

改进的混合混沌烟花爆炸搜索算法

王培崇^{1,2}, 李丽荣³

(1 中国矿业大学(北京) 机电与信息工程学院, 北京 100833;

2 石家庄经济学院 信息工程学院, 河北 石家庄 050031; 3 石家庄经济学院 艺术设计学院, 河北 石家庄 050031)

摘要: 烟花爆炸搜索算法通过模拟烟花爆炸的炸点随机扩散进制, 实现对于解空间内最优解的寻优. 该算法具有搜索速度快, 参数较少的优点. 为了克服烟花爆炸算法容易收敛于局部最优的缺点, 提高算法的求解精度, 提出了一种混沌烟花爆炸搜索算法. 该算法在解空间内首先进行烟花爆炸搜索机制, 然后淘汰掉一部分适应度值低的个体, 并针对当前最佳个体执行混沌搜索, 收缩搜索空间. 最后在解空间内随机产生相应的部分个体. 多个仿真实验证明该算法求解精度高, 鲁棒性强.

关键词: 智能计算; 烟花爆炸搜索; 混沌搜索; 仿真实验

中图分类号: TP18

文献标识码: A

文章编号: 1000-7180(2014)11-0069-05

Improved Hybrid Chaos and Fireworks Explosion Search Algorithm

WANG Pei-chong^{1,2}, LI Li-rong³

(1 School of Mechanical Electronic & Information Engineering, China University of Mining & Technology(Beijing), Beijing 100083, China; 2 School of Information Engineering,

Shijiazhuang University of Economics, Shijiazhuang 050031, China;

3 School of Art Designing, Shijiazhuang University of Economics, Shijiazhuang 050031, China)

Abstract: To overcome the shortcoming which Fireworks Explosion Search(FES) algorithm convergents to local best easily and improve its precision, this paper proposed an improved hybrid chaos and fireworks explosion search algorithm. At first, it excutes FES in the solution space, and washing out some individuals which has low fitness. Then, it excutes Chaos search with global best individual and the search space is minished. Finally, some new individuals are generated randomly in the space. Some experiments on the classic benchmark functions show that the new improved algorithm is robust and has higher precision.

Key words: Intelligence Computation; Fireworks Explosion Search; Chaos Search; Simulator Experiments

1 引言

通过观察烟花爆炸的过程, 文献[1]中提出了一种新颖的智能搜索算法-烟花爆炸搜索^[1-3]算法(Fireworks Explosion Search, FES), 在 11 个 Benchmark 函数(高维和低维)上进行了测试并与其他多个改进的智能优化算法对比, 实验结果显示该算法具有较高的求解精度和全局收敛性. 为了提高该算法的求解精度克服算法早熟的缺点, 文献[2]

中在传统的爆炸搜索算法中引入了两个新算子: 迁移算子和变异算子. 利用迁移算子将远离当前最佳点的部分炸点向当前最佳点移动. 利用变异算子更新每一个炸点的位置, 从而保证种群的多样性. 实验证明改进的算法稳定性高、求解精度高. 文献[3]中利用凸函数理论对该算法的收敛性进行了证明, 并得出保优的 FES 算法能够以概率 1 收敛, 迁移算子和爆炸搜索算子能加速算法收敛^[4].

该算法原理简单, 参数较少, 求解速度快, 但是

收稿日期: 2014-01-20; 修回日期: 2014-03-11

基金项目: 河北省科技攻关基金项目(13214711)

算法也具有容易陷入局部最优、早熟等缺点^[2]. 考虑到混沌搜索^[5]算法的局部搜索能力较佳,能够跳出局部最优,可以较好地弥补 FES 算法的缺点,本文提出一个引入混沌搜索机制的 FES 算法.

2 烟花爆炸搜索算法

将解空间的寻优搜索问题看做是一个烟花爆炸覆盖问题,烟花爆竹自身的状态信息以及其所产生的各个炸点的状态信息均可以被视为候选解,设定相应的适应度函数即可以对这些炸点的状态进行评价,根据所求解问题选择出当前最佳解,并设定为下一次爆竹搜索的起点,淘汰其他部分炸点信息. 因为采用优胜劣汰的机制,如此迭代,如果炸点能够覆盖到全局最优解,则最后所产生的炸点必然集中在问题的最优解附近,最后的当前最佳解必然是问题的一个最佳可选解.

为了符合群体智能中术语的规范,以下将统一使用种群、个体等概念,不再使用炸点等概念. 引入算法而定各个参数如下:

r_0 : 初始爆炸半径; r_{end} : 最终的爆炸半径; T : 迭代次数; N : 种群数.

算法 1 FES 算法

step 1 初始化算法的各个参数;

step 2 在 n 维的解空间内随机初始化产生 N 个个体,组成一个种群,迭代次数 $t=1$;

step 3 种群中的每一个个体,在某点 $r_j \in [0, r]$ 上产生子代个体,公式如下: $X_i = X_i + r_j \cdot b_k (j=1, 2, 3, \dots, n)$, 式中, $b_k (k=1, 2, \dots, m)$ 为爆炸的方向向量, m 是爆炸的方向数. 如果所生成子个体的位置超过解空间范围,则设置为随机值.

step 4 当前最佳个体如果满足终止条件,则输出数据并终止算法;

step 5 修改爆炸范围的半径值 $r = (\frac{T-t}{T})^k (r_0 - r_{\text{end}}) + r_{\text{end}}$;

step 6 对种群中的个体按照其适应度值进行排序,仅保留 N 个最优个体,其余个体淘汰,返回 step 3 继续迭代.

根据文献^[1]中的研究成果,解空间维度 $n < 5$ 时,对每一个父个体沿着标准坐标轴的 $2n$ 个方向,分别以半径 r 、 $2r/3$ 、 $r/3$ 爆炸产生子个体;若 $n > 5$,则从 n 个标准坐标轴的正方向和反方向分别随机挑选出互不相同的三组方向,每组方向个数为 $n/5$,构成个数分别为 $2n/5$ 的三组方向 X 、 Y 、 Z . 父个体

沿着 X 中的方向按半径 r 产生子个体,沿着 Y 中的方向按半径 $2r/3$ 产生个体,沿 Z 中的方向按半径 $r/3$ 产生个体.

3 混合混沌烟花爆炸搜索算法

3.1 烟花爆炸搜索算法早熟分析

由上节的算法原理可知,该算法采用了优胜劣汰的方式淘汰适应度低的个体,收敛速度较快. 但是,因为 FES 算法是一种模拟自然现象的智能算法,缺少严密的数学理论支持,所以还无法从数学的角度严格说明该算法的收敛性和收敛能力. 采用马氏链全局收敛证明的前提是算法的种群数量足够多和进化时间足够长(甚至是无限长),而这在现实计算中是无法存在的,所以说这种概率性搜索算法本身是存在了一定的缺陷. 任何一种改进机制只能是针对算法的收敛性进行改善,而不是保证算法全局收敛.

影响算法提前早熟的主要原因之一是所求解问题的解空间分布. 由于 FES 算法中炸点的个体数量有限,其分布具有一定的随机性,对于较复杂的问题而言,当前优质解的前进方向不一定就是全局最优解的收敛方向. 在单峰问题中下面两种最情况就容易发生提前收敛. 第一,最优解区域附近种群个体适应度跨度变化较大;第二,代表最优解的个体和次优解的个体适应度相差较小. 而在多峰问题中,如果解与解之间相距较远,也很难保证种群搜索能够扩展到全部最优区域.

在实际的算法应用中,为了加快运算速度,采用的种群规模有限. 个体的初始状态一般随机生成,故炸点覆盖的解空间是不确定的. 在初始种群没有覆盖到全局最优解的情况下,如果在有限迭代次数内,无法覆盖到最优解区域,则算法早熟是难以避免的. 为了改善算法的收敛性,一方面要改善种群的多样性一方面要能够跳出局部最优的约束.

3.2 混沌搜索机制

考虑到混沌搜索机制可以很好地避免算法陷入局部最优,协助种群跳出局部极值的约束,故将两种算法混合以达到优势互补,有效改善爆炸搜索算法求解能力. 为了克服混沌搜索算法对于参数初值敏感的问题,将爆炸搜索算法的全局搜索结果作为混沌搜索的初始值,达到降低混沌搜索算法迭代次数的目的. 同时,在解空间内重新随机生成部分新个体,替换掉适应度低的个体,以提高种群的多样性.

混沌搜索首先需要执行 Logistic^[5]映射,利用

混沌映射函数 $x_{k+1} = u \cdot x_k(1 - x_k)$, ($0 \leq x_0 \leq 1$) 将混沌空间映射到待优化问题的解空间上;其次,针对该空间利用混沌动态特性实现求解.由混沌机制可知,当 $u = 4$ 并且 $x_0 \notin \{0, 0.25, 0.5, 0.75, 1\}$ 时, Logistic 映射序列呈现混沌动态特征.混沌搜索算法的进化公式是 $mx_i^{k+1} = 4mx_i^k(1 - mx_i^k)$, $i = 1, 2, \dots, n$, 其中 mx_j 为第 j 个混沌变量, k 为迭代次数, 当 $mx_i^0 \in (0, 1)$ 并且 $mx_i^0 \notin (0.25, 0.5, 0.75)$ 时, mx_i^k 将会在 $(0, 1)$ 区间上分布.

算法2 混沌搜索算法(CAS)

Step1 设定第 i 维变量的上界和下界 $x_{\min, i}$ 和 $x_{\max, i}$ 迭代次数 $t = 0$. 设决策变量为 x_i^k , 执行映射公式 $mx_i^k = \frac{x_i^k - x_{\min, i}}{x_i^k - x_{\max, i}} x_i^k$ ($i = 1, 2, \dots, n$);

Step2 计算下一次迭代的变量 $mx_i^{k+1} = 4mx_i^k(1 - mx_i^k)$, $i = 1, 2, \dots, n$;

Step3 转换混沌变量为决策变量 $x_i^{k+1} = x_{\min, i} + mx_i^{k+1}(x_{\max, i} - x_{\min, i})$, $i = 1, 2, \dots, n$.

Step4 根据决策变量 x_i^{k+1} , $i = 1, 2, \dots, n$ 对新解进行性能评价;

Step5 设初始解为 $\mathbf{X}^0 = [x_1^{(0)}, \dots, x_n^{(0)}]$, 若满足算法的终止条件, 则将新解输出, 否则返回 step 2.

算法3 混沌烟花爆炸搜索算法(CAFES)

Step1 初始化算法的各个参数, 最大迭代次数 maxiter 等;

Step2 在 n 维的解空间内随机初始化产生 N 个个体, 组成一个种群, 迭代次数 iter = 0;

Step3 种群中的每一个个体, 依据公式 $X_i = X_i + r_j \mathbf{b}_k$ ($j = 1, 2, 3, \dots, n$), 在某点 $r_j \in [0, r]$ 上产生子代个体. 式中 $\mathbf{b}_k$ ($k = 1, 2, \dots, m$) 爆炸的方向向量, m 是爆炸的方向数. 如果所生成子代个体的位置超过解空间范围, 则设置为该方向上的一个随机值;

Step4 对种群依据个体的适应度值进行排序, 仅保留 70% 的较优秀的个体, 淘汰其余 30%;

Step5 修改爆炸范围的半径值 $r = (\frac{T-t}{T})^k (r_0 - r_{\text{end}}) + r_{\text{end}}$;

Step6 对当前最佳个体 $\mathbf{X}_{\text{gbest}}$ 执行混沌搜索算法(CA), 并将 CA 的求解结果替换 $\mathbf{X}_{\text{gbest}}$;

Step7 若算法迭代次数满足或 $\mathbf{X}_{\text{gbest}}$ 在限定次数内基本不变化, 则输出 $\mathbf{X}_{\text{gbest}}$, 结束程序; 否则继续执行如下步骤.

Step8 按下述公式收缩搜索区域.

$$x_{\min, i} = \max(x_{\min, i}, x_{g, i} - \alpha(x_{\max, i} - x_{\min, i})), 0 < \alpha < 1,$$

$$x_{\max, i} = \min(x_{\max, i}, x_{g, i} + \alpha(x_{\max, i} - x_{\min, i})), 0 < \alpha < 1.$$

注: $x_{g, i}$ 表示当前最佳 $\mathbf{X}_{\text{gbest}}$ 的第 i 维变量的值.

Step9 在初始的种群空间内随机产生其余的新个体, 保证总体群体数量为 N , 然后返回 step 3.

从上述算法运算步骤可以看出, 烟花爆炸搜索算法每执行一次之后, 将会针对当前最佳 $\mathbf{X}_{\text{gbest}}$ 进行混沌搜索 CAS, 以使算法能够跳出局部最优的约束. 另外, 在步骤 9 中, 通过在解空间中随机产生部分新个体, 保证了种群的多样性. 两种机制的融合能够使 FSE 算法在一定程度上克服算法的早熟现象.

4 实验结果及分析

4.1 仿真实验 1

为了验证算法的求解效果, 利用开源 SCILAB 的实验环境对本文中的两个算法(FES、CAFES)进行实现, 并与其他两个算法进行对比. 计算机配置的硬件环境是, Intel 赛扬 3.0 GHz 双核, 内存 4 GHz, 软件环境配置 SCILAB5. 4. 1. FSE、CAFSE 算法的种群规模均设定一致为 20, $r_0 = \beta \times (X_{\max} - X_{\min})$, $\beta \in [0.05, 0.3]$, $r_{\text{end}} = 10^{-4}$, 迭代次数 1 000 次. 两个算法均运行 30 次, 对比求解结果的均值和方差两项. 实验结果见表 1, 表中列出了 FES 算法、CAFES 算法、PSO 算法、GA 算法的实验结果, 其中 PSO 算法以及 GA 算法的数据直接取自参考文献 [4]. 测试用 10 个 Benchmark 函数如下所示.

$$(1) f(x) = \sum_{i=1}^{30} x_i^2, (|x_i| \leq 100);$$

$$\text{最佳: } \min(f(x^*)) = f(0, 0 \dots 0) = 0;$$

$$(2) f(x) = \sum_{i=1}^{30} |x_i| + \prod_{i=1}^{30} |x_i|, (|x_i| \leq 10);$$

$$\text{最佳: } \min(f(x^*)) = f(0, 0 \dots 0) = 0;$$

$$(3) f(x) = \sum_{i=1}^{30} ([x + 0.5])^2, (|x_i| \leq 100);$$

$$\text{最佳: } \min(f(x^*)) = f(0, 0 \dots 0) = 0;$$

$$(4) f(x) = -\sum_{i=1}^{30} (x_i \sin \sqrt{x_i}), (|x_i| \leq 500);$$

$$\text{最佳: } \min(f(x^*)) = f(420.9687, 420.9687, \dots, 420.9687) = -12569.5;$$

$$(5) f(x) = \sum_{i=1}^{30} [x_i^2 - \cos(2\pi x_i) + 10], (|x_i| \leq$$

$$5.12);$$

最佳: $\min(f(x^*)) = f(0, 0 \cdots 0) = 0$;

(6) $f(x) =$

$$\sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2, (|x_i| \leq 5);$$

最佳: $\min(f(x^*)) = f(0.192\ 8, 0.198\ 0, 0.123\ 1, 0.135\ 8) \approx 0.000\ 307\ 5$,

其中,

$a_i = (0.195\ 7, 0.194\ 7, 0.173\ 5, 0.16, 0.084\ 4, 0.062\ 7, 0.045\ 6, 0.034\ 2, 0.032\ 3, 0.023\ 5, 0.024\ 6)$,
 $(1/b_i) = (0.25, 0.5, 1, 2, 4, 6, 8, 10, 12, 14, 16)$;

$$(7) f(x) = \frac{1}{500} + \sum_{i=1}^{25} \frac{1}{j + \sum_{i=1}^2 (x_i - a_{ij})^6},$$

$(|x| \leq 65.56)$;

最佳: $\min(f(x^*)) = f(-32, 32) \approx 1$, 其中,

$a_{ij} =$

$$\begin{bmatrix} -32, & -16, & 0 & 16, & 32, & -32 \dots & 0, & 16, & 32 \\ -32, & -32, & -32, & -32, & -32, & -16, \dots & 32, & 32, & 32 \end{bmatrix};$$

(8) $f(x) = 4x_1^2 - 2.1x_1^4 + x_1^6/3 + x_1x_2 - 4x_2^2 + 4x_2^4, (|x_i| \leq 5)$;

最佳: $\min(f(x^*)) = f(0.089\ 83, -0.712\ 6) = f(-0.089\ 83, 0.712\ 6) = -1.031\ 628$;

(9) $f(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi} - 6)^2 + 10(1 - \frac{1}{8\pi})\cos x_1 + 10 (-5 \leq x_1 \leq 10, 0 \leq x_2 \leq 15)$;

最佳: $\min(f(x^*)) = f(-3.142, 2.275) = f(3.142, 2.275) = f(9.425, 2.425) = 0.398$;

(10) $f(x) = -\sum_{i=1}^4 c_i \exp[-\sum_{i=1}^3 a_{ij}(x_i - p_{ij})^2]$,
 $(0 \leq x_i \leq 2)$;

最佳: $\min(f(x)) = f(0.114, 0.556, 0.852) = -3.862$,

其中,

$$a_{ij} = \begin{bmatrix} 3, & 12, & 30 \\ 0.1, & 10, & 35 \\ 3, & 10, & 30 \\ 0.1, & 10, & 35 \end{bmatrix}, c_i = (1, 1, 2, 3, 3, 2),$$

$$p_{ij} = \begin{bmatrix} 0.368\ 9, & 0.117\ 0, & 0.2673 \\ 0.469\ 9, & 0.438\ 7, & 0.7670 \\ 0.109\ 1, & 0.873\ 2, & 0.5547 \\ 0.0381\ 5, & 0.574\ 3, & 0.8828 \end{bmatrix}.$$

为了进一步对比 FES 和 CAFES 两个算法的性能,表 2 中列出了这两个算法在上述 10 个 Benchmark 函数上找到最佳解的成功次数和找到最佳解

(阈值 0.01)的平均收敛代数两个指标。

从表 1 所示的实验数据可以看出,本文的改进算法 CAFES 在这 10 个 Benchmark 测试函数上的求解精度 还是很高的,明显优于 PSO、GA 和 FES 三个算法的表现.其他的三个算法能够找到最优解时,CAFES 算法也能找到最优解;在部分函数上其他函数找不到最优解的情况下,CAFES 算法同样找到了最优解.当所有算法都不能找到最优解的情况下,CAFES 算法的求解精度要明显优于其他三个算法,更接近最优解.另外,通过方差的比较可以知道,CAFES 的方差更小,说明算法运行更加稳定、鲁棒性更佳。

表 1 多个算法在 10 个函数上的结果对比

函数	维度	PSO		GA	
		均值	方差	均值	方差
f1	30	6.43e+2	83.762 7	2.45e-4	3.84e-4
f2	30	1.17e+1	11.215 6	2.96e-1	0.253 2
f3	30	1.02e+3	1 273.31	1.72e+1	7.169 8
f4	30	-7 102.87	958.943	-9 569.48	626.573
f5	30	107	42.637 6	6.37	3.462 3
f6	4	3.65e-3	6.83e-3	3.46e-3	1.43e-2
f7	2	0.002	1.41e-18	0.002	1.23e-16
f8	2	-1.03	4.511e-16	-1.03	7.601e-12
f9	2	0.398	0	0.398	5.1e-12
f10	3	-3.862	2.71e-15	-3.862	2.71e-15

续表 1

函数	维度	FES		CAFES	
		均值	方差	均值	方差
f1	30	6.81e-12	9.15e-12	7.81e-28	1.83e-64
f2	30	3.42e-13	5.62e-12	2.92e-20	3.213e-20
f3	30	2.971e-91	1.262e-99	0	0
f4	30	-10 175.21	869.098 1	-12 018.02	20.872
f5	30	3.881	1.017	4.13e-19	7.54e-18
f6	4	1.73e-3	2.48e-3	3.061e-4	9.07e-10
f7	2	0.002 92	4.95e-17	0.978 56	6.832e-81
f8	2	-1.189 1	3.418e-9	-1.031 901	1.621e-76
f9	2	0.398	2.716e-10	0.398	0
f10	3	-3.862	1.928 7e-11	-3.862	7.662e-198

从表 2 中的数据对比可以看出 CAFES 算法寻找到最佳解的次数明显多于 FES 算法,并且其找到最佳解的平均迭代次数也要优于 FES 算法.通过单独对比 CAFES 与 FES 两个算法,可以看出算法的改进是非常有效的,通过引入混沌搜索机制能够较

好克服 FES 算法的早熟,提高了算法的求解精度.

表 2 FES 算法和 CAFES 算法收敛代数数和成功率对比

函数	FES		CAFES	
	成功次数	平均收敛代数	成功次数	平均收敛代数
f_1	26	189.51	28	130.78
f_2	26	210.13	26	101.67
f_3	29	265.08	30	189.18
f_4	23	351.50	26	209.65
f_5	21	286.45	27	100.45
f_6	22	258.32	27	98.87
f_7	20	300.65	28	146.75
f_8	28	259.10	29	118.87
f_9	28	242.08	30	96.76
f_{10}	28	246.72	30	132.54

4.2 仿真实验 2

为进一步衡量该算法的性能,将 CAFES 与现有的其他的相关改进算法 ESA^[2]进行对比,所使用 4 个 Benchmark 函数参见文献[2]. CAFES 算法的种群规模均设定为 100, $r_0 = \beta \times (X_{\max} - X_{\min})$, $\beta \in [0.07, 0.5]$, $r_{\text{end}} = 10^{-6}$,迭代次数 2 000,算法运行 30 次,对比项目包括平均值和标准差两个,对比实验结果见表 3,表中 ESA 算法的数据直接取自文献[2]. 两个算法在四个标准函数上的获得最优解的平均收敛次数和成功次数对比列于表 4.

表 3 ESA 与 CAFES 在 4 个 Benchmark 函数上的求解结果对比

算法	对比项目	f_1	f_2	f_3	f_4
ESA	平均值	3.51e-23	-12 569.50	0.000	7.08e-15
	标准差	8.21e-23	5.37e-5	0.000	2.24e-15
CAFES	平均值	4.63e-33	-12 569.46	0.000	9.04e-56
	标准差	2.78e-79	8.53e-10	0.000	2.36e-45

从表 3、表 4 所列的数据对比可以看出,CAFES 算法在 f_2 函数上的平均值以及成功次数较 ESA 稍差.但是,CAFES 算法在求解 f_2 函数取得成功时,其迭代次数明显较 ESA 少.在其余三个函数上的表现

均较 ESA 优越.这说明,增加混沌搜索机制要较简单的采用增强邻域搜索机制更能使算法跳出局部极值的约束.该算法的改进是比较成功的.

表 4 ESA 与 CAFES 的平均迭代次数等对比

函数	FES		CAFES	
	成功次数	平均收敛代数	成功次数	平均收敛代数
f_1	28	103.65	29	127.18
f_2	30	810.46	26	154.36
f_3	30	535.90	30	216.02
f_4	23	189.73	27	209.81

5 结束语

分析了爆炸搜索算法缺点,并在该算法中通过引入混沌局部搜索机制,来改善爆炸搜索算法的全局搜索能力,克服其容易收敛于局部最优造成算法早熟的问题.通过仿真实验可知,改进的 CAFES 算法计算精度高,鲁棒性强,是一个值得推广的全局优化算法.爆炸搜索算法诞生时间不长,进一步的研究应该针对如何应用该算法解决离散问题以及约束优化问题展开.

参考文献:

- [1] 曹炬,贾红,李婷婷. 烟花爆炸优化算法[J]. 计算机工程与科学,2011,33(1):138-142.
- [2] 曹炬,侯学卿. 带有邻域搜索机制的爆炸搜索算法[J]. 计算机工程,2011,36(18):183-187.
- [3] 曹炬,侯学卿. 爆炸搜索算法及其收敛性证明[J]. 计算机科学,2011,38(11):231-233.
- [4] 侯学卿. 爆炸搜索算法研究[D]. 武汉:华中科技大学,2011.
- [5] 邱劲,王平. 基于混沌映射的压缩图像加密算法[J]. 计算机科学,2012,39(6):44-46.
- [6] 曹炬,李婷婷,贾红. 带有遗传算子的烟花爆炸优化算法[J]. 计算机工程,2010,36(12):149-151.

作者简介:

王培崇 男,(1972-),博士,副教授.研究方向为模式识别.