DOI:10.13718/j.cnki.xsxb.2020.11.009

混沌小生境萤火虫算法求解有界背包问题①

任静敏[1], 潘大志[1,2]

1. 西华师范大学 数学与信息学院,四川 南充 637009;2. 西华师范大学 计算方法与应用研究所,四川 南充 637009

摘要:针对有界背包问题,提出一种混沌小生境萤火虫算法.采用混沌理论对萤火虫种群初始化,为了增加种群多样性,使用小生境技术计算个体共享适应度,以一定概率将共享半径内相似个体进行排挤处理,对所有被排挤个体实行 Levy 飞行操作,同时对较优个体进行局部搜索,对陷入早熟的个体使用混沌理论重新随机产生位置进行更新.仿真实验表明新改进算法能有效求解有界背包问题.

关 键 词:萤火虫算法;有界背包问题;混沌理论;小生境技术;Levy 飞行

中图分类号:TP18 **文献标志码**:A **文章编号**:1000-5471(2020)11-0059-07

针对实际问题,背包问题[1]可构造出一系列子问题,如 0-1 背包问题[2]、有界背包问题[3]、折扣{0-1}背包问题[4]等.目前求解有界背包问题(BKP)的方法有精确算法和近似算法[5-7].

本文在标准萤火虫算法的基础上结合混沌理论和小生境技术提出混沌小生境萤火虫算法(NCFA)用于求解有界背包问题,并与小生境萤火虫算法(ULFA)、带权重的贪心萤火虫算法(WGFA)[8]进行对比实验,结果表明本文算法能有效解决有界背包问题.

1 有界背包问题

有界背包问题[3]是组合优化中经典的 NP 问题之一,其具体描述如下:

考虑背包问题中的背包容量为 C,使用 n 项特定的物品来填充它,其中第 i 项物品价值为 p_i,对应重量为 w_i,可选择物品个数最多为 b_i,现选择若干个物品,在保证总重量不超过背包容量 C 的情况下使得背包内物品的价值总和最大.有界背包问题的数学模型如下:

$$\max \quad z=\sum_{i=1}^{n} p_i x_i \tag{1}$$

$$\text{subject} \quad \text{to} \quad \sum_{i=1}^{n} w_i x_i \leqslant C \tag{2}$$

$$x_i \in \{0,1,\cdots,b_i\} \tag{3}$$

其中:系数 $p_i, w_i, b_i (i=1,2,\cdots,n)$ 和 C 均为正整数.

2 算法思想

2.1 标准萤火虫算法

萤火虫算法[9]是基于萤火虫闪光行为的一种启发式算法.萤火虫的吸引度取决于它的亮度,亮度越强

① 收稿日期:2019-07-05
基金项目:国家自然科学基金项目(11871059);四川省教育厅自然科学基金项目(18ZA0469);西华师范大学英才科研基金项目(17YC385);西华师范大学校级科研团队项目(CXTD2015-4).
作者简介:任静敏(1993-),女,硕士研究生,中国计算机学会(CCF)会员,主要从事智能算法研究.
通信作者:潘大志,博士,教授,硕士生导师.

吸引度就越强，因而可以将萤火虫亮度与有界背包问题的目标函数相关联，可将目标函数值直接作为萤火虫 $x_i(i=1,2,\cdots,n)$ 的亮度：

$$I_0 = f(x_i) \tag{4}$$

同时萤火虫的一部分亮度在传输过程中会被介质吸收. 在最大化问题中，萤火虫相对亮度较低者会被相对亮度较高者吸引，相对亮度受萤火虫自身亮度和距离影响，其描述如下：

$$I(r) = I_0 \mathrm{e}^{-\gamma r^2} \tag{5}$$

其中：γ 为光强吸收因子，r 为两萤火虫之间的欧式距离. 两只萤火虫间的吸引度也将随着萤火虫 i 和萤火虫 j 之间的距离 r_{ij} 变化而变化. 因此，可以定义萤火虫的吸引度 β 为

$$\beta(r) = \beta_0 \mathrm{e}^{-\gamma r^2} \tag{6}$$

其中：β_0 为萤火虫在距离为 $r=0$ 时的吸引度，γ 为光强吸收因子，r 为两萤火虫之间的欧氏距离. 若萤火虫 i 向萤火虫 j 的方向移动，移动公式如下：

$$x_i^{t+1} = x_i^t + \beta(x_j^t - x_i^t) + \alpha\left(rand - \frac{1}{2}\right) \tag{7}$$

其中：x_i^t, x_j^t 为萤火虫 i 和萤火虫 j 在第 t 代时所处的位置；α 为步长因子，$rand$ 为一个从0到1均匀分布的随机数，$\alpha\left(rand - \frac{1}{2}\right)$ 为随机搜索项.

2.2 混沌思想

混沌是自然界广泛存在的一种非线性现象，具有随机性、遍历性、初始条件敏感性等特点[10]，本文采用的混沌映射是立方映射[11]，其表达式为：

$$y(n+1) = 4y(n)^2 - 3y(n) \qquad -1 \leqslant y(n) \leqslant 1,\ n = 0,1,2,\cdots \tag{8}$$

实际问题中，在初始值不为0的情况下都会产生混沌，对比 logistic 映射，立方映射产生的混沌序列均匀性较好，这里将其用于萤火虫种群的初始化.

2.3 Levy 飞行

Levy 飞行[12] 是一种随机漫步，其步长服从于一个重尾的概率分布，使用 Levy 飞行的位置更新如下所示：

$$x_i^{t+1} = x_i^t + \alpha \oplus d_{\mathrm{Levy}}(\lambda) \tag{9}$$

其中：x_i^t 表示个体 i 在第 t 代的位置，$\oplus$ 为点对点乘法，α 为步长控制量，$d_{\mathrm{Levy}}(\lambda)$ 为随机步长，$i=1,2,\cdots,n$. 由于 Levy 飞行具有快速增长且方差无限的特点，于是运用于大规模搜索时，其搜索效果比较好[13]. 由于 Levy 飞行是短距离的探索和偶尔长距离的飞行相结合，所以若对更新后的萤火虫全部进行 Levy 飞行将不能保护优秀个体. 本文为了使较差萤火虫得到充分利用，于是选取惩罚个体的 $\frac{1}{5}$ 进行 Levy 飞行. 目的是种群在增加全局搜索能力的同时避免破坏较优个体.

2.4 小生境技术

小生境技术[14] 包含了共享机制与排挤机制. 传统的小生境共享机制采用单一的共享半径与两个个体的距离进行比较，虽然能判断多个子群内的拥挤程度，但是并不能判断子群内个体的优劣. 本文提出计算小生境内个体的共享函数如下

$$Sh(d_{ij}) = \begin{cases} 1 - \dfrac{f_{ibest} f_{iavg}}{e_1 e_2}, & f_{ibest} < e_1 \,\&\, f_{iavg} < e_2 \\ 1 - \dfrac{f_{ibest}}{e_1}, & f_{ibest} < e_1 \,\&\, f_{iavg} \geqslant e_2 \\ 1 - \dfrac{f_{iavg}}{e_2}, & f_{ibest} \geqslant e_1 \,\&\, f_{iavg} < e_2 \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

其中：f_{ibest} 和 f_{iavg} 分别表示个体 i 与最好个体的适应度差异和个体 i 与其他个体的平均适应度差异，e_1 和 e_2

表示个体 i 的两种共享半径

$$e_1 = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} + \sum_{i=1}^{n}\frac{f(x_i)-f_{\min}}{f(x_i)} + \sum_{i=1}^{n}\frac{f(x_i)}{f(x_i)+f_{\max}}}{n^2} \tag{11}$$

$$e_2 = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} |f(x_i)-f(x_j)|}{n^2} \tag{12}$$

种群内个体的共享适应度为

$$f_s(x_i) = \frac{f(x_i)}{\sum_{j=1}^{n} Sh(d_{ij})} \tag{13}$$

其中：n 为种群内的个体数，d_{ij} 为小生境内个体间的欧氏距离. 为了保证种群多样性，本文根据共享适应度对所有个体进行排序，并对较差的部分个体进行惩罚：

$$f_s(x_i) = penalty \times f_s(x_i) \tag{14}$$

其中 $penalty$ 为惩罚系数.

2.5　局部搜索

为了使算法快速找到问题的最优解，本文利用每次迭代产生的除最优个体外前 k 个较优个体产生新的位置，并将其用于替换相等数量的最差个体，位置更新公式如下

$$x_{new} = \lambda x_{best} + (1-\lambda)x_i \tag{15}$$

其中：x_{new} 为更新后的个体，x_{best} 为最优个体，λ 为一个属于(0,1) 的随机数，x_i 为除最优个体外的前 k 个较优个体 $i = 2,3,\cdots,k+1$.

2.6　编码方式

由于有界背包问题是离散化问题，而萤火虫算法针对连续型问题求解，于是需要一种编码方式将问题中的连续解离散化，本文采用如下编码方式：

$$x_{ik} = \begin{cases} bound(k), & x_{ik} \geqslant bound(k) \\ \lceil x_{ik} - 0.5 \rceil, & 0 < x_{ik} < bound(k) \\ 0, & x_{ik} \leqslant 0 \end{cases} \tag{16}$$

其中：$bound(k)$ 为背包内第 k 项物品的可选择数上限，x_{ik} 为第 i 个个体的第 k 个分量，$i = 1,2,\cdots n$.

2.7　对不可行解的修复处理

由于萤火虫算法在求解有界背包问题过程中会产生大量不可行解，本文采用文献[6] 提出的一种改进贪心算法对不可行解进行修复.

3　算法实现的具体流程

改进后的萤火虫算法(NCFA) 用于求解有界背包问题实施步骤具体如下：

Step1 初始化参数，利用混沌理论对萤火虫种群初始化.

Step2 根据式(4) — (7) 更新萤火虫位置，利用改进贪心算法修复不可行解.

Step3 根据式(10) — (13) 计算 Step2 中解的共享适应度并排序，对共享半径内的较差个体进行惩罚.

Step4 对种群内被惩罚个体选取一定比例进行 Levy 飞行，并计算种群内萤火虫适应度.

Step5 利用局部搜索得到的新解替换较差解，利用改进贪心算法修复不可行解.

Step6 计算种群内个体适应度，更新全局最优值及最优解.

Step7 判断是否达到结束条件，若是，转 Step8；若否，转 Step2.

Step8 输出全局最优值.

4 仿真实验

本文利用3类BKP实例(UBKP,WBKP,SBKP)对混沌小生境萤火虫算法进行测试，每一类都包含10个规模在100～1 000的实例，所有实例数据均来自于文献[6].

本文仿真实验所用硬件环境为：AMD A6－3420M APU with Radeon (TM) HD Graphics 1.50 GHz，内存为4.0 GB，操作系统为64位Windows 7旗舰版. 编程语言为MATLAB，编译环境为MATLAB R2018a，并使用MATLAB在编译环境中绘制折线图.

为了验证本文算法的有效性，对3类BKP实例进行仿真实验，并与其他两类算法(ULFA,WGFA[8])实验数据进行对比，其中ULFA是融合了小生境技术与改进贪心算子的萤火虫算法，将其作为测试本文算法的对比算法. 各算法的参数设置见表1.

表1 求解BKP的3种算法参数设置

Algorithm	种群大小	最大迭代次数/次	γ	β_0	α	惩罚系数
ULFA	60	200	0.6	1	0.6	—
WGFA	60	200	0.6	1	0.6	—
NCFA	60	100	0.6	1	0.6	1e-5

表1中γ为光强吸收因子，β_0为吸引度，α为步长因子. 为了直观比较各算法性能，本文将各算例分别独立运行50次，利用测试结果与理论最优值进行对比，并比较各算法间的最优值、平均值、最差值以及标准差. 实验结果见表2－4，其中Opt是理论最优值，Best是最优值，Mean是平均值，Worst是最差值，Std是标准差.

表2 ULFA,WGFA,NCFA求解UBKP1-UBKP10的实验结果

		UBKP1	UBKP2	UBKP3	UBKP4	UBKP5	UBKP6	UBKP7	UBKP8	UBKP9	UBKP10
	Opt	201 616	414 114	594 613	831 629	1 003 643	1 228 085	1 524 770	1 692 853	1 869 142	2 066 060
	Best	201 616	414 114	594 598	831 592	1 003 622	1 228 073	1 524 770	1 692 835	1 869 108	2 066 031
ULFA	Mean	201 533	414 114	594 540	831 581	1 003 594	1 228 073	1 524 736	1 692 835	1 869 096	2 066 026
	Worst	201 499	414 114	594 523	831 580	1 003 589	1 228 073	1 524 726	1 692 835	1 869 095	2 066 025
	Std	41.54	0	25.49	3.69	10.52	0	15.04	0	3.24	1.28
	Best	201 499	414 114	594 586	831 612	1 003 628	1 228 083	1 524 755	1 692 835	1 869 131	2 066 048
WGFA	Mean	201 499	414 114	594 558	831 600	1 003 614	1 228 075	1 524 739	1 692 835	1 869 111	2 066 027
	Worst	201 499	414 114	594 526	831 580	1 003 589	1 228 073	1 524 739	1 692 835	1 869 095	2 066 025
	Std	0	0	19.27	14.34	13.54	4.39	2.87	0	14.11	5.63
	Best	201 616	414 114	594 606	831 629	1 003 642	1 228 085	1 524 770	1 692 846	1 869 131	2 066 060
NCFA	Mean	201 613	414 114	594 598	831 612	1 003 636	1 228 075	1 524 734	1 692 836	1 869 107	2 066 027
	Worst	201 590	414 114	594 593	831 599	1 003 604	1 228 073	1 524 726	1 692 835	1 869 095	2 065 025
	Std	7.91	0	5.71	5.99	8.32	3.74	14.42	2.84	11.17	8.76

表3 ULFA,WGFA,NCFA求解WBKP1-WBKP10的实验结果

		WBKP1	WBKP2	WBKP3	WBKP4	WBKP5	WBKP6	WBKP7	WBKP8	WBKP9	WBKP10
	Opt	119 312	297 700	444 156	605 678	772 191	890 314	1 045 302	1 210 947	1 407 365	1 574 079
	Best	119 308	297 700	444 142	605 653	772 178	890 297	1 045 298	1 210 936	1 407 364	1 574 070
ULFA	Mean	119 292	297 698	444 142	605 653	772 169	890 290	1 045 291	1 210 936	1 407 364	1 574 066
	Worst	119 272	297 681	444 142	605 653	772 168	890 289	1 045 291	1 210 936	1 407 364	1 574 066
	Std	10.38	0	0	0	2.92	3.05	2.13	0	0	1.65
	Best	119 309	297 700	444 147	605 671	772 182	890 305	1 045 297	1 210 944	1 407 364	1 574 069
WGFA	Mean	119 308	297 700	444 142	605 656	772 176	890 295	1 045 291	1 210 944	1 407 364	1 574 067
	Worst	119 305	297 700	444 142	605 653	772 174	890 289	1 045 291	1 210 944	1 407 364	1 574 067
	Std	1.03	0	1.54	5.53	3.25	5.03	1.86	0	0	0.48
	Best	119 312	297 700	444 156	605 678	772 188	890 312	1 045 300	1 210 945	1 407 364	1 574 076
NCFA	Mean	119 308	297 700	444 146	605 668	772 185	890 307	1 045 296	1 210 936	1 407 364	1 574 069
	Worst	119 297	297 700	444 142	605 657	772 183	890 296	1 045 291	1 210 924	1 407 364	1 574 066
	Std	2.72	0	3.94	5.20	1.34	3.97	3.02	4.11	0	3.09

表 4　ULFA,WGFA,NCFA 求解 SBKP1-SBKP10 的实验结果

		SBKP1	SBKP2	SBKP3	SBKP4	SBKP5	SBKP6	SBKP7	SBKP8	SBKP9	SBKP10
ULFA	Opt	144 822	259 853	433 414	493 847	688 246	849 526	1 060 106	1 171 576	1 263 609	1 412 095
	Best	144 822	259 853	433 414	493 847	688 246	849 526	1 060 106	1 171 576	1 263 609	1 412 095
	Mean	144 807	259 841	433 402	493 835	688 233	849 517	1 060 093	1 171 567	1 263 597	1 412 084
	Worst	144 787	259 817	433 378	493 847	688 246	849 526	1 060 106	1 171 576	1 263 609	1 412 095
	Std	9.67	9.29	7.52	10.70	8.52	6.14	11.58	6.09	8.08	9.08
WGFA	Best	144 820	259 853	433 414	493 847	688 246	849 526	1 060 106	1 171 576	1 263 609	1 412 095
	Mean	144 817	259 852	433 412	493 844	688 243	849 521	1 060 101	1 171 569	1 263 604	1 412 087
	Worst	144 812	259 852	433 408	493 841	688 240	849 513	1 060 086	1 171 560	1 263 591	1 412 067
	Std	1.76	0.18	1.72	2.35	1.97	3.95	3.56	4.58	4.28	6.80
NCFA	Best	144 822	259 853	433 414	493 847	688 246	849 526	1 060 106	1 171 576	1 263 609	1 412 095
	Mean	144 820	259 852	433 413	493 845	688 243	849 523	1 060 100	1 171 572	1 263 605	1 412 087
	Worst	144 815	259 847	433 408	493 837	688 233	849 515	1 060 086	1 171 559	1 263 590	1 412 069
	Std	2.16	1.61	1.56	2.68	3.62	2.74	5.14	4.72	5.07	6.88

通过表 2－4 可以看出 3 种算法用于求解不同类型的有界背包问题的结果. 对于 UBKP 而言，第 1,2,4,6,7,10 例算例本文算法均可找到理论最优值，即使不能找到最优值，最终结果也会优于另外两种算法；对于 WBKP 而言，第 1,2,3,4 例算例本文算法能寻到理论最优值，除第 9 例算例外，其他各例均优于其他两类算法；对于 SBKP 而言，3 种算法的寻优结果都较好. 本文参考文献[6]的平均近似比和最佳近似比对以上 3 类算例进行对比，对比结果见图 1－3.

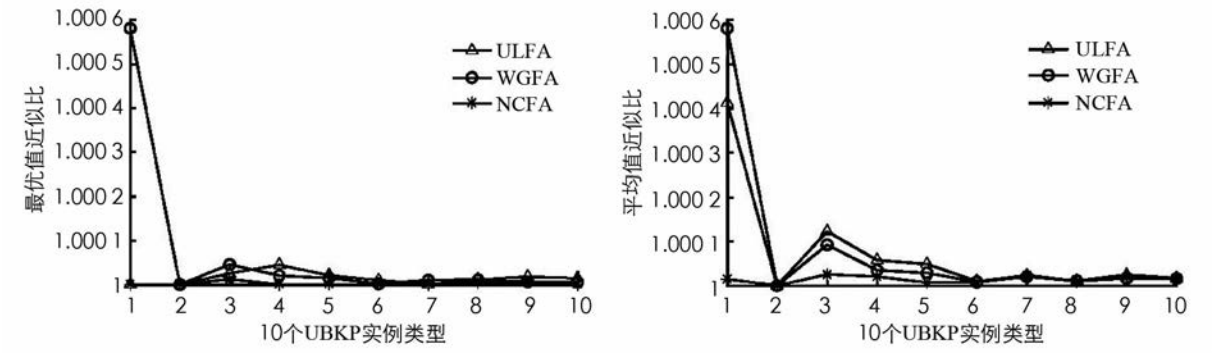

图 1　3 种算法针对 10 组 UBKP 算例的近似比

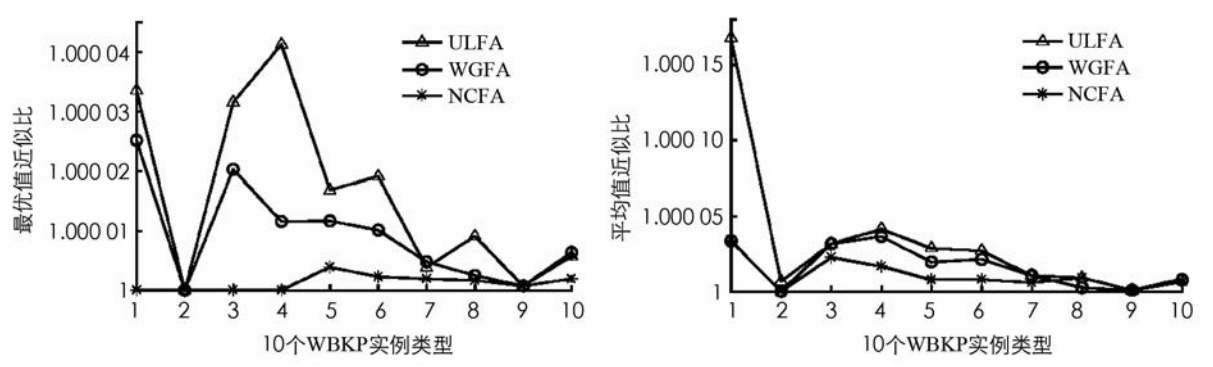

图 2　3 种算法针对 10 组 WBKP 算例的近似比

从图 1－3 中 3 组算法针对 3 类问题的近似比可以看出，NCFA 除了在计算 WBKP 实例 8 时的平均值近似比劣于其他两种算法外，其他近似比均优于另外两种算法，并且 NCFA 的近似比曲线较平稳，说明该算法用于求解 BKP 问题更有效. 为了更直观比较 3 种算法寻优所需要的迭代次数，下面给出 3 种算法的平均收敛曲线(图 4－6).

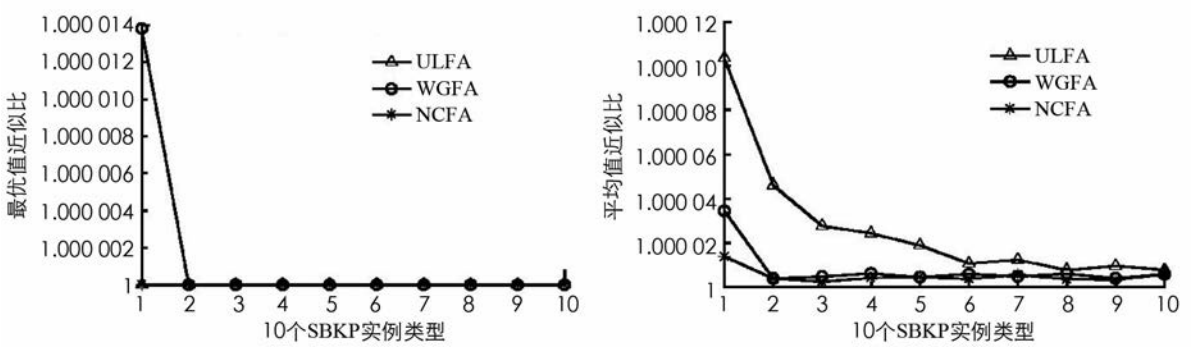

图3 3种算法针对10组SBKP算例的近似比

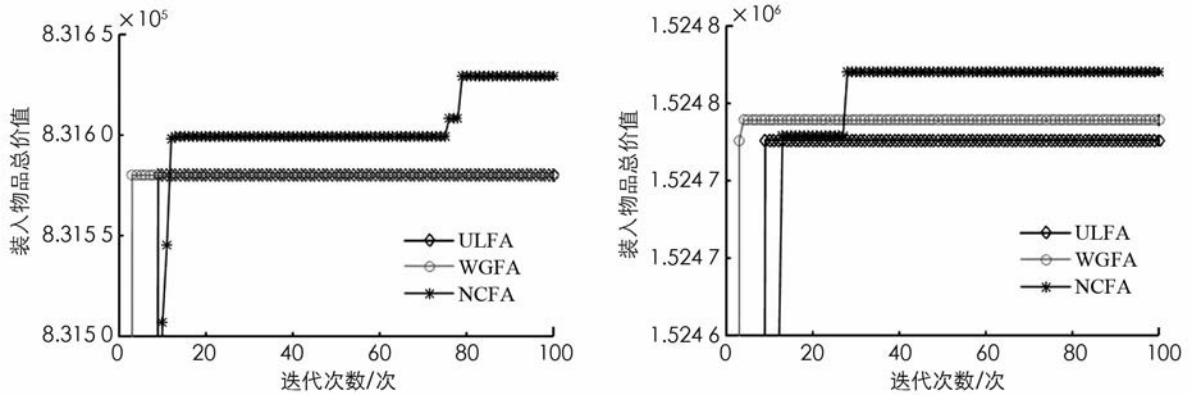

图4 3种算法求解UBKP4与UBKP7的收敛曲线

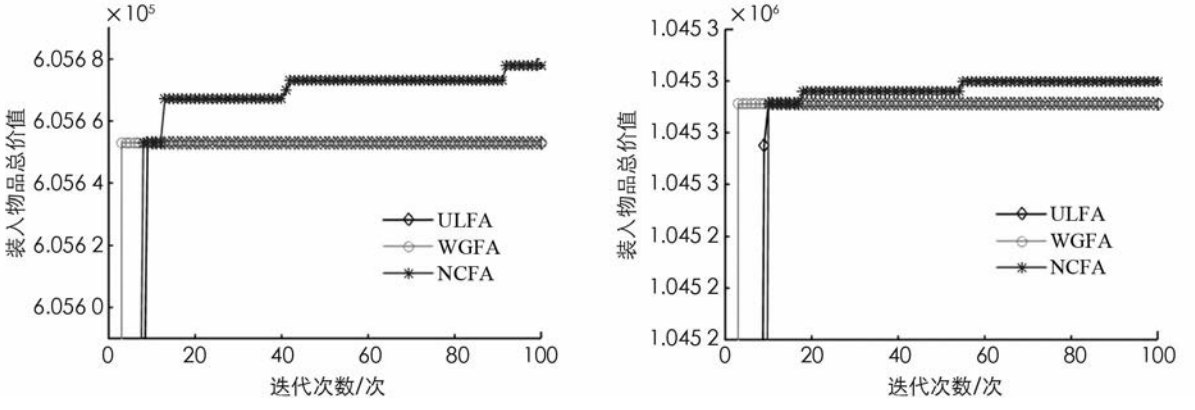

图5 3种算法求解WBKP4与WBKP7的收敛曲线

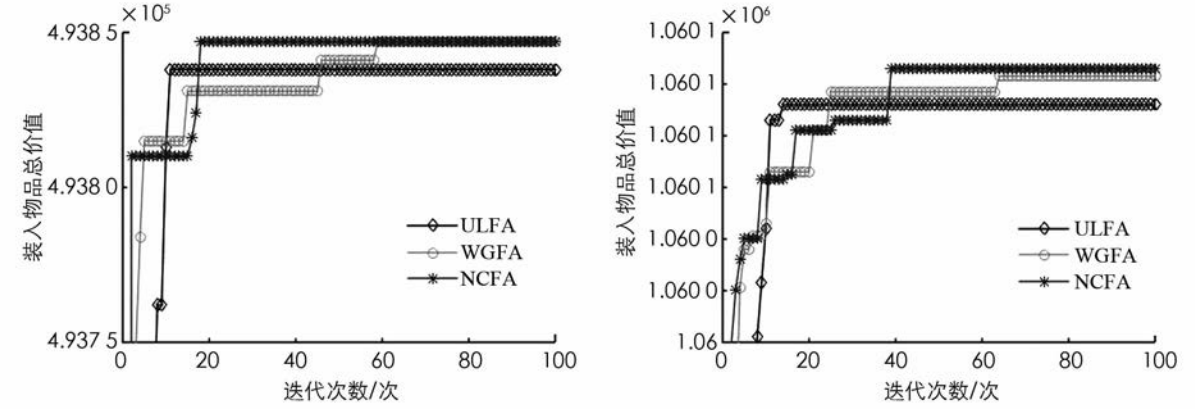

图6 3种算法求解SBKP4与SBKP7的收敛曲线

从图4—6收敛曲线可以看出，本文算法均能在100代内搜索到理论最优值，在求解WBKP7时能搜索

到相对较优值，本文算法比 WGFA 更适合用于求解 BKP，全局搜索能力更强，能有效跳出早熟区域寻找到理论最优值，可以作为求解 BKP 的有效算法.

5 总　　结

本文在萤火虫算法基础上加入改进的小生境技术，在一定程度上增加了全局搜索能力，在算法初期利用混沌初始化得到的解均匀分布在解空间，并利用改进贪心算法增加了有效个体的数量，为后期寻找全局最优奠定了基础，基于萤火虫算法的改进算法 NCFA 可以有效求解 BKP.

参考文献：

[1] CONNOLLY D, MARTELLO S, TOTH P. Knapsack Problems: Algorithms and Computer Implementations [J]. The Journal of the Operational Research Society, 1991, 42(6): 513.

[2] 徐小平，庞润娟，王　峰，等. 求解 0-1 背包问题的烟花算法 [J]. 计算机系统应用，2019，28(2)：166-172.

[3] KELLERER H, PFERSCHY U, PISINGER D. The Bounded Knapsack Problem [M]//Knapsack Problems. Berlin: Springer, 2004: 185-209.

[4] 贺毅朝，王熙照，李文斌，等. 基于遗传算法求解折扣{0-1}背包问题的研究 [J]. 计算机学报，2016，39(12)：2614-2630.

[5] PISINGER D. A Minimal Algorithm for the Bounded Knapsack Problem [M]//Integer Programming and Combinatorial Optimization. Berlin: Springer, 1995: 95-109.

[6] 贺毅朝，李泽文，李焕哲，等. 离散灰狼优化算法求解有界背包问题 [J]. 计算机工程与设计，2019，40(4)：1008-1015.

[7] LI Y, HE Y C, LI H Z, et al. A Binary Particle Swarm Optimization for Solving the Bounded Knapsack Problem [M]// Communications in Computer and Information Science. Berlin: Springer, 2019: 50-60.

[8] 任静敏，潘大志. 带权重的贪心萤火虫算法求解 0-1 背包问题 [J]. 计算机与现代化，2019(5)：86-91.

[9] YANG X S. Firefly Algorithm, Stochastic Test Functions and Design Optimisation [J]. International Journal of Bio-Inspired Computation, 2010, 2(2): 78-84.

[10] THIÉTART R A, FORGUES B. Chaos Theory and Organization [J]. Organization Science, 1995, 6(1): 19-31.

[11] 桂传志. 混沌序列在优化理论中的应用 [D]. 南京：南京理工大学，2006.

[12] YANG X S, DEB S. Cuckoo Search via Levy Flights [C]//Proceeding of World Congress on Nature & Biologically Inspired Computing 2009. New York: IEEE Publications, 2009.

[13] YANG X S. Nature-Inspired Metaheuristic Algorithms [M]. 2nd ed. Frome: Luniver Press, 2010.

[14] LI M, CHEN T H. A Study of Genetic Algorithm Based on Niche Technique [J]. Advanced Materials Research, 2012, 1670: 152-155.

Chaotic Niche Firefly Algorithm Solving Bounded Knapsack Problem

REN Jing-min[1], PAN Da-zhi[1,2]

1. *College of Mathematics and Information, China West Normal University, Nanchong Sichuan 637009, China;*

2. *Institute of computing Method and Application Software, China West Normal University, Nanchong Sichuan 637009, China*

Abstract: To solve the Bounded Knapsack Problem, a chaotic niche firefly algorithm has been proposed. The chaotic theory has been used to initialize the firefly population. In order to increase the diversity of the population, the niche technology has been used to calculate the individual shared fitness. The similar individuals in the shared radius have been excluded by a certain probability, and the Levy flight operation has been performed on all the excluded individuals. Excellent individuals perform local searches, and the precocious individuals have re-randomly been generated to update their positions using chaos theory. The simulation results show that the improved algorithm can effectively solve the Bounded Knapsack Problem.

Key words: firefly algorithm; Bounded Knapsack Problem; Chaotic theory; niche technology; Levy Flights

责任编辑　张　枸