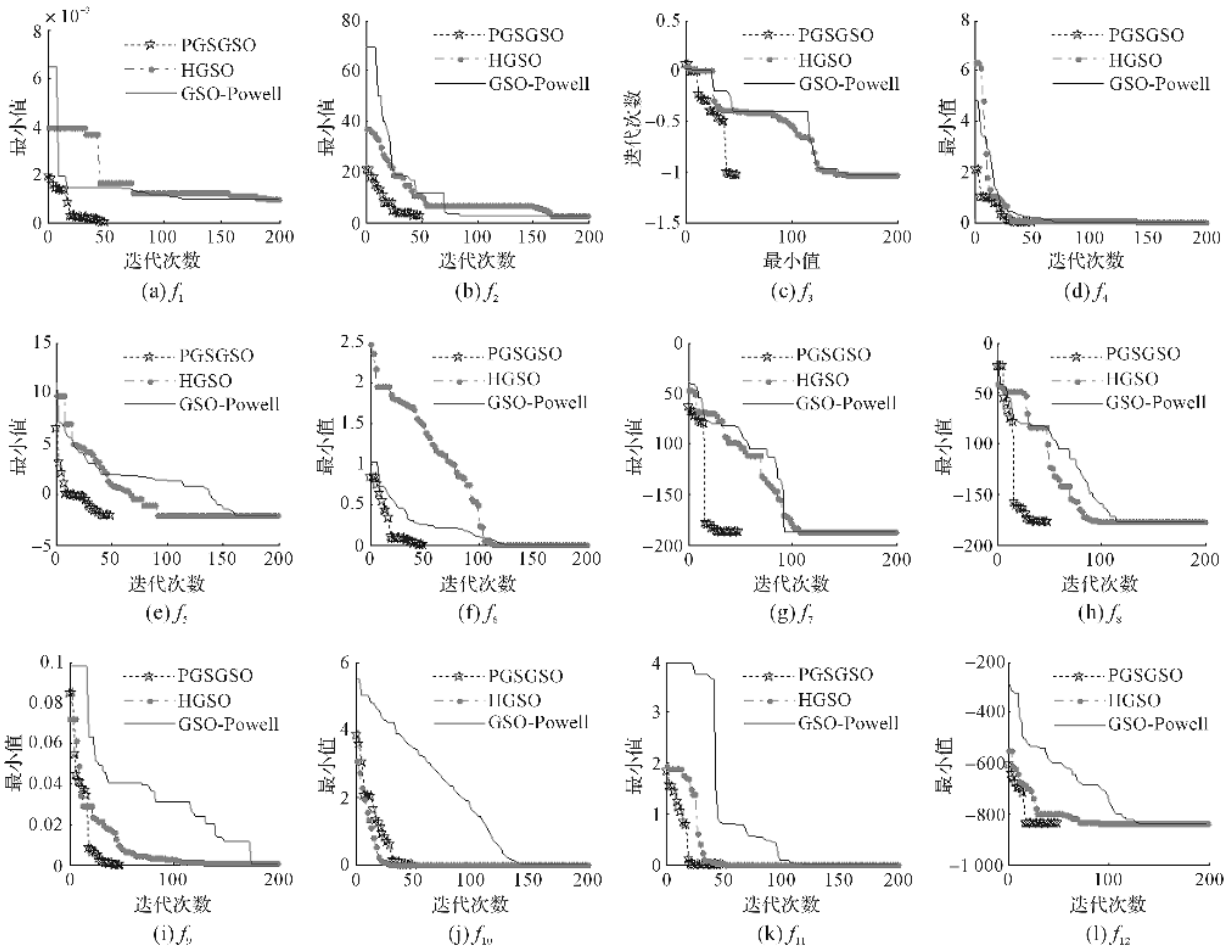

图 3　12 个函数的寻优曲线

由表 3 和图 3 可以看出,在捕峰能力方面,PGSGSO 除了 f_{11} 捕峰不"成功"外,其余均"成功"。而 HGSO 对 f_6、f_9—f_{12} 捕峰均不"成功",GSO-Powell 对 f_{10}—f_{12} 捕峰均不"成功"。在全局解的精度方面,总体来看,PGSGSO 稍优于 HGSO,GSO-Powell 次之。在算法稳定性能方面,观察方差可以发现 PGSGSO 明显更稳定于另外两种算法。在收敛速度方面,PGSGSO 在寻优过程中总是存在跳跃点,能迅速加快收敛速度,原因就在于采用了聚类后再创佳点集种群操作。从图 3 可以发现 PGSGSO 在对所有测试函数寻优时,第 1 次迭代时的值相对来说要好,原因是佳点集的均匀思想使得解分布更均匀,很容易遍布空间的各个角落。此外,本文 PGSGSO 算法采取了相对更少的种群规模与迭代次数,节省了时间开销。

4　结　语

GSO 算法在多模态函数优化中存在自身优势,但受到种群生成方式的随机性和往往需要大规模的萤火虫种群才能得到较为满意的寻优结果这两方面的制约,从而导致算法稳定性能差、收敛速度慢、寻优精度不高、时间花费久。本文在 GSO 算法基础上进行改进,提出一种针对多模态函数的聚类后再创种群的 PGSGSO 算法,并通过实验验证该算法的有效性。与相关文献所提的改进方法相比,总体上来说,PGSGSO 算法使用更少的迭代次数,更少的种群规模即可达到更好的效果。但本文所有测试函数均为二维空间中的多模态函数,因此下一步将运用 PGSGSO 算法对高维空间的多模态函数进行研究。

参考文献:

[1] KENNEDY J. Stereotyping: improving particle swarm performance with cluster analysis[C]// Congress on Evolutionary Computation. IEEE,2000:1507-1512.

[2] SHELOKAR P S, SIARRY P, JAYARAMAN V K, et al. Particle swarm and ant colony algorithms hybridized for improved continuous optimization [J]. Applied Mathematics & Computation,2007,188(1):129-142.

[3] AGRAWAL S, SILAKARI S. FRPSO: Fletcher-Reeves based particle swarm optimization for multimodal function optimization [J]. Soft Computing, 2014, 18 (11):2227-2243.

[4] FAN L, LIU X Y, JIA L P. A minimum-elimination-escape function method for multimodal optimization problems[C]// Tenth International Conference on Computational Intelligence and Security. IEEE,2014:312-316.

[5] ZHOU Z, XIE Y, PHAM D T, et al. Bees Algorithm for multimodal function optimisation[J]. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science,2015,203-210(3):1989-1996.
[6] ZHANG Y L, MA X P, GU Y, et al. A modified glowworm swarm optimization for multimodal functions [C]// Control and Decision Conference. IEEE, 2011: 2070-2075.
[7] ZHOU Y Q, ZHOU G, ZHANG J L. A hybrid glowworm swarm optimization algorithm to solve constrained multimodal functions optimization[J]. Optimization, 2015,64(4):1-24.
[8] 李敬明,倪志伟,许莹,等.基于二进制萤火虫算法的属性选择方法研究[J].系统科学与数学,2017(2):407-424.
[9] 祝华正,何登旭.一种小规模多种群萤火虫群优化算法[J].计算机工程与应用,2011,47(23):48-50.
[10] KRISHNANAND K N, GHOSE D. Detection of multiple source locations using a glowworm metaphor with applications to collective robotics[C]// Swarm Intelligence Symposium, 2005. Sis 2005. Proceedings. IEEE, 2005: 84-91.
[11] KRISHNANAND K N, GHOSE D. Multimodal Function Optimization using a Glowworm Metaphor with Applications to Collective Robotics [C]// Indian International Conference on Artificial Intelligence, Pune, India, December. DBLP,2005:328-346.
[12] 黄正新,周永权.自适应步长萤火虫群多模态函数优化算法[J].计算机科学,2011,38(7):220-224.
[13] ALJARAH I, LUDWIG S A. A MapReduce based glowworm swarm optimization approach for multimodal functions[C]// Swarm Intelligence. IEEE,2013:22-31.
[14] ZHOU Y Q, ZHOU G, ZHANG J. A hybrid glowworm swarm optimization algorithm for constrained engineering design problems[J]. Applied Mathematics & Information Sciences,2013,7(1):379-388.
[15] JAYAKUMAR D N, VENKATESH P. Glowworm swarm optimization algorithm with topsis for solving multiple objective environmental economic dispatch problem[J]. Applied Soft Computing, 2014, 23(5): 375-386.
[16] LUDWIG S A. Improved glowworm swarm optimization algorithm applied to multi-level thresholding [C]// Evolutionary Computation. IEEE,2016:1533-1540.
[17] 华罗庚,王元.数论在近似分析中的应用[M].北京:科学出版社,1978:83-86.
[18] 张铱,张铃.佳点集遗传算法[J].计算机学报,2001,24(9):917-922.
[19] 张军丽,周永权.一种用Powell方法局部优化的人工萤火虫算法[J].模式识别与人工智能,2011,24(5):680-684.
[20] CHENG S, SHI Y H, QIN Q D, et al. Population diversity maintenance in brain storm optimization algorithm[J]. Journal of Artificial Intelligence & Soft Computing Research,2015,4(2):83-97.
[21] 谢健,周永权,陈欢.一种基于Lévy飞行轨迹的蝙蝠算法[J].模式识别与人工智能,2013,26(9):829-837.
[22] TUO S H, ZHANG J Y, YONG L Q, et al. A harmony search algorithm for high-dimensional multimodal optimization problems[J]. Digital Signal Processing,2015,46(C):151-163.
[23] AHRARI A, ATAI A A. Grenade explosion method-a novel tool for optimization of multimodal functions[J]. Applied Soft Computing,2010,10(4):1132-1140.

A Parallel Search Good-Point Set Glowworm Swarm Optimization of Re-created Population after Clustering of Multi-Modal Functions

FANG Xian[1], *TIE Zhixin*[1], *LI Jingming*[2], *GAO Xiong*[1]
(1. School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou 310018, China;
2. School of Management, Hefei University of Technology, Hefei 230009, China)

Abstract: In the event that the glowworm swarm optimization algorithm is used to solve multi-modal function, a larger population size is usually needed to obtain an ideal result as the number of peak values increases. In addition, the uniform distribution of the initial population also has a great influence on the results. In view of this, a parallel search good-point set glowworm swarm optimization based on re-created population after clustering of multi-modal functions (PGSGSO) is proposed in this paper. Firstly, the glowworms are evenly distributed in the search space based on the good-point set theory, and the function peaks are captured with density-based clustering algorithm after rough search is finished, to recreate parallel spaces with the same number of peak values; a small amount of glowworms are added into the spaces for fine search to obtain the locally optimal solution of the parallel spaces and the globally optimal solution of the whole space. The comparison with the test results of other algorithms in 12 typical multi-modal functions show that the proposed algorithm is superior in respect of population size, convergence speed, searching precision, time cost and stability.

Key words: glowworm swarm optimization; multi-modal function; good-point set; density clustering algorithm; parallel search

(责任编辑:康 锋)