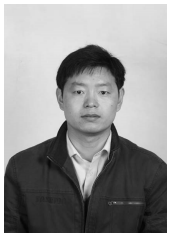
文章编号：1671-7848(2019)05-0991-06 **DOI:** 10.14107/j.cnki.kzgc. 161767

基于粒子群-模拟退火算法的背包问题研究

耿亚，吴访升

(常州工程职业技术学院 信息学院，江苏 常州 213164)



摘 要：针对组合优化中的经典背包问题，提出一种基于直觉模糊熵的粒子群-模拟退火算法(IFEPSO-SA)。该算法以种群的直觉模糊熵(IFE)为测度，建立一种基于熵值的自适应惯性权重和变异操作的策略，以维持种群的多样性；同时，对粒子群算法进化过程中的局部最优解，进行交换操作和模拟退火机制选择，得到更优的局部最优解和全局最优解，以增强算法的寻优能力。实验结果表明，所提算法具有很好的鲁棒性和寻优能力，能较好地求解 0-1 背包组合优化问题。

关键词：直觉模糊熵；模拟退火机制；粒子群算法；背包问题

中图分类号：TP301 **文献标识码**：A

Research on Knapsack Problem Based on the Hybrid Algorithm of Particle Swarm Optimization and Simulated Annealing

GENG Ya, WU Fang-sheng

(School of Information, Changzhou Vocational Institute of Engineering, Changzhou 213164, China)

Abstract: A hybrid algorithm of particle swarm optimization and simulated annealing based on intuitionistic fuzzy entropy (IFEPSO-SA) is proposed for solving the classical knapsack problems. In order to keep the diversity of population, an inertia weight of dynamic changes and mutation operation are built in the algorithm by using a metric based on intuitionistic fuzzy entropy (IFE) of the population. Furthermore, the partial best-solution of PSO is updated by using an exchange operation and a simulated annealing mechanism to get the better partial best-solution and global best-solution, and increase the searching ability. The experimental results show that the algorithm has better robustness and searching ability, and is useful of dealing with 0-1 knapsack problem.

Key words: Intuitionistic fuzzy entropy; simulated annealing mechanism; particle swarm optimization; knapsack problem

1 引 言

0-1背包问题是组合约束优化中经典的NP-hard问题[1,2]，一直以来都是学者研究的重点，传统算法求解此问题时，会随着问题规模的增加而算法所需时间成指数增长，而智能算法很好地解决了这一缺点，因而被广泛应用到求解背包问题中，如遗传算法[3]、粒子群算法[4]、蚁群算法[5]及其混合算法[6]等。

粒子群算法[7,8]是Kennedy等人提出的一种群体智能算法，它的工作原理是通过群体间的协同竞争作用不断进化搜索得到最优解，有着参数少、实现简单且收敛速度快的优点。而模拟退火算法(Simulated Annealing, SA)是20世纪80年代发展起来的一种智能算法，它采用Metropolis接受准则，并通过控制温度等参数来模拟固体退火过程，对于局部搜索算法的拓展有着一定的优势。因此，两种算法都被广泛的应用于各种优化问题，如文献[4]利用直觉模糊熵对种群描述的优势应用到离散粒子群算法中，一定程度上增强了算法的寻优能力，但在求解较大规模的背包问题时求解效率有所下降；文献[9]则通过结合模拟退火与粒子群算法各自的优势，增强了算法的局部搜索能力，能较好求解较大规模的组合优化问题。

为此，本文结合直觉模糊集对处理模糊信息的

收稿日期：2017-02-27；**修回日期**：2017-05-12

基金项目：江苏省自然科学基金项目（16KJB520047）；江苏省科技重点研发项目（BE2017067）；科技部重点研发专项子课题（2018YFB1703505）

作者简介：耿亚(1975−)，男，江苏徐州人，硕士，讲师，主要从事计算机技术等方面的教学与科研工作（本文通讯作者）；吴访升(1969−)，男，浙江缙云人，博士，教授，主要从事图形图像处理等方面的教学与科研工作。

优势以及模拟退火算法在局部寻优能力强的特点，与粒子群算法进行优势互补，从而提出一种基于直觉模糊熵的粒子群-模拟退火算法(IFEPSO-SA)。通过针对不同规模的0-1背包问题进行实验证明，该算法很好的避免了早熟收敛，增强了算法的寻优能力。

2 背包模型与直觉模糊熵

2.1 经典 0-1 背包数学模型

经典的0-1背包问题描述为：给定一个背包和 n 种物品，其中，背包的容量为 C，第 i 种物品的质量为 c_i，价值为 p_i，如何通过物品选择，使得装入背包中的物品总价值最大。其数学模型如下：

$$\begin{aligned} & f=\max \sum_{i=1}^{n} p_i x_i \\ & \text{s.t. } \sum_{i=1}^{n} c_i x_i \le C,\ i=1,2,\ldots,n \\ & x_i=0,1 \end{aligned} \tag{1}$$

2.2 直觉模糊熵

模糊熵[10]是在模糊集的基础上提出的用来描述事物的模糊状态，它度量了事件的模糊不确定性。而直觉模糊集在模糊集的基础上增加了非隶属度和犹豫度，能够更加细腻地描述事件的本质，因而在此基础上的直觉模糊熵[11,12]比模糊熵能更好的反映事物的模糊状态。因此，在算法的进化过程中引入直觉模糊熵，并通过其值的变化来控制种群的多样性和收敛程度，避免算法陷入局部收敛。

定义 1 对种群(pop)所有粒子的适应度值进行归一化操作，使每个粒子适应度值都属于区间[0,1]。而后将区间[0,1]进行 N（种群规模）等分，计算适应度值在 N 个子区间中粒子个数，对个数大于1的子区间单独作为一个集合 $P_i, i=1,2,\ldots,k$（k 为个数大于1的子区间个数），对个数为1的子区间整合为一个集合 P_{k+1}，因此可满足 $\forall i, j \in \{1,2,\cdots,k, k+1\}$，都有 $P_i \cap P_j=\varnothing, \bigcup_{i=1}^{k+1} P_i = pop$，根据以上条件可定义：

$$\mu_i^t=\frac{|P_i|}{N}, \pi_i^t=\frac{|P_{k+1}|}{N}, \nu_i^t=1-\mu_i^t-\pi_i^t \tag{2}$$

式中，$i=1,2,\ldots,k$，μ_i^t 为隶属度，表示第 t 代中所有粒子隶属于第 i 个子区间的程度，$\mu_i^t \in [0,1]$；π_i^t 为犹豫度，$\pi_i^t \in [0,1]$；ν_i^t 为所有粒子不属于第 i 个子区间的程度，$\nu_i^t \in [0,1]$。

归一化操作如下：

先求出群体所有粒子适应度值的最小值 $f_{\min}$ 和最大值 $f_{\max}$，对于 $\forall i \in N$，都有 $f^t(x_i)=\left(f(x_i)-f_i^{\min}\right)/\left(f_i^{\max}-f_i^{\min}\right)$ 处理，使每个粒子的适应度值归一化后都在区间[0,1]。

定义2 直觉模糊熵[12]：设 H^t 是种群的直觉模糊熵，其定义如下：

$$H^t=\frac{1}{k}\sum_{i=1}^{k}\frac{\min\left(\mu_i^t,\nu_i^t\right)+\pi_i^t}{\max\left(\mu_i^t,\nu_i^t\right)+\pi_i^t} \tag{3}$$

式中，$H^t \in [0,1]$，当所有粒子收敛于同一子区间时，$k=1$，$H^t=0$；当粒子均匀分散在各子区间，即 $k=N$ 时，$H^t=1$。而种群中粒子、在子区间中分散的越均匀，H^t 的值越大，反之，则越小。

3 基于 IFE 的粒子群−模拟退火算法

3.1 基于 IFE 的惯性权重变化

在粒子群优化算法中，惯性权重(ω^t)的大小表示自身历史信息对现有粒子的影响程度，其值越大则能提高粒子的全局搜索性能，越小则有助于粒子进行局部搜索。在粒子群算法求解组合优化问题时，由于问题具有离散性，进化的方向难以控制，因此，采用基于直觉模糊熵的自适应惯性权重，以便有效地搜索到全局最优解，ω 的变化公式如下：

$$\omega^t=\omega_{\max}-(\omega_{\max}-\omega_{\min})\times\left(1-H^t\right) \tag{4}$$

式中，$\omega_{\max}$，$\omega_{\min}$ 分别为惯性权重最大值和最小值，H^t 为当前种群的熵值大小。

由于 $H^t \in [0,1]$，使得 ω 在 $\omega_{\max}$ 和 $\omega_{\min}$ 之间变化，当 H^t 接近0时，有助于算法进行局部搜索，当 H^t 接近1时，有助于算法进行全局搜索。本文 $\omega_{\max}=0.9$，$\omega_{\min}=0.4$。

3.2 粒子群算法更新公式

粒子群算法多用于在连续的空间中求解问题，而 0-1 背包是组合优化问题，具有离散性质，因而修改 PSO 算法更新公式如下：

设第 i 个粒子在 D 维空间中的表示为 $x_i=(x_{i1},x_{i2},\ldots,x_{iD})$，此粒子经历的历史最优位置记为 $p_i=(p_{i1},p_{i2},\ldots,p_{iD})$，即局部最优解 p_{best}。在群体中所有粒子经历过的最优位置记为全局最优解 g_{best}。粒子 i 的速度表示为 $v_i=(v_{i1},v_{i2},\ldots,v_{iD})$。第 $t+1$ 次迭代时，其第 i 个粒子的第 d 维 $(1\le d\le D)$ 的速度和位置更新公式如下：

$$\begin{aligned} v_{id}^{t+1}=&\omega^t\times v_{id}^t+c_1\times r_1\times\left(p_{best}-x_{id}^t\right)+\\ &c_2\times r_2\times\left(g_{best}-x_{id}^t\right) \end{aligned} \tag{5}$$

$$v_{id}^{t+1}=\begin{cases} -v_{\max}, & v_{id}^{t+1}<-v_{\max} \\ v_{\max}, & v_{id}^{t+1}>v_{\max} \end{cases} \tag{6}$$

$$x_{id}^{t+1}=x_{id}^t+v_{id}^{t+1} \tag{7}$$

$$x_{id}^{t+1}=\begin{cases}1, & x_{id}^{t+1}\geq 0.5\\ 0, & x_{id}^{t+1}<0.5\end{cases} \tag{8}$$

式(5)中 c_1,c_2 为学习因子，用来平衡个体和群体认知能力，r_1,r_2 是分布在 $[0,1]$ 上相互独立的随机数。式(6)则是为了缓解粒子的进化过程，将粒子的速度控制在区间 $[-v_{\max},v_{\max}]$ 中；式(7)为位置更新公式；式(8)则是将 x_{id}^{t+1} 转化为 0 或 1 的整数值。本文参数设置 $v_{\max}=1$，$c_1=0.8$，$c_2=0.8$。

3.3 模拟退火操作

模拟退火算法(SA)基本思想是在给定解和其局部领域随机产生的新解之间，通过 Metropolis 接受准则使适应度较好的解被接受或者使适应度较差的解以一定的概率被接受。在粒子群算法的过程中，由于算法每次迭代所产生的局部最优解不一定都满足问题中的约束条件，因此，引入模拟退火机制对局部最优解进行优化搜索，以期得到更多满足约束条件且适应值更好的局部最优解，进而产生更优的全局最优解，以引导种群的进化。

同时，为了不使模拟退火中局部领域产生的新解与给定解相比变化过大而引起进化过程失控，本文采用了一种交换机制来产生新解，即在粒子群算法所产生的局部最优解 p_{best} 中，随机选择一个解，再随机选择此解的两位置进行交换得到新解 $xbest$，如假设粒子维数为 $D=6$，所选旧解为 $p_{best}=101101$，随机选取得到位置 2 和 4，通过交换机制得到新解 $xbest=111001$。

其模拟退火操作过程伪代码如下：

Begin

for $k=1:L$

从 *size*（种群大小）个局部最优解中随机选择一个解 $pbest_i$；

通过交换机制得到新解 $xbest_i$；

计算出新解 $xbest_i$ 的适应度值 $f'(x_i)$ 以及质量大小 cx_i；

if $cx_i\leq C$ *Then*

计算出 $pbest_i$ 的 $f(x_i)$ 与 $xbest_i$ 的 $f'(x_i)$ 之间的差值，记为 ΔE；

if $\Delta E\leq 0$ *Then*

将新解 $xbest_i$ 与 $f'(x_i)$ 赋值给 $pbest_i$ 和 $f(x_i)$；

elseif $\exp(-\Delta E/T)>rand$

将新解 $xbest_i$ 与 $f'(x_i)$ 赋值给 $pbest_i$ 和 $f(x_i)$；

end

end

通过公式 $T=alpha\times T$ 降低温度，一轮退火操作结束。

end

式中，cx_i 为第 i 个解的背包质量，C 为背包的容量值。

随着退火机制的进行和温度的降低以及对局部最优解的交换操作，使非可行解被接受的概率逐渐减小，同时增加了满足问题中背包质量约束条件的局部最优解的数量，有利于产生更优的全局最优解 g_{best}，进而使种群在下次迭代时朝着满足约束条件的方向进化。本文参数设置 $L=10\times D$，$T_0=500$，$alpha=0.95$。

3.4 基于 IFE 的变异操作

由于粒子群算法随着迭代次数的增加易陷入局部最优，而熵值 H^t 的大小表明当前种群的收敛程度，因而通过种群熵值 H^t 对种群进行变异操作，可以一定程度上增加种群的多样性，减少陷入局部最优的可能。其操作如下：

当 $H^t<\alpha$ 时，开始进行变异操作，而后对种群中的每个粒子，根据 *rand* 是否小于变异概率 $P=(1-H^t)$ 判断，若 $rand<P$，则对该粒子进行变异，即随机生成满足维数大小的粒子更换此粒子；否则，不对该粒子进行变异操作。本文参数设置 $\alpha=0.5$。

3.5 IFEPSO-SA 算法步骤

IFEPSO-SA 算法的详细步骤如下：

Step 1 设置算法参数，其中种群规模为 *size*，迭代次数为 *MaxIt*，当前迭代次数 $t=1$，初始熵值 $H^t=1$。

Step 2 随机产生规模为 *size* 大小的种群，依次计算出种群中每个粒子的适应度值 $f(x_i)$ 和质量大小 $c(x_i)$，进而产生第一次迭代的全局最优解 g_{best} 和局部最优解 p_{best}。

Step 3 进入迭代循环，通过公式(4)-(7)更新每个粒子的速度和位置，同时按粒子的历史最佳位置求出局部最优解 p_{best}。

Step 4 通过 3.3 节对当前的局部最优解 p_{best} 进行模拟退火操作和交换操作，得到更优的局部最优解 p_{best}，而后求出全局最优解 g_{best}。

Step 5 通过式(2)、(3)求出种群的熵值 H^t，而后按照熵值 H^t 大小对种群进行 3.4 节的变异操作。

Step 6 $t=t+1$，当 $t>MaxIt$ 时，迭代循环结束，输出每次迭代时的全局最优解和其适应度值。

否则，返回步骤 Step 3。

4 测试结果与分析

4.1 测试数据

为了测试算法 IFEPSO-SA 在处理不同规模的 0-1 背包问题上的寻优性能，引入文献[13]中的数据，选择规模从 10-100 的 9 个经典 0-1 背包问题数据(f_1–f_9)来对算法进行测试，测试数据，见表 1。

算法对表 1 中的每一组数据都独立运行 20 次所得到的最优解状况，见表 2。

表 1 测试数据

Tab. 1 Test data

编号	维数	数据(c, p, V)	最优解
1	10	c=[95, 4, 60, 32, 23, 72, 80, 62, 65, 46]; p=[55, 10, 47, 5, 4, 50, 8, 61, 85, 87]; V=269;	295/269
2	15	c=[56.358 531, 80.874 050, 47.987 304, 89.596 240, 74.660 48, 85.894 345, 51.353 496, 1.498 459, 36.445 204, 16.589 862, 44.569 23, 0.466 9, 37.788 018, 57.118 442, 60.716 575]; p=[0.125 126, 19.330 424, 58.500 931, 35.029 145, 82.284 005, 17.410 810, 71.050 142, 30.399 487, 9.140 294, 14.731 285, 98.852 504, 11.908 322, 0.891 140, 53.166 295, 60.176 397]; V=375;	481.07/ 354.96
3	20	c=[92, 4, , 43, 83, 84, 68, 92, 82, 6, 44, 32, 18, 56, 83, 25, 96, 70, 48, 14, 58]; p=[44, 46, 90, 72, 91, 40, 75, 35, 8, 54, 78, 40, 77, 15, 61, 17, 75, 29, 75, 63]; V=878;	1 024/ 871
4	23	p=[981, 980, 979, 978, 977, 976, 487, 974, 970, 485, 485, 970, 970, 484, 484, 976, 974, 482, 962, 961, 959, 958, 857]; c=[983, 982, 981, 980, 979, 978, 488, 976, 972, 486, 486, 972, 972, 485, 485, 969, 966, 483, 964, 963, 961, 958, 959]; V=10 000;	9 767/ 9 768
5	50	p=[220, 208, 198, 192, 180, 180, 165, 162, 160, 158, 155, 130, 125, 122, 120, 118, 115, 110, 105, 101, 100, 100, 98, 96, 95, 90, 88, 82, 80, 77, 75, 73, 72, 70, 69, 66, 65, 63, 60, 58, 56, 50, 30, 20, 15, 10, 8, 5, 3, 1,]; c=[80, 82, 85, 70, 72, 70, 66, 50, 55, 25, 50, 55, 40, 48, 50, 32, 22, 60, 30, 32, 40, 38, 35, 32, 25, 28, 30, 22, 50, 30, 45, 30, 60, 50, 20, 65, 20, 25, 30, 10, 20, 25, 15, 10, 10, 10, 4, 4, 2, 1]; V=1 000;	3103/ 1 000
6	50	p=[72, 490, 651, 833, 883, 489, 359, 337, 267, 441, 70, 934, 467, 661, 220, 329, 440, 774, 595, 98, 424, 37, 807, 320, 501, 309, 834, 851, 34, 459, 111, 253, 159, 858, 793, 145, 651, 856, 400, 285, 405, 95, 391, 19, 96, 273, 152, 473, 448, 231]; c=[438, 754, 699, 587, 789, 912, 819, 347, 511, 287, 541, 784, 676, 198, 572, 914, 988, 4, 355, 569, 144, 272, 531, 556, 741, 489, 321, 84, 194, 483, 205, 607, 399, 747, 118, 651, 806, 9, 607, 121, 370, 999, 494, 743, 967, 718, 397, 589, 193, 369]; V=11 258;	16 102/ 11 231
7	60	c=[135, 133, 130, 11, 128, 123, 20, 75, 9, 66, 105, 43, 18, 5, 37, 90, 22, 85, 9, 80, 70, 17, 60, 35, 57, 35, 61, 40, 8, 50, 32, 40, 72, 35, 100, 2, 7, 19, 28, 10, 22, 27, 30, 88, 91, 47, 68, 108, 10, 12, 43, 11, 20, 37, 17, 4, 3, 21, 10, 67]; p=[350, 310, 300, 295, 290, 287, 283, 280, 272, 270, 265, 251, 230, 220, 215, 212, 207, 203, 202, 200, 198, 196, 190, 182, 181, 175, 160, 155, 154, 140, 132, 125, 110, 105, 101, 92, 83, 77, 75, 73, 72, 70, 69, 66, 60, 58, 45, 40, 38, 36, 33, 31, 27, 23, 20, 19, 10, 9, 4, 1]; V=2 400;	8 362/ 2 393
8	80	c=[40, 27, 5, 21, 51, 16, 42, 18, 52, 28, 57, 34, 44, 43, 52, 55, 53, 42, 47, 56, 57, 44, 16, 2, 12, 9, 40, 23, 56, 3, 39, 16, 54, 36, 52, 5, 53, 48, 23, 47, 41, 49, 22, 42, 10, 16, 53, 58, 40, 1, 43, 56, 40, 32, 44, 35, 37, 45, 52, 56, 40, 2, 23, 49, 50, 26, 11, 35, 32, 34, 58, 6, 52, 26, 31, 23, 4, 52, 53, 19]; p=[199, 194, 193, 191, 189, 178, 174, 169, 164, 164, 161, 158, 157, 154, 152, 152, 149, 142, 131, 125, 124, 124, 124, 122, 119, 116, 114, 113, 111, 110, 109, 100, 97, 94, 91, 82, 82, 81, 80, 80, 80, 79, 77, 76, 74, 72, 71, 70, 69, 68, 65, 65, 61, 56, 55, 54, 53, 47, 47, 46, 41, 36, 34, 32, 32, 30, 29, 29, 26, 25, 23, 22, 20, 11, 10, 9, 5, 4, 3, 1]; V=1 173;	5 183/ 1 170
9	100	c=[54, 95, 36, 18, 4, 71, 83, 16, 27, 84, 88, 45, 94, 64, 14, 80, 4, 23, 75, 36, 90, 20, 77, 32, 58, 6, 14, 86, 84, 59, 71, 21, 30, 22, 96, 49, 81, 48, 37, 28, 6, 84, 19, 55, 88, 38, 51, 52, 79, 55, 70, 53, 64, 99, 61, 86, 1, 64, 32, 60, 42, 45, 34, 22, 49, 37, 33, 1, 78, 43, 85, 24, 96, 32, 99, 57, 23, 8, 10, 74, 59, 89, 95, 40, 46, 65, 6, 89, 84, 83, 6, 19, 45, 59, 26, 13, 8, 26, 5, 9]; p=[297, 295, 293, 292, 291, 289, 284, 284, 283, 283, 281, 280, 279, 277, 276, 275, 273, 264, 260, 257, 250, 236, 236, 235, 235, 233, 232, 232, 228, 218, 217, 214, 211, 208, 205, 204, 203, 201, 196, 194, 193, 193, 192, 191, 190, 187, 187, 184, 184, 184, 181, 179, 176, 173, 172, 171, 160, 128, 123, 114, 113, 107, 105, 101, 100, 100, 99, 98, 97, 94, 94, 93, 91, 80, 74, 73, 72, 63, 63, 62, 61, 60, 56, 53, 52, 50, 48, 46, 40, 40, 35, 28, 22, 22, 18, 15, 12, 11, 6, 5]; V=3 820;	15 170/ 3 818

表 2 IFEPSOSA 算法测试结果及其对比

Tab. 2 IFEPSOSA algorithm test results and their comparison

f	维数	最好	最差	平均值	命中次数	文献[14]-[20]提供的最优解
1	10	295/269	295/269	295	20	294/260_遗传算法[14]，295/269_模糊粒子群算法[14]
2	15	481.07/354.96	481.07/354.96	481.07	20	481.07/354.96_自适应和声搜索算法[15]
3	20	1 024/871	1 024/871	1024	20	889/865_遗传算法[14]，1 024/871_模糊粒子群算法[14]
4	23	9 767/9 768	9 767/9 768	9767	20	9 757/9 777_降维替换算法[16]
5	50	3 103/1 000	3 085/1 000	3 090.03	5	3 090/-_基于模拟退火的遗传算法[17]，3 082/-_模拟退火算法[17]
6	50	16 102/11 231	15 986/11 242	16 073	11	15 565/-_二进制粒子群算法[18]，16 052/-_二进制混合粒子群算法[18]
7	60	8 362/2 393	8 356/2 395	8 361.4	18	8 362/-_引入侦查子群的蚁群算法[19]， 7 775/2 371_基于贪心策略改进的遗传算法[19]
8	80	5 183/1 170	5 167/1 171	5 178.9	8	5 107/1 172_基于罚函数的离散粒子群算法[20]， 5 107/1 167_混合粒子群算法[20]， 5 183/1 170_基于贪心变换策略的离散微粒群算法[20]
9	100	15 170/3 818	15 141/3 807	15 162	10	15 089/3 817_基于罚函数的离散粒子群算法[20]，15 080/3 819_混合离散粒子群算法[20]，15 170/3 818_基于贪心变换策略的离散微粒群算法[20]

4.2 测试结果与分析

算法的运行环境为：Intel(R) Core(TM) i7-4790 CPU 处理器和 8 G 内存的计算机以及 Matlab 2014 的测试软件。本文种群规模 *size* 和迭代次数 *MaxIt*

设置为 100/200，IFEPSO-SA 算法在求解问题时的直觉模糊熵值变化图，如图 1 所示。

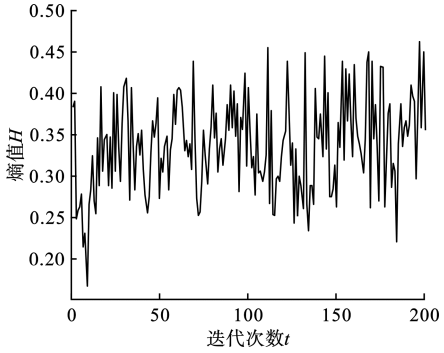


图 1 测试 f_9 时的直觉模糊熵值变化图

Fig. 1 Change graph based on intuitionistic fuzzy entropy in testing f_9

图 1 熵值变化反映了种群的收敛情况，由图可知，熵值基本维持在[0, 0.5]的区间内，这说明算法中的模拟退火操作有效地提高了算法的局部探索能力，使种群保持着一定的局部收敛程度；同时，随着迭代次数的增加，熵值在上下波动，说明算法的变异操作很好的维持了种群的多样性，使种群具有很好的全局搜索能力，并逐渐向着全局最优解的方向收敛。

而从表 2 中可知，IFEPSO-SA 算法在求解维数比较小的 0-1 背包问题时能够有效地获得最优解，如对于 f_1-f_4 4 个维数较小的 0-1 背包问题，算法都能准确找到最优解，而在求解维数比较大的背包问题(f_5-f_9)时能以一定的概率获得最优解，如本文算法在求解 f_6 时，20 次测试中有 11 次得到给定的最优解，同时本文算法获得的最差解也依然优于许多文献中所提部分算法的最优解，如求解 f_7 时，本文算法所得的最差解 8 356/2 395 优于文献[19]中基于贪心策略改进的遗传算法所得最优解 7 775/2 371，f_5、f_8 以及 f_9 亦是如此，说明算法可以很好地解决背包问题。而且本文算法所得的最差值和最好值之间相差不大，说明算法具有很好的鲁棒性和寻优能力。

4.3 算法比较分析

为了进一步测试 IFEPSO-SA 算法在不同规模的 0-1 背包问题下的求解效果，本文采用粒子群算法(PSO)和模拟退火(SA)算法与本文算法在相同环境下进行比较分析，其中，PSO 算法中 $size=100$，$MaxIt=200$，其余部分沿用本文 PSO 部分中的参数设置；SA 算法中截止温度设置为 0.000 0001，其余参数设置与本文 SA 部分中的参数设置一样；采用的测试实例为：选择 n 种物品，其质量 c_i 和价值 p_i ($i=1,2,\ldots,n$)在[1,100]区间内随机生成，背包容量为所有物品总质量的 0.8 倍，即 $C=0.8\sum_{i=1}^{n}c_i$。在维数 n 为 50, 100, 200 下 3 种算法的最优值变化曲线图，如图 2 所示。

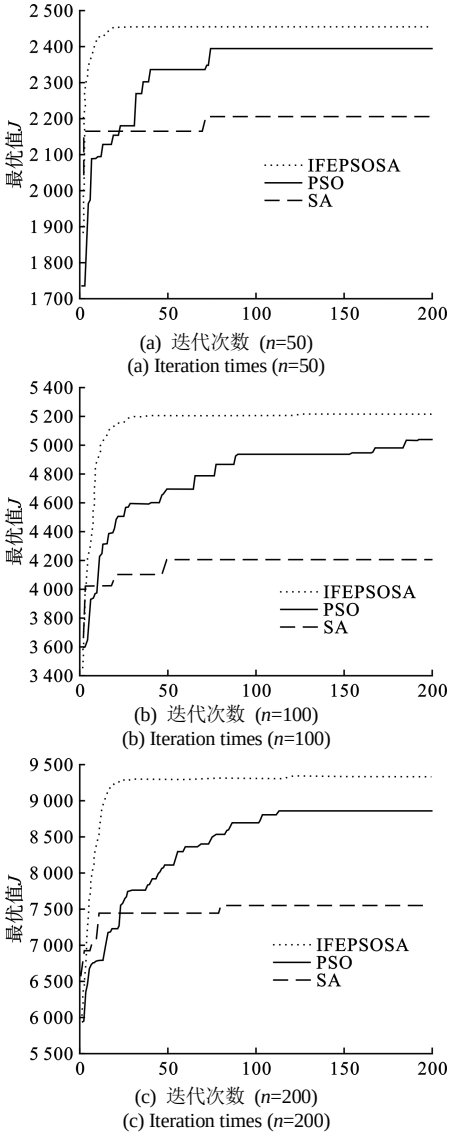


图 2 三种算法的最优值变化图

Fig. 2 Optimal value change diagrams of three algorithms

从图 2 可知，不管维数是 50，还是 100，抑或是 200，IFEPSO-SA 算法所得最优解一直高于 PSO 算法和 SA 算法，且当迭代次数小于 50 时，IFEPSO-SA 算法就能取得非常接近最优解的次优解，而 PSO 算法和 SA 算法却在迭代次数超过 100 后才搜索到次优解，说明 IFEPSO-SA 算法通过模拟退火操作能找到更好的局部最优解和全局最优解，进而使种群在粒子群算法的引导下快速的进化，而 PSO 算法只能通过自身的每次迭代产生局部最优解，因而进化速度比较慢。

同时，随着迭代次数的增加，IFEPSO-SA 算法的最优值曲线一直在向最优解的方向递进，即使在迭代次数接近 200 时，曲线也有所变化，而 PSO 算法和 SA 算法所得曲线在前期快速增加，在后期则

一直维持不变，说明 PSO 算法和 SA 算法陷入局部收敛，而 IFEPSO-SA 算法则通过基于种群熵值的变异操作有效的避免了这种情况，使算法很好的朝着最优解的方向进化。

5 结　论

本文提出了一种基于直觉模糊熵的粒子群-模拟退火算法来求解经典的 0-1 背包问题，算法以种群的直觉模糊熵描述当前的种群状态，进而通过基于熵值的自适应惯性权重变化和变异操作，控制种群的收敛程度，从而增加种群的多样性；同时，通过交换操作和模拟退火机制，产生更好的局部最优解和全局最优解，使算法有更强的寻优能力。在大量实验的基础上，相对于现有许多研究背包问题的算法以及 PSO 算法和 SA 算法而言，IFEPSO-SA 算法有更优的性能。虽然在解决大规模的背包问题上，IFEPSO-SA 算法只能以一定的概率找到最优解，求解效率有所下降，但即使是找到的次优解与给定最优解之间相差不大，而且算法能有效的求解小规模的背包问题。

参考文献(References)

[1] Fayard D, Plateau G. Resolution of the 0-1 knapsack problem comparison of methods[J]. Mathematical Programming, 1975, 8(1): 272-307.

[2] 李枝勇, 马良, 张惠珍. 求解多维背包问题的改进布谷鸟搜索算法[J]. 控制工程, 2016, 23(7): 1069-1075.
Li Z Y, Ma L, Zhang H Z. Modified Cuckoo Search Algorithm for Solving Multi-dimensional Knapsack Problem[J]. Control Engineering of China, 2016, 23(7): 1069-1075.

[3] 贺毅超, 宋建明, 张敬敏, 等. 利用遗传算法求解静态与动态背包问题的研究[J]. 计算机应用研究, 2015, 32(4): 1011-1015.
He Y C, Song J M, Zhang J M, et al. Research on genetic algorithms for solving static and dynamic knapsack problems[J]. Application Research of Computers, 2015, 32(4): 1011-1015.

[4] 包广清, 毛开富. 改进粒子群算法及其在风电系统中的应用[J]. 控制工程, 2013, 20(2): 262-266.
Bao G Q, Mao K F. An improved PSO algorithm and its utilization in wind power generation system[J]. Control Engineering of China, 2013, 20(2): 262-266.

[5] 廖灿星, 李行善, 张平, 等. 一种求解背包问题的正态分布蚁群算法[J]. 系统仿真学报, 2011, 23(6): 1156-1160.
Liao C X, Li X S, Zhang P, et al. Improved Ant Colony Algorithm Base on Normal Distribution for Knapsack Problem[J]. Journal of System Simulation, 2011, 23(6): 1156-1160.

[6] 於世为, 魏一鸣, 诸克军. 基于粒子群-遗传的混合优化算法[J]. 系统工程与电子技术, 2011, 33(7): 1647-1653.
Yu S W, Wei Y M, Zhu K J. Hybrid optimization algorithms based on particle swarm optimization and genetic algorithm[J]. Systems Engineering and Electronics, 2011, 33(7): 1647-1653.

[7] Kennedy J, Eberhart R C. Particle swarm optimization[C]. Proceedings of IEEE International Conference on Neural Networks. 1995: 1942- 1948.

[8] Kennedy J, Eberhart R C. A discrete binary version of the particle swarm optimization[A]. Proceeding of the conference on System, Man, and Cybernetics[C]. NJ, USA: IEEE Service Center, 1997, 4104-4109.

[9] 丁铸, 马大为, 汤铭端, 等. 基于禁忌退火粒子群算法的火力分配[J]. 系统仿真学报. 2006, 18(9): 2480-2483.
Ding Z, Ma D W, Tang M D, et al. A hybrid search algorithm of Tabu Search and annealing particle swarm optimization for weapon-target assignment[J]. Journal of System Simulation, 2006, 18(9): 2480-2483.

[10] Zadeh L A. Fuzzy sets[J]. Information and Control, 1965, 8(3): 338-356.

[11] Szmidt E, Kacprzyk J. Entropy for intuitionistic fuzzy sets[J]. Fuzzy Sets and Systems, 2001, 118(3): 467-477.

[12] 王毅, 雷英杰. 一种直觉模糊熵的构造方法[J]. 控制与决策, 2007, 12(22): 1390-1394.
Wang Y, Lei Y J. A technique for constructing intuitionistic fuzzy entropy[J]. Control and Decision, 2007, 22(12): 1390-1394.

[13] 吴虎胜, 张凤鸣, 战仁军, 等. 求解 0-1 背包问题的二进制狼群算法[J]. 系统工程与电子技术, 2014, 36(8): 1660-1668.
Wu H S, Zhang F M, Zhan R J, et al. A binary wolf pack algorithm for solving 0-1 knapsack problem[J]. Systems Engineering and Electronics, 2014, 36(8): 1660-1668.

[14] 柳寅, 马良. 0-1 背包问题的模糊粒子群算法求解[J]. 计算机应用研究, 2011, 28(11): 4026-4027&4031.
Liu Y, Ma L. Solving 0-1 knapsack problem by fuzzy particle swarm optimization[J]. Application Research of Computers, 2011, 28(11): 4026-4027&4031.

[15] Zhang X G, Huang S Y, Hu Y, et al. Solving 0-1 knapsack problems based on amoeboid organism algorithm[J]. Applied Mathematics and Computation, 2013, 219(19): 9959-9970.

[16] 高天, 王梦光, 唐立新, 等. 特殊一维背包问题的降维替换算法研究[J]. 系统工程理论方法与应用, 2002, 11(2): 125-130.
Gao T, Wang M G, Tang L X, et al. The research for the reductive dimension and replacive variable algorithm of special restrict 0-1 ILP[J]. Systems Engineering-Theory Methodolog Applications, 2002, 11(2): 125-130.

[17] 张盛意, 蔡之华, 占志刚. 基于改进模拟退火的遗传算法求解 0-1 背包问题[J]. 微电子学与计算机, 2011, 28(2): 61-64.
Zhang S Y, Cai Z H, Zhan Z G. Solving 0-1 knapsack problem based on genetic algorithm with improved simulated annealing[J]. Microelectronics & Computer, 2011, 28(2): 61-64

[18] 罗健文. 基于交叉操作的二进制混合粒子群算法求解背包问题[J]. 中南林业科技大学学报, 2011, 31(9): 170-174.
Luo J W. Binary hybrid particle swarm optimization algorithm base on crossover operation for solving knapsack problem[J]. Journal of Central South University of Forestry & Technology, 2011, 31(9): 170-174.

[19] 胡中华, 赵敏. 引入侦查子群的蚁群算法求解 0/1 背包问题[J]. 贵州师范大学学报(自然科学版), 2009, 27(3): 82-88.
Hu Z H, Zhao M. Using ant Colony optimization algorithm with scout subgroup to solve 0/1 knapsack problem[J]. Journal of GuiZhou Normal University (Natural Sciences), 2009, 27(3): 82-88.

[20] 刘建芹, 贺毅朝, 顾茜茜. 基于离散微粒群算法求解背包问题研究[J]. 计算机工程与设计, 2007, 28(13): 3189-3191.
Liu J Q, He Y C, Gu Q Q. Solving knapsack problem based on discrete particle swarm optimization[J]. Computer Engineering and Design, 2007, 28(13): 3189- 3191.