计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(5):133−138 [doi: 10.15888/j.cnki.csa.006340]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于猴群算法求解 0-1 背包问题①

徐小平[1], 师喜婷[1], 钱富才[2]

[1](西安理工大学 理学院, 西安 710054)
[2](西安理工大学 自动化与信息工程学院, 西安 710048)
通讯作者: 师喜婷, E-mail: 1067928905@qq.com

摘　要: 0-1 背包问题是一个经典的 NP 完全问题, 该问题在实际生活中具有广泛的应用. 针对现有算法在求解 0-1 背包问题时精度不高的缺点, 提出了一种诱导因子猴群算法. 所给诱导因子猴群算法的基本思想是, 在基本猴群算法的爬过程中引入诱导因子, 诱导其向上爬行, 从而可以逃逸局部最优解, 找到全局最优解. 在仿真试验中, 与已有方法进行比较, 结果说明利用所给诱导因子猴群算法求解 0-1 背包问题是有效的.

关键词: 0-1 背包问题; 组合优化; 群智能; 诱导因子; 猴群算法

引用格式: 徐小平,师喜婷,钱富才.基于猴群算法求解 0-1 背包问题.计算机系统应用,2018,27(5):133–138. http://www.c-s-a.org.cn/1003-3254/6340.html

Solving 0-1 Knapsack Problem Based on Monkey Algorithm

XU Xiao-Ping[1], SHI Xi-Ting[1], QIAN Fu-Cai[2]

[1](School of Sciences, Xi'an University of Technology, Xi'an 710054, China)
[2](School of Automation and Information Engineering, Xi'an University of Technology, Xi'an 710048, China)

Abstract: The 0-1 knapsack problem is a classical NP complete problem, which has a wide range of applications in real life. In view of the existing algorithms with the shortcoming of low precision in solving 0-1 knapsack problem, an inducing factor monkey algorithm is proposed to solve the problem. The basic idea of the proposed inducing factor monkey algorithm is that an inducing factor is adopted in the climbing process of the basic monkey algorithm to induce it to crawl upward, which can escape from local optimal solution and find the global optimal solution. In the simulation, compared with the existing methods, the results show that the proposed inducing factor monkey algorithm for solving 0-1 knapsack problem is valid.

Key words: 0-1 knapsack problem; combinatorial optimization; swarm intelligence; inducing factor; monkey algorithm

背包问题是由 Merkel 和 Hellman 在 1978 年提出的[1], 它是一个典型的组合优化问题, 属于 NP-hard 难题, 随着问题规模越来越大, 复杂度增强. 在实际应用中, 许多工业和金融等问题都可以用背包问题来描述, 如预算控制、资源分配、项目选择、资本投资、金融组合、材料切割和物件装箱等[2]. 近年来, 背包问题已成为众多学者研究的一个热点问题, 寻找新的方法来求解背包问题具有重要的理论和实际意义[3]. 目前, 求解背包问题的方法主要有两种: 最优算法和启发式算法. 最优算法包括穷举法、动态规划法、递归算法、回溯法和分支界限法等[4–7]. 启发式算法包括差分进化算法[8], 粒子群算法[9]和遗传算法[10]等. 前者的思想比较简单, 不需要其它专业知识, 一般适用于求解规模较小的问题; 后者一般适用于求解规模较大的问题, 可是对其求解的精度还需继续探讨.

猴群算法是一种新兴的群智能优化算法[11], 该算

① 基金项目: 国家自然科学基金(61773016); 陕西省自然科学基础研究计划项目(2014JM8325); 陕西省教育厅专项科研计划项目(14JK1538); 西安理工大学科技创新计划项目 (2016CX013)
收稿时间: 2017-08-26; 修改时间: 2017-09-15; 采用时间: 2017-09-25; csa 在线出版时间: 2018-04-23

法的突出特点在于求解高维的优化问题时, 无需考虑函数是否可导或可微, 只需要计算当前位置的伪梯度, 就可以确定算法的搜索方向, 并且该算法的结构简单, 参数少和易实现. 因此, 猴群算法在提出不久便得到了众多学者的广泛关注, 在各个领域内得到了快速地发展, 例如在大跨径连续刚构桥传感器优化布置问题[12]、入侵检测系统存在高漏报率的问题[13]、加气站项目进度的问题[14]、混合动力优化[15,16]、模糊规则的分类器[17]等领域. 而文献[12–17]中均是基本猴群算法的应用, 可以看出在某种程度上解决了一些问题. 但是, 在上述问题的应用中, 随着测点数目或者规则等的增加, 基本猴群算法暴露出易陷入局部最优, 导致存在求解精度不高的弊端. 因此, 对该算法进一步研究是很必要的, 并将其的应用进行扩展, 来解决实际生活中的诸多应用问题具有重要的实际意义.

为了提高猴群算法的寻优性能, 本文将诱导因子引入基本猴群算法的爬过程中, 给出了一种诱导因子猴群算法, 并将其应用到求解 0-1 背包问题. 最后, 通过仿真实验验证了该方法的可行性.

1 0-1 背包问题

0-1 背包问题是一种常见的组合优化问题. 一般可简单叙述为: 设物品的数量为 D, 第 i 个物品的体积(重量) 和价格分别为 w_i 和 p_i, 一个背包的能承受的最大容量为 V, 选择一些物品, 放入背包中. 如何选择物品, 在满足背包约束条件下, 使背包中物品的总价值最大, 这个问题被称为 0-1 背包问题, 其数学模型建立如下:

$$\max\ f = RX = \sum_{i=1}^{D} R_i x_i \tag{1}$$

$$\text{s.t.}\quad WX = \sum_{i=1}^{D} w_i x_i \le V \tag{2}$$

其中, $R=(R_1,R_2,\cdots,R_D)$ 表示物品的价值向量, $W=(w_1,w_2,\cdots,w_D)$表示物品对应的体积 (重量) 向量, $X=(x_1,x_2,\cdots,x_n)$表示解向量. 式 (1) 表示目标函数; 式 (2) 中 x_i 为决策变量, x_i=0 表示第 i 个物品未装入包中, x_i=1 表示第 i 个物品装入包中.

2 猴群算法

2.1 基本猴群算法

基本猴群算法是由 Zhao 和 Tang 两位学者首次在期刊 Journal of Uncertain Systems 上提出的[11]. 该算法主要包括解的表示和初始化、爬过程、望过程和跳过程, 其具体的操作过程如下:

Step1. 解的表示和初始化: 设 $X_i=(x_{i1},x_{i2},\cdots,x_{ij},\cdots,x_{iD})(1\le i\le M,1\le j\le D)$ 是目标函数的一个可行解, 表示第 i 只猴子当前的位置, 由式 (3) 生成.

$$X_i=(x_{\max}-x_{\min})\cdot rand(1,D)+x_{\min} \tag{3}$$

其中, M 表示种群规模, D 为维数, x_{ij} 表示第 i 只猴子在第 j 维的位置, 变量的区间为 $[x_{\min},x_{\max}]$, $rand(1,D)$是由[0, 1]之间的均匀分布的随机数组成的 D 维向量.

Step2. 执行爬过程: 爬过程是每只猴子在当前的范围内通过逐步爬行迭代, 寻找优化问题的目标函数值的过程, 具体过程如下:

1) 随机生成向量 $\Delta X_i=(\Delta x_{i1},\Delta x_{i2},\cdots,\Delta x_{ij},\cdots,\Delta x_{iD})$, 其中, Δx_{ij}由式 (4) 生成. 参数$a(a>0)$为猴群每次爬的步长.

$$\Delta x_{ij}=\begin{cases} a, & \text{以概率0.5}\\ -a, & \text{以概率0.5}\end{cases} \tag{4}$$

2) 计算 $f_{ij}'(X_i)=\dfrac{f(X_i+\Delta X_i)-f(X_i-\Delta X_i)}{2\Delta X_i}$, 其中, $f_i'(X_i)=(f_{i1}'(X_i),f_{i2}'(X_i),\cdots,f_{iD}'(X_i))$是目标函数所在位置的伪梯度.

3) 令$Y_i=X_i+a\cdot sign(f_{ij}'(X_i))$, 其中, $sign$ 为符号函数.

4) 如果向量$Y_i=(y_{i1},y_{i2},\cdots,y_{ij},\cdots,y_{iD})$在$[x_{\min},x_{\max}]$范围内, 并且 $f(Y_i)<f(X_i)$, 更新 X_i 为 Y_i: 否则, X_i 不变.

5) 重复 1)–4), 直到达到预定的执行次数 Nc.

Step3. 执行望过程: 在爬过程之后, 每只猴子都到达了各自所在的山峰的顶端, 并向四周眺望, 观察视野范围内是否存在比当前位置更高的山峰.若存在, 就从当前位置跳到更高的位置. 具体过程如下:

1) 在视野范围内$[x_{ij}-b,x_{ij}+b]$, 随机产生实数 y_{ij}, 令 $Y_i=(y_{i1},y_{i2},\cdots,y_{ij},\cdots,y_{iD})$, 那么 $Y=(Y_1,Y_2,\cdots,Y_j,\cdots,Y_D)$. 其中, b 为视野长度, 它决定了猴子从当前位置眺望的最远距离.

2) 如果向量$Y_i=(y_{i1},y_{i2},\cdots,y_{ij},\cdots,y_{iD})$在$[x_{\min},x_{\max}]$的范围内, 并且$f(Y_i)<f(X_i)$, 更新 X_i 为 Y_i; 否则, 重复 1), 直至找到可行的 Y_i.

3) 以 Y 作为初始位置, 执行爬过程.

Step4. 执行跳过程: 它的主要目的是由当前区域转移到新的区域进行搜索. 选择所有猴子的重心位置作为支点, 每只猴子从当前位置朝着支点的方向跳到新的区域进行搜索, 具体过程如下:

1) 在跳区间[c, d]内随机生成一个实数 θ.

2) 令:

$$Y_i = x_{ij} + \theta\left(p_j - x_{ij}\right) \tag{5}$$

其中, $p_j = \frac{1}{M}\sum_{i=1}^{M} x_{ij}$, 其中$p = (p_1, p_2, \cdots, p_D)$被称为支点.

3) 如果向量$Y_i = \left(y_{i1}, y_{i2}, \cdots, y_{ij}, \cdots, y_{iD}\right)$在$[x_{\min}, x_{\max}]$范围内, 并且$f(Y_i) < f(X_i)$, 更新 X_i 为 Y_i; 否则, 重复 1) 和 2), 直至找到可行的 Y_i.

Step5. 终止条件: 重复 Step2–Step4, 直到达到预先设定的迭代次数 N, 算法终止, 输出结果.

2.2 诱导因子猴群算法

在基本猴群算法中, 爬过程是非常重要的, 它是控制算法的搜索精度. 可是, 在基本猴群算法中, 猴群位置在爬过程中的更新, 是没有规律的, 往往具有很大的随机性, 使得算法很难找到局部范围内的最优个体, 从而不容易找到全局最优个体, 这样就降低了算法的求解精度.

为了克服算法的这个缺陷, 本文在爬过程中引入诱导过程, 提出了一种诱导因子猴群算法. 具体为: 在猴子爬行过程中, 给定猴子向上爬行的指示, 诱导它们向最优位置的方向爬行, 这样可以避免绕路, 尽快找到局部范围内的最优个体, 从而迅速找到全局最优个体, 来提高算法的寻优能力, 达到提高算法的精度, 其具体操作如下.

在爬过程中, 随机生成爬向量 $\Delta X_i = (\Delta x_{i1}, \Delta x_{i2}, \cdots, \Delta x_{ij}, \cdots, \Delta x_{iD})(1 \le i \le M, 1 \le j \le D)$, 其中, 分量 Δx_{ij}由下述过程生成.

随机生成一个 $M \times D$ 阶矩阵 $\beta = (\beta_{i1}, \beta_{i2}, \cdots, \beta_{ij}, \cdots, \beta_{iD})(1 \le i \le M, 1 \le j \le D)$, 其中, 分量 β_{ij} 是在区间[0, 1]上随机生成的一个实数. 当 $\beta_{ij} > e^{1-\frac{Nc}{Nc+1-kc}}$时, $\Delta x_{ij} = 0$; 否则, $\Delta x_{ij} = a$. 这样, 分量 Δx_{ij}就可由下式 (6) 来表示.

$$\Delta x_{ij} = \begin{cases} 0, & \beta_{ij} > e^{1-\frac{Nc}{Nc+1-kc}} \\ a, & \beta_{ij} \le e^{1-\frac{Nc}{Nc+1-kc}} \end{cases} \tag{6}$$

这样, 在猴子爬行的过程中, 就可以通过爬向量 ΔX_i中 0 的个数来诱导猴子的爬行, 来提高算法的寻优能力, 进而达到提高算法的求解精度. 其中, M 为种群规模, D 为维数, N_c 为爬过程中爬的次数, k_c 为每次爬行的迭代次数, 参数$a(a > 0)$为猴子的爬步长.

在爬过程中引入诱导过程时, 为了达到提高算法寻优能力的目的, 那么就需要选择一个合适的诱导次数. 如果诱导次数过低, 提高算法寻优能力就不太明显. 如果诱导次数过高, 算法往往会出现陷入局部最优, 达不到提高算法寻优能力的目的. 因此, 本文经过多次仿真后, 找到当诱导次数$guide = 5$时, 既可以提高局部寻优能力, 又可以避免因过高的局部寻优能力而降低了全局寻优能力, 从而提高了算法的寻优性能.

3 利用诱导因子猴群算法求解 0-1 背包问题

利用诱导因子猴群算法求解 0-1 背包问题时, 就是以式 (1) 作为目标函数, 寻找式 (1) 的最大值. 在求解中, 它们的对应关系为: 每只猴子的位置, 相当于式 (1) 的一组可行解; 猴群规模对应可行解的个数; 维数对应物品的数量. 实施步骤如下.

Step1. 参数值设定, 如种群规模 M=20, 最大迭代次数$nMax = 500$等.

Step2. 随机生成一个二进制向量 $X_i = (x_{i1}, x_{i2}, \cdots, x_{ij}, \cdots, x_{iD})$作为初始位置 (其中, 第 i 位为 1 表示第 i 件物件被选中, 若为 0, 则表示第 i 件物件未被选中). 并计算所有猴子的目标函数值$f(X_i)$, 找出较优位置.

Step3. 执行爬过程, 计算, $Y_i = X_i + a \cdot sign\left(f_{ij}'(X_i)\right)$, 产生新的二进制向量$Y_i = \left(y_{i1}, y_{i2}, \cdots, y_{ij}, \cdots, y_{iD}\right)$, 若$f(Y_i) > f(X_i)$, 则更新第 i 只猴子当前的位置, 即$X_i = Y_i$. 直到预定的执行次数 Nc. 然后, 为了能够优先选取物品中单位体积价值较高的装入背包, 在这里设计一个诱导因子, 具体过程如下.

① 求出每件物品的单位体积的价值量, 第 i 件物品为:

$$k_i = \frac{p_i}{w_i} \tag{7}$$

② 求出每件物品所占背包的比例, 第 i 件物品为:

$$t_i = \frac{w_i}{V} \tag{8}$$

③ 据此设计诱导因子为:

$$q_i = k_i + \frac{\alpha}{t_i} \tag{9}$$

$$m_i = \frac{q_i - q_{\min}}{q_{\max} - q_{\min}} \tag{10}$$

其中, α 为常量, 用来调整 k_i 和$\frac{1}{t_i}$的关系.

④ 随机选取两个值 c_1 和 c_2, 其中, $c_1, c_2 \in \{1,2,\cdots,D\}$. 如果$m_{c_1} > m_{c_2}$并且$y_{ic_1} \neq y_{ic_2}$, 则更新猴子的位置的变量为$\begin{cases} y'_{ic_1} = 1 \\ y'_{ic_2} = 0 \end{cases}$, 否则更新为$\begin{cases} y'_{ic_2} = 1 \\ y'_{ic_1} = 0 \end{cases}$. 计算$f(Y_i)$, 若$f(Y_i) > f(X_i)$, 则更新第 i 只猴子的位置. 直到一定的诱导次数 *guide*.

Step4. 执行望过程, 在视野范围$\left[x_{ij} - b, x_{ij} + b\right]$内生成新的二进制向量$Y_i = \left(y_{i1}, y_{i2}, \cdots, y_{ij}, \cdots, y_{iD}\right)$, 并计算$f(Y_i)$, 若$f(Y_i) > f(X_i)$, 则更新第 i 只猴子当前的位置, 即$X_i = Y_i$.

Step5. 执行跳过程, 计算, $Y_i = x_{ij} + \theta\left(p_j - x_{ij}\right)$, 产生新的二进制向量$Y_i = \left(y_{i1}, y_{i2}, \cdots, y_{ij}, \cdots, y_{iD}\right)$, 计算计算$f(Y_i)$, 若$f(Y_i) > f(X_i)$, 则更新第 i 只猴子当前的位置, 即$X_i = Y_i$.

Step6. 重复 Step3–Step5, 直到达到预定的最大迭代次数 *nMax*, 算法结束, 输出最优个体和最优值.

4 实例仿真及其分析

为了验证所给算法的可行性, 下边给出两个算例, 它们的规模由小变大, 具体数据见以下相关算例数据. 比较差分进化算法 (Differential Evolution Algorithm, DEA)[8]、粒子群算法 (Particle Swarm Optimization, PSO)[9]、遗传算法 (Genetic Algorithm, GA)[10]、基本猴群算法 (Basic Monkey Algorithm, BMA)[11]以及本文所给诱导因子猴群算法 (Inducing Factor Monkey Algorithm, IFMA) 求解问题的效果.

算例 1. 物品数量 n=50, 背包容量 V=1000, 物品价值 R=[220, 208, 198, 192, 180, 180, 65, 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], 物品重量 W=[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].

算例 2. 物品数量 n=100, 背包容量 V=2010, 物品价值 R=[68, 101, 125, 159, 65, 146, 28, 92, 143, 37, 5, 154, 183, 117, 96, 127, 139, 113, 100, 95, 12, 134, 65, 112, 69, 45, 158, 60, 142, 179, 36, 43, 107, 143, 49, 6, 130, 151, 102, 149, 24, 155, 41, 177, 109, 40, 124, 139, 83, 142, 116, 59, 131, 107, 187, 146, 73, 30, 174, 13, 91, 37, 168, 175, 53, 151, 125, 31, 192, 138, 88, 184, 110, 159, 189, 147, 31, 169, 192, 56, 160, 138, 84, 42, 151, 37, 21, 22, 200, 85, 135, 200, 139, 189, 68, 94, 84, 22, 18, 115], 物品重量 W=[42, 35, 70, 79, 63, 6, 82, 62, 96, 28, 92, 3, 93, 22, 19, 48, 72, 70, 68, 36, 4, 23, 74, 42, 54, 48, 63, 38, 24, 30, 17, 91, 89, 41, 65, 47, 91, 71, 7, 94, 30, 85, 57, 67, 32, 45, 27, 38, 19, 30, 34, 40, 5, 78, 74, 22, 25, 71, 78, 98, 87, 62, 56, 56, 32, 51, 42, 67, 8, 8, 58, 54, 46, 10, 22, 23, 7, 14, 1, 63, 11, 25, 49, 96, 3, 92, 75, 97, 49, 69, 82, 54, 19, 1, 28, 29, 49, 8, 11, 14].

在仿真中, 用本文所给 IFMA 求解上述问题时, 算法的参数设定为: 种群规模 M=20, 最大迭代次数 *nMax*=500, 爬步长 a=1, 爬的迭代次数 N_c=20, 诱导次数 *guide*=5, 视野长度 b=0.5, 望的迭代次数 N_w=3, 跳区间$[c,d] = [-1,1]$, 常量 α=1.2.

利用上述五种算法分别对算例 1 和算例 2 进行一次求解, 得到算例 1 的最优值和最优个体罗列在表 1, 寻优过程如图 1 所示, 算例 2 的最优值和最优个体罗列在表 2, 优化过程曲线如图 2 所示.

表 1 算例 1 的最优值和最优个体

算法	最优值	最优个体
DEA	2619	11001000010111101001001001011111101011111111011011
PSO	2482	11011101000010101010010001000111111000101111111000
GA	2810	00101000111111111101100111111011010100101001101000
BMA	2877	11011001111010011111100100110100000010111101100001
IFMA	3019	11010101111011011011100111110000001011010011111111

表 2 算例 2 的最优值和最优个体

算法	最优值	最优个体
DEA	5079	1101010100001000101110001110111000100000111100101 1100110000101011000000110001101000011011111000011
PSO	6346	0111010111001110010010001001001010100010101010001000 0011001010001101001101111001101000100010111100000
GA	6067	0001000010001011110000010010010010000101010110010100 0001000100011110111010111011011001010100111000100
BMA	6659	1100010101010111000011110010111010010100000111111110 0100100100001111011011011111110001111100111100111
IFMA	7883	1100010000010111100111010011111001000110100110111111 0111010001101101101111111110111010001011111111111

接着, 进一步利用这 5 种算法分别对算例 1 和算例 2 进行 50 次求解, 得出它们的最优值、最差值、平均值和方差分别罗列在如表 3 和表 4.

表 3 算例 1 的结果

算法	最优值	最差值	平均值	方差
DEA	2845	2433	2.6496e+003	6.9232e+003
PSO	2999	2482	2.8772e+003	1.2611e+004
GA	2991	2622	2.8783e+003	4.5210e+003
BMA	2963	2840	2.9063e+003	802.5322
IFMA	3072	3019	3.0438e+003	132.7629

表 4 算例 2 的结果

算法	最优值	最差值	平均值	方差
DEA	5809	4818	5.2604e+003	7.1192e+004
PSO	7391	5499	6.9648e+003	1.5032e+005
GA	6639	4906	5.9914e+003	1.4987e+005
BMA	7034	6176	6.5908e+003	3.7698e+004
IFMA	7968	7818	7.8955e+003	951.6004

通过以上仿真结果中的图 1, 图 2 和表 1 至表 4 可以看出, 文中在 BMA 算法中的爬过程中引入诱导过程后, 很好地诱导了猴子向着指定的方向爬行, 从而避免了算法走弯路, 提高了算法的局部寻优能力和全局寻优能力. 而且随着物品数量的增大, 复杂度的增强, 相对于 DEA, PSO, GA 和 BMA 算法, 所给 IFMA 算法的求解精度仍然较高, 且方差较小. 从而说明所给 IFMA 算法具有较高的可行性和稳定性, 达到了预期的效果, 为求解 0-1 背包问题提供了新的途径.

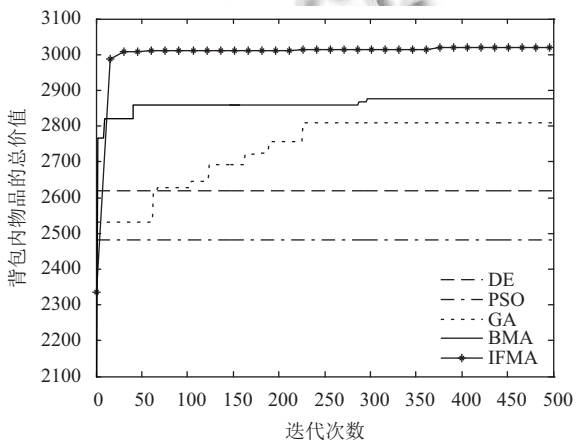

图 1 算例 1 优化过程曲线

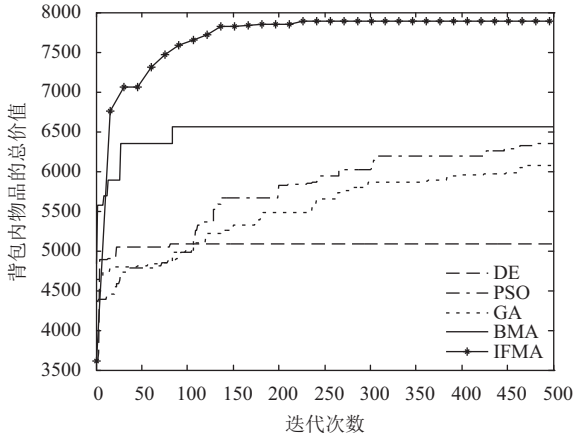

图 2 算例 2 优化过程曲线

5 结束语

在利用猴群算法求解 0-1 背包问题时, 在爬过程中, 引入诱导因子后, 可以诱导猴子向着指定的方向爬行, 从而避免绕路, 使得所给 IFMA 能够找到全局最优解. 通过仿真实验表明, 利用所给的 IFMA 求解 0-1 背包问题达到了预期的结果, 提高了求解精度.

参考文献

1 Merkle R, Hellman M. Hiding information and signatures in trapdoor knapsacks. IEEE Transactions on Information Theory, 1978, 24(5): 525–530. [doi: 10.1109/TIT.1978.1055927]

2 赵学武, 刘向娇, 王兴, 等. 求解 0-1 背包问题的遗传算法. 南阳师范学院学报, 2014, 13(6): 21–25.

3 Srinivasan V, Varghese G. Fast address lookups using controlled prefix expansion. ACM Transactions on Computer Systems, 1999, 17(1): 1–40. [doi: 10.1145/296502.296503]
4 李鸣山, 郑海虹. 0-1 背包问题的多重分枝一限界算法. 武汉测绘科技大学学报, 1995, 20(1): 83–87.
5 宁爱兵, 马良. 0/1 背包问题快速降价法及其应用. 系统工程理论方法应用, 2005, 14(4): 372–375.
6 王粉兰, 孙小玲. 不可分离凸背包问题的拉格朗日分解和区域分割方法. 运筹学学报, 2004, 8(4): 45–53.
7 王乐, 王世卿, 张静乐. 基于 Matlab 的 0-1 背包问题的动态规划方法求解. 计算机技术与发展, 2006, 16(4): 88–89, 92.
8 荆源. 背包问题的差分进化算法. 才智, 2011, (7): 92–93.
9 赵传信, 季一木. 粒子群优化算法在 0/1 背包问题的应用. 微机发展, 2005, 15(10): 23–25. [doi: 10.3969/j.issn.1673-629X.2005.10.009]
10 乐天. 遗传算法求解 0/1 背包问题的综述. 浙江海洋学院学报 (自然科学版), 2013, 32(1): 71–74.
11 Zhao RQ, Tang WS. Monkey algorithm for global numerical optimization. Journal of Uncertain Systems, 2008, 2(3): 164–176.
12 李晓, 倪富陶, 孙维刚, 等. 基于猴群算法的连续刚构桥传感器优化布置研究. 公路, 2016, 61(3): 65–69.
13 张佳佳, 张亚平, 孙济洲. 基于猴群算法的入侵检测技术. 计算机工程, 2011, 37(14): 131–133. [doi: 10.3778/j.issn.1002-8331.2011.14.037]
14 赵涛, 夏雨, 宗玛利. 基于猴群算法的加气站项目进度研究. 价值工程, 2010, 29(8): 90–92.
15 申彩英, 高韬. 基于猴群算法的混合动力汽车能量管理策略. 计算机工程与应用, 2014, 50(14): 9–13, 120. [doi: 10.3778/j.issn.1002-8331.1308-0015]
16 Ituarte-Villarreal CM, Lopez N, Espiritu JF. Using the monkey algorithm for hybrid power systems optimization. Procedia Computer Science, 2012, (12): 344–349. [doi: 10.1016/j.procs.2012.09.082]
17 Hodashinsky IA, Samsonov SS. Design of fuzzy rule based classifier using the monkey algorithm. Mathematical Methods and Algorithms of Business Informatics, 2017, 1(39): 61–67.