

# 求解集合联盟背包问题的二次贪心变异乌鸦算法

刘雪静, 贺毅朝, 吴聪聪

(河北地质大学 信息工程学院, 河北 石家庄 050031)

**摘 要:** 针对确定性算法难以求解的集合联盟背包问题(Set-Union Knapsack Problem, SUKP), 提出了二次贪心变异乌鸦算法(quadratic greedy mutated crow search algorithm, QGMCSA). 首先结合 SUKP 问题模型对贪心策略进行改进, 提出了处理其潜在解的二次贪心修复和优化策略; 其次, 为了扩大乌鸦个体的搜索范围, 对乌鸦算法进行变异操作, 在跟踪过程中引入莱维飞行; 最后, 利用三类 SUKP 实例验证本文算法. 仿真结果表明: QGMCSA 是比二进制人工蜂群算法求解 SUKP 的结果更优的一个高效算法.

**关键词:** 集合联盟背包问题; 乌鸦算法; 二次贪心修复与优化; 莱维飞行

**中图分类号:** TP18

**文献标识码:** A

**文章编号:** 1000-7180(2018)11-0013-07

## Quadratic Greedy Mutated Crow Search Algorithm for Solving Set-Union Knapsack Problem

LIU Xue-jing, HE Yi-chao, WU Cong-cong

(College of Information Engineering, Hebei GEO University, Shijiazhuang 050031, China)

**Abstract:** In order to solve the Set-Union Knapsack Problem (SUKP) which is difficult to be solved by the deterministic algorithm, the quadratic greedy mutated crow search algorithm (QGMCSA) is proposed. Firstly, the greedy strategy is improved by combining the SUKP model, and the quadratic greedy repair and optimization strategy is proposed to deal with its potential solutions. Secondly, in order to enlarge the search scope of the individual, Lévy flight is introduced into the tracking process of crow search algorithm. Finally three kinds of SUKP instances are used to verify the algorithm. Simulation results show that QGMCSA is better than binary artificial bee colony algorithm for solving SUKP, it is a an effective algorithm.

**Key words:** set-union knapsack problem; crow search algorithm; quadratic greedy repair and optimization; Lévy flight

### 1 引言

集合联盟背包问题(Set-Union Knapsack Problem, SUKP)<sup>[1-2]</sup>是 0-1 背包问题(0-1 knapsack problem, 0-1 KP)的一种扩展形式, 在财务决策<sup>[2]</sup>、智慧城市<sup>[3]</sup>、建立公钥模型<sup>[4]</sup>等领域具有广泛的应用价值. SUKP 是 NP 完全问题, 非常难于求解. 目前求解 SUKP 的算法主要有两类: 确定性算法和非确定性算法. Goldschmidt 等<sup>[1]</sup>提出利用动态规划法求解 SUKP, 算法的时间复杂度是指数阶

的, 不具有实用性; Arulselvan<sup>[2]</sup>基于贪心策略提出了求解 SUKP 的近似算法 A-SUKP, 但是算法的近似性能较低, 而且计算复杂度仍然较高; He 等<sup>[5]</sup>提出利用人工蜂群算法求解 SUKP, 并基于贪心策略给出了利用演化算法求解 SUKP 时处理不可行解的修复与优化算法. 目前, SUKP 问题已成为了 KP 问题的一个研究热点, 如何快速求解它是一个非常值得研究的问题.

乌鸦算法(Crow Search Algorithm, CSA)<sup>[6]</sup>是一种新颖的群智能算法, 在医学<sup>[7]</sup>、网络优化分

配<sup>[8]</sup>、背包问题<sup>[9]</sup>等领域有较少的研究成果. Oliva 等<sup>[7]</sup>将乌鸦算法应用到图像领域,提出基于乌鸦算法的图像分割方法; Satpathy 等<sup>[8]</sup>提出基于乌鸦算法的云数据中心资源感知虚拟机配置策略,以优化能耗和提高数据中心利用率; 刘雪静等<sup>[9]</sup>提出利用改进的乌鸦算法求解折扣 $\{0-1\}$ 背包问题,得到较好的近似解. 本文将研究如何用乌鸦算法求解 SUKP 这一新型的背包问题.

## 2 SUKP 的数学模型

SUKP 的数学模型如下: 假定  $U = \{1, 2, \dots, n\}$  是元素集合, 每一个元素  $j (j = 1, 2, \dots, n)$  具有一个重量  $w_j > 0$ ;  $S = \{1, 2, \dots, m\}$  是项的集合, 每一个项  $i (i = 1, 2, \dots, m)$  为元素的非空子集  $U_i \subseteq U$ , 其价值为  $p_i > 0$ ; 对于任意非空子集  $A \subseteq S$ , 其价值为  $P(A) = \sum_{i \in A} p_i$ , 重量为  $W(A) = \sum_{j \in \bigcup_{i \in A} U_i} w_j$ . SUKP 为在给定的背包载重  $C$  下, 求得子集  $S^* \in S$ , 使其在满足  $W(S^*) \leq C$  的前提下  $P(S^*)$  达到最大.

假定 SUKP 中项  $S$  即子集族  $\{U_1, U_2, \dots, U_m\}$  是元素集合  $U$  的一个覆盖, 且  $U_i \neq \emptyset$ ; 所有项的重量之和  $W(S) > C$  且对  $i \in S$  有  $\sum_{j \in U_i} w_j \leq C$ ;  $p_i, w_j$  和  $C$  均为正整数. 设  $X = [x_1, x_2, \dots, x_m] \in \{0, 1\}^m$  是  $m$  维  $0-1$  向量,  $A_X = \{i \mid x_i \in X \text{ 且 } x_i = 1, 1 \leq i \leq m\} \subseteq S$ , 则 SUKP 的整数规划模型<sup>[1,5]</sup>为

$$\max f(X) = \sum_{i=1}^m p_i x_i \quad (1)$$

$$\text{s. t. } W(A_X) = \sum_{j \in \bigcup_{i \in A} U_i} w_j \leq C \quad (2)$$

明显地, 任意向量  $X = [x_1, x_2, \dots, x_m] \in \{0, 1\}^m$  仅是 SUKP 的一个潜在解, 满足约束条件 (2) 的  $X$  时才是 SUKP 的一个可行解.

## 3 二进制乌鸦算法

### 3.1 乌鸦算法

Askarzadeh<sup>[6]</sup>在研究乌鸦的觅食行为之后提出乌鸦算法, 自然界中的乌鸦有如下行为: 乌鸦能记住自己藏匿食物的最佳位置 (memory, M), 为偷窃其他乌鸦的食物而跟踪其他乌鸦, 被跟踪的乌鸦能以一定的感知概率 (awareness probability, AP) 保护自己的食物防止被窃取, 跟踪成功后乌鸦将参照飞

行长度 (flight length, fl) 到达被跟踪乌鸦的最佳位置附近. CSA 描述如下.

#### 算法 1. CSA

Step1: 初始化.

初始化参数 MaxIter, fl, AP 和种群大小  $N$ , 随机生成乌鸦  $X_i = [x_1, x_2, \dots, x_m] (1 \leq i \leq N)$ .

Step2: 初始化  $X_i$  的记忆值  $M_i$ , 计算  $f(X_i)$ ,  $1 \leq i \leq N$ .

Step3: 更新乌鸦位置.

假定乌鸦  $i$  随机选择乌鸦  $j$  跟踪, 分两种情况:

① 乌鸦  $j$  不知道乌鸦  $i$  跟踪它, 则乌鸦  $i$  将接近乌鸦  $j$  的最佳位置  $M_j$ ;

② 乌鸦  $j$  知道乌鸦  $i$  跟踪它, 则乌鸦  $j$  把乌鸦  $i$  带到随机位置.

由此乌鸦  $i$  的位置更新公式如式 (3) 所示.

$$X_i^{\text{iter}+1} = \begin{cases} X_i^{\text{iter}} + r_i * fl * (M_j^{\text{iter}} - X_i^{\text{iter}}) & r_j \geq AP \\ \text{a random position} & \text{else} \end{cases} \quad (3)$$

式中,  $r_i, r_j$  为服从  $[0, 1]$  均匀分布的随机数;  $M_j^{\text{iter}}$  为 iter 次迭代时乌鸦  $j$  的最佳位置,  $X_i^{\text{iter}}$  为 iter 次迭代时乌鸦  $i$  的位置.

Step4: 检查  $X_i^{\text{iter}+1}$  的可行性并计算  $f(X_i^{\text{iter}+1})$ .

Step5: 更新  $M_i^{\text{iter}+1}$ .

$$M_i^{\text{iter}+1} = \begin{cases} X_i^{\text{iter}+1} & f(X_i^{\text{iter}+1}) > f(M_i^{\text{iter}}) \\ M_i^{\text{iter}} & \text{else} \end{cases} \quad (4)$$

Step6: 重复 Step 3 ~ Step 5, 直至迭代结束.

Step7: 返回  $M_{\text{best}}$ , 即问题的最优解.

由于  $N, \text{MaxIter}$  是关于  $m$  的线性函数, 显然地, CSA 的时间复杂度为  $O(m^3)$ .

### 3.2 混合编码方法

在 CSA 中, 由于乌鸦个体由实型向量  $X' = [x'_1, x'_2, \dots, x'_m]$  表示, 故不能直接用于求解 SUKP 问题, 则需将  $X'$  转换为二进制向量  $X$ , 以实现离散问题的求解. 将  $X'$  中每个维度上的实型分量  $x'_i (i = 1, 2, \dots, m)$  采用式 (5) 的方式映射为  $0-1$  分量, 投影映射为  $\phi: [-a, a]^m \rightarrow \{0, 1\}^m$ .

$$x_i = \begin{cases} 1, & \text{sig}(x'_i) \geq 0.5 \\ 0, & \text{sig}(x'_i) < 0.5 \end{cases} \quad (5)$$

式中,  $X = [x_1, x_2, \dots, x_m], x_i \in \{0, 1\}, i = 1, 2, \dots, m$ .

基于上述转换方式, 利用映射  $\phi$  来重新定义乌鸦个体的适应值, 原乌鸦个体的适应度值为  $f(X')$ , 通

过映射  $\psi$  将  $X'$  转化为  $X$ , 其适应度为  $f(X)$ . 采用混合编码方式的二进制乌鸦算法即可求解离散问题.

## 4 求解 SUKP 的二次贪心变异乌鸦算法

### 4.1 二次贪心修复与优化算法

由于 SUKP 是一个约束优化问题, 当使用二进制乌鸦算法求解 SUKP 时肯定会产生不可行解, 因此如何处理不可行解是需要解决的关键问题. 不可行解的解决方法通常有罚函数法和修复法, Michalewicz<sup>[10]</sup> 比较了罚函数法和修复法, 发现修复法更适合处理 0-1 KP 的不可行解.

本文在 S-GROS<sup>[5]</sup> 的基础上, 提出了处理二元向量  $X = [x_1, x_2, \dots, x_m]$  的二次贪心修复与优化策略 (quadratic greedy repair and optimization strategy, QGROS). 假定  $d_j (j = 1, 2, \dots, n)$  是元素  $j (j \in U)$  在项集  $\{U_1, U_2, \dots, U_m\}$  中出现的频率, 令  $\text{weight}_i = \sum_{j \in U_i} (\omega_j / d_j) (i = 1, 2, \dots, m)$  为处理后的每一项集的重量,  $H[1 \dots m]$  中存放采用快排序 QuickSort() 对  $p_i / \text{weight}_i$  进行非升序排序的下标序. QGROS 算法描述如下.

#### 算法 2 QGROS

输入:  $X = [x_1, x_2, \dots, x_m] \in \{0, 1\}^m$  和  $H[1 \dots m]$   
 输出:  $Y = [y_1, y_2, \dots, y_m] \in \{0, 1\}^m$  和  $f(Y)$   
 (1) for  $i = 1$  to  $m$  do  $y_i = 0$  end for  
 (2) for  $i = 1$  to  $m$  do  
 (3) if  $(x_{H[i]} = 1 \&\& W(A_Y \cup \{H[i]\}) \leq C)$  then  
 (4)  $y_{H[i]} = 1, A_Y = A_Y \cup \{H[i]\}$   
 (5) endif  
 (6) endfor  
 (7) 不考虑已经加入背包的元素, 重新计算  $d_j (j = 1, 2, \dots, n)$ ,  $\text{weight}_i (i = 1, 2, \dots, m)$  和  $H[1 \dots m]$   
 (8) for  $i = 1$  to  $m$  do  
 (9) if  $(x_{H[i]} = 0 \&\& W(A_Y \cup \{H[i]\}) \leq C)$  then  
 (10)  $y_{H[i]} = 1, A_Y = A_Y \cup \{H[i]\}$   
 (11) Endif  
 (12) endfor  
 (13) return( $Y, f(Y)$ )

(1) 为  $Y$  赋初值, 时间复杂度  $O(m)$ ; (2) ~ (6) 为修复阶段, 获得初步可行解  $Y$ , 时间复杂度为  $O(nm)$ ; (7) 不考虑已经加入背包的元素, 重新计算未加入背包的物品频度和每一项在修正后的重量, 利用快排序对  $H[1 \dots m]$  重新赋值, 时间复杂度为

$O(m \log m) + O(nm)$ ; (8) ~ (12) 对可行解  $Y$  进行进一步优化, 得到最终解, 时间复杂度为  $O(nm)$ ; 最后输出  $Y$  及  $f(Y)$ . 显然地, QGROS 的时间复杂度是  $O(nm)$ , 是一种处理 SUKP 不可行解的有效方法.

### 4.2 莱维飞行

莱维 (Lévy) 分布是一种概率分布, 布谷鸟、萤火虫等动物的觅食轨迹符合 Lévy 分布<sup>[11]</sup>. Lévy 飞行<sup>[12-13]</sup> 是服从 Lévy 分布的随机搜索路径, 是一种将偶尔较长距离的行走与短距离的搜索相间的行走方式, 布朗运动、随机行走等随机现象都可用 Lévy 飞行解释. Reynolds<sup>[12]</sup> 研究发现, 当种群中个体相互独立, 可行解在解空间中随机且稀疏分布时, Lévy 飞行是一种非常理想的搜索策略, 它能扩大搜索范围, 增加种群多样性, 有助于跳出局部最优.

为了提高乌鸦个体的搜索能力, 在 CSA 中引入 Lévy 飞行, 当被跟踪乌鸦感知到乌鸦  $i$  跟踪它时, 乌鸦  $i$  将进行 Lévy 飞行, 乌鸦  $i$  的位置更新公式如下:

$$X_i^{\text{iter}+1} = X_i^{\text{iter}} + \text{len} \oplus \text{Levy}(\lambda), i = 1, 2, \dots, n \quad (6)$$

式中,  $X_i^{\text{iter}}$  表示  $X_i$  在第  $\text{iter}$  代的位置;  $\text{len}$  为步长;  $\text{Levy}(\lambda)$  为随机搜索路径;  $\oplus$  为点对点乘法.

### 4.3 求解 SUKP 的 QGMCSA 算法

在 CSA 中引入 QGROS 和 Lévy 飞行两种策略, 得到求解 SUKP 的 QGMCSA, 算法描述如下.

#### 算法 3 QGMCSA

输入: 参数  $AP, fl, n, \text{MaxIter}, N, P, W, C$   
 输出: 最优解  $Y$  及  $f(Y)$   
 (1) 计算  $d_j (j = 1, 2, \dots, n)$  和  $\text{weight}_i = \sum_{j \in U_i} (\omega_j / d_j) (i = 1, 2, \dots, m)$   
 (2)  $H[1 \dots m] \leftarrow \text{QuickSort}(p_i / \text{weight}_i)$ ;  
 (3) for  $i = 1$  to  $N$  do  
 (4) 生成初始解  $X_i \in [-a, a]^m$ , 按式(5)生成  $X_i \in \{0, 1\}^m$   
 (5)  $(Y_i, f(Y_i)) \leftarrow \text{QGROS}(X_i, H[1 \dots m])$   
 (6) 初始化  $M_i$   
 (7) endfor  
 (8) while  $\text{iter} < \text{MaxIter}$   
 (9) for  $i = 1$  to  $N$   
 (10) 随机选择乌鸦  $j$   
 (11) if  $r_j \geq AP$   
 (12) 根据式(3)计算  $Y_i^{\text{iter}+1}$   
 (13) else

- (14)  $Y_i$  做 Lévy 飞行
- (15) Endif
- (16) 利用 QGROS 修复并优化  $Y_i$  得  $f(Y_i)$
- (17) 更新  $M_i$
- (18) endfor
- (19)endwhile
- (20)return( $M_{best}$ )

在 QGMCSA 中,QuickSort( $p_i/\text{weight}_i$ ) 的时间复杂度是  $O(m\log m)$ ,SecGROS 的时间复杂度是  $O(mn)$ ,由于  $N$ 、MaxIter 是  $m$  的线性函数,则算法的时间复杂度为  $O(mn) + O(m\log m) + O(mN) + O(n) + \text{MaxIter} * (N * (O(m) + O(m) + O(mn)) \approx O(mn^3)$ ,假定  $M = \max\{m, n\}$ ,则 QGMCSA 的时间复杂度为  $O(M^4)$ ,是一个多项式时间的进化算法.

### 5 实例计算与比较

为了说明本文算法的高效性,利用 QGMCSA 与文献中的 A-SUKP 和文献[5]中 BABC 进行比

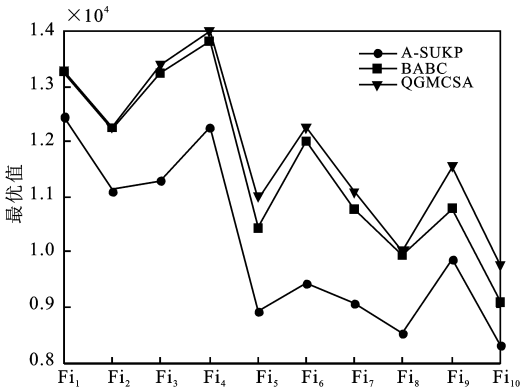
较.所使用的电脑为 Inter(R) Core(TM) i3-4005U CPU @ 1.7GHz,采用 C 语言编程,编译环境为 CodeBlocks,并利用 Matlab(R2012a)绘制曲线.

计算所使用的三类大规模实例  $Fi_1 \sim Fi_{10}$ 、 $Si_1 \sim Si_{10}$  和  $Ti_1 \sim Ti_{10}$  取自文献[5],其下载地址为: <https://www.researchgate.net/project/Set-union-knapsack-problem>. 算法 A-SUKP 和 BABC 的数据来自文献[5],算法 QGMCSA 的参数设置如下:  $N=100$ ,  $\text{MaxIter}=250$ ,  $fl=2.0$ ,  $AP=0.3$ ,其实验数据为算法独立运行 30 次所得.

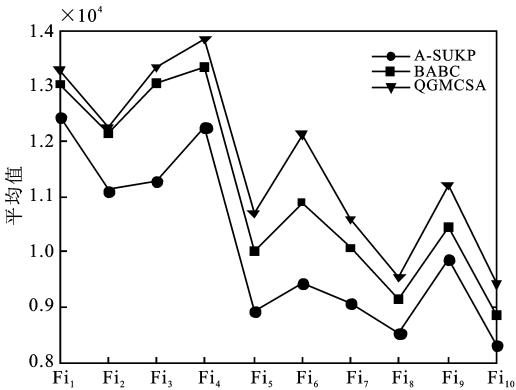
表 1 给出了 A-SUKP, BABC 和 QGMCSA 三种算法求解第一类 SUKP 实例时的最好值、平均值和方差. 图 1 对三种算法求解  $Fi_1 \sim Fi_{10}$  实例时的最好值和平均值进行了比较. 从图 1(a)可以看出, QGMCSA 的求解性能最好, BABC 次之, A-SUKP 最差. 图 1(b)与图 1(a)相比, QGMCSA 对 BABC 来说平均值的优势明显, A-SUKP 求解性能仍然最差.

表 1 求解第一类 SUKP 实例的结果比较

| 算法     | 结果  | $Fi_1$   | $Fi_2$   | $Fi_3$   | $Fi_4$   | $Fi_5$   | $Fi_6$   | $Fi_7$   | $Fi_8$  | $Fi_9$   | $Fi_{10}$ |
|--------|-----|----------|----------|----------|----------|----------|----------|----------|---------|----------|-----------|
| A-SUKP | 最好值 | 12 459   | 11 119   | 11 292   | 12 262   | 8 941    | 9 432    | 9 076    | 8 514   | 9 864    | 8 299     |
|        | 平均值 | 12 459   | 11 119   | 11 292   | 12 262   | 8 941    | 9 432    | 9 076    | 8 514   | 9 864    | 8 299     |
|        | 标准差 | 0.00     | 0.00     | 0.00     | 0.00     | 0.00     | 0.00     | 0.00     | 0.00    | 0.00     | 0.00      |
| BABC   | 最好值 | 13 251   | 12 238   | 13 241   | 13 829   | 10 428   | 12 012   | 10 766   | 9 946   | 10 784   | 9 090     |
|        | 平均值 | 13 028.5 | 12 155.0 | 13 064.4 | 13 359.2 | 9 994.8  | 10 902.9 | 10 065.2 | 9 136.0 | 10 452.2 | 8 857.9   |
|        | 标准差 | 92.63    | 53.29    | 99.57    | 234.99   | 154.03   | 449.45   | 241.45   | 151.90  | 114.35   | 94.55     |
| QGMCSA | 最好值 | 13 283   | 12 272   | 13 405   | 13 993   | 10 960   | 12 245   | 11 080   | 10 023  | 11 517   | 9 750     |
|        | 平均值 | 13 283.0 | 12 247.3 | 13 357.9 | 13 863.4 | 10 705.1 | 12 123.2 | 10 585.3 | 9 540.0 | 11 203.4 | 9 413.1   |
|        | 标准差 | 0.00     | 16.94    | 64.81    | 81.20    | 99.24    | 162.48   | 309.85   | 233.89  | 210.19   | 151.92    |



第一类 SUKP 实例  
(a) 三种算法求解第一类 SUKP 实例的最优值比较



第一类 SUKP 实例  
(b) 三种算法求解第一类 SUKP 实例的平均值比较

图 1 三种算法求解第一类 SUKP 实例的结果比较

实验结果表 2 给出了 A-SUKP, BABC 和 QG-MCSA 三种算法求解第二类 SUKP 实例时的最好值、平均值和方差. 图 2 对三种算法求解  $Si_1 \sim Si_{10}$  实例时的最好值和平均值进行了比较. 从图 2(a)可以看出, QGMCSA 的求解性能最好, BABC 次之, A-SUKP 最差. 仅在  $Si_9$  实例上 QGMCSA 所求最好值不如算法 BABC, 在其他实例上 QGMCSA 虽然所求最好值较 BABC 好, 但是优势不大. 图 2(b)是三

种算法求解  $Si_1 \sim Si_{10}$  时的平均值比较图可知, QG-MCSA 所求的平均值相对 BABC 算法来说优势略有增大, 在  $Si_9$  实例上的平均求解性能两种算法相差不大, A-SUKP 的平均求解性能仍然最差.

表 3 给出了 A-SUKP, BABC 和 QGMCSA 三种算法求解第三类 SUKP 实例时的最好值、平均值和方差. 图 3 对三种算法求解  $Ti_1 \sim Ti_{10}$  实例时的最好值和平均值进行了比较. 从图 3(a)可以看出, A-SUKP 的求解性

表 2 求解第二类 SUKP 实例的结果比较

| 算法     | 结果  | $Si_1$   | $Si_2$   | $Si_3$   | $Si_4$   | $Si_5$   | $Si_6$   | $Si_7$   | $Si_8$  | $Si_9$   | $Si_{10}$ |
|--------|-----|----------|----------|----------|----------|----------|----------|----------|---------|----------|-----------|
| A-SUKP | 最好值 | 13 634   | 11 325   | 10 328   | 9 784    | 10 208   | 9 183    | 9 751    | 8 497   | 9 615    | 7 883     |
|        | 平均值 | 13 634   | 11 325   | 10 328   | 9 784    | 10 208   | 9 183    | 9 751    | 8 497   | 9 615    | 7 883     |
|        | 标准差 | 0.00     | 0.00     | 0.00     | 0.00     | 0.00     | 0.00     | 0.00     | 0.00    | 0.00     | 0.00      |
| BABC   | 最好值 | 13 860   | 13 508   | 11 846   | 11 521   | 12 186   | 10 382   | 10 626   | 9 541   | 10 755   | 9 318     |
|        | 平均值 | 13 734.9 | 13 352.4 | 11 194.3 | 10 945.0 | 11 945.8 | 9 859.7  | 10 101.1 | 9 032.9 | 10 328.5 | 9 180.7   |
|        | 标准差 | 70.76    | 155.14   | 249.58   | 255.14   | 127.80   | 177.02   | 196.99   | 194.18  | 91.62    | 84.91     |
| QGMCSA | 最好值 | 14 044   | 13 508   | 12 211   | 11 868   | 12 644   | 10 513   | 11 067   | 9 946   | 10 443   | 9 765     |
|        | 平均值 | 13 827.0 | 13 501.7 | 12 027.4 | 11 750.8 | 12 501.5 | 10 336.9 | 10 879.2 | 9 654.8 | 10 287.1 | 9 557.0   |
|        | 标准差 | 40.87    | 4.82     | 159.17   | 155.97   | 131.56   | 93.37    | 52.35    | 185.60  | 80.86    | 126.66    |

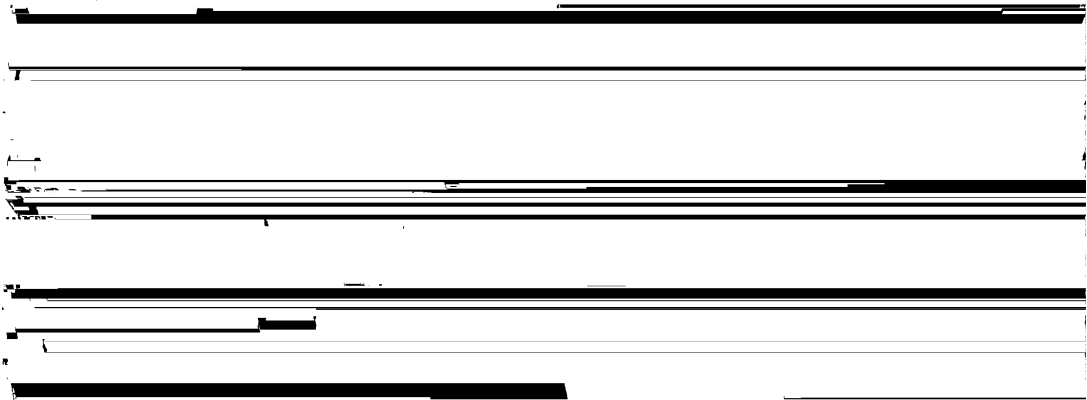


图 2 三种算法求解第二类 SUKP 实例的结果比较

表 3 求解第三类 SUKP 实例的结果比较

| 结果     | Ti <sub>1</sub> | Ti <sub>2</sub> | Ti <sub>3</sub> | Ti <sub>4</sub> | Ti <sub>5</sub> | Ti <sub>6</sub> | Ti <sub>7</sub> | Ti <sub>8</sub> | Ti <sub>9</sub> | Ti <sub>10</sub> |         |
|--------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|------------------|---------|
| A-SUKP | 最好值             | 10 231          | 10 483          | 11 508          | 8 621           | 9 961           | 9 618           | 8 672           | 8 064           | 9 559            | 8 157   |
|        | 平均值             | 10 231          | 10 483          | 11 508          | 8 621           | 9 961           | 9 618           | 8 672           | 8 064           | 9 559            | 8 157   |
|        | 标准差             | 0.00            | 0.00            | 0.00            | 0.00            | 0.00            | 0.00            | 0.00            | 0.00            | 0.00             | 0.00    |
| BABC   | 最好值             | 11 664          | 12 369          | 13 047          | 10 602          | 11 158          | 10 528          | 10 085          | 9 456           | 10 823           | 9 333   |
|        | 平均值             | 11 182.7        | 12 081.6        | 12 522.8        | 10 150.6        | 10 775.9        | 9 897.9         | 9 537.5         | 9 090.0         | 10 483.4         | 9 085.6 |
|        | 标准差             | 183.57          | 193.79          | 201.35          | 152.91          | 116.80          | 186.53          | 184.62          | 156.69          | 228.34           | 115.62  |
| QGMCSA | 最好值             | 11 947          | 12 369          | 13 647          | 10 920          | 11 374          | 10 829          | 10 019          | 9 562           | 10 508           | 9 198   |
|        | 平均值             | 11 541.3        | 12 369.0        | 13 467.9        | 10 797.4        | 11 049.3        | 10 503.6        | 9 644.1         | 9 172.7         | 10 310.2         | 8 997.8 |
|        | 标准差             | 115.80          | 0.00            | 141.41          | 124.95          | 128.07          | 150.51          | 131.95          | 111.40          | 133.07           | 69.04   |

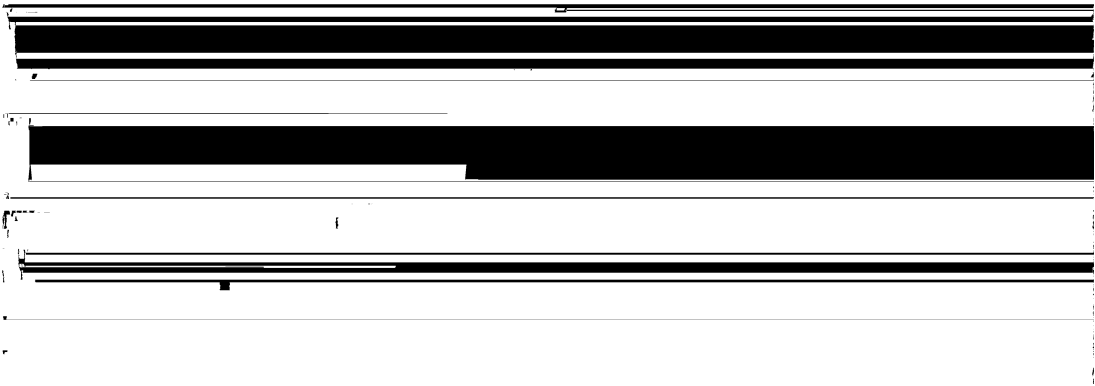


图 3 三种算法求解第三类 SUKP 实例的结果比较

能最差,QGMCSA 在  $Ti_1 \sim Ti_6$ 、 $Ti_8$  实例上的求解性能最好,在  $Ti_7$ 、 $Ti_9$  和  $Ti_{10}$  三个实例上的求解性能不如 BABC. 图 3(b)是三种算法的平均值比较图,QGMCSA 算法在  $Ti_1 \sim Ti_8$  实例上所求的平均值最优,BABC 在  $Ti_9$  和  $Ti_{10}$ 上比 QGMCSA 略优,算法 A-SUKP 在第三类实例上求解性能仍然最差.

图 4 对 BABC 和 QGMCSA 求解三类 SUKP 实例时的标准差进行了比较,A-SUKP 的标准差均为 0,故未列出. 由图 4(a)可知,求解  $Fi_1 \sim Fi_{10}$  实例

时,QGMCSA 的标准差小于 310,而 BABC 最大接近 450;由图 4(b)可知,求解  $Si_1 \sim Si_{10}$  实例时,QGMCSA 的标准差小于 200,大部分实例上的标准差在 150 以下,而 BABC 大部分实例上的标准差在 150~250 之间;由图 4(c)可知,求解  $Ti_1 \sim Ti_{10}$  实例时,QGMCSA 的标准差小于 150,基本比 BABC 小,仅  $Ti_5$  例外. 总的来看,QGMCSA 比 BABC 求解 SUKP 问题的效果更佳,因此是求解 SUKP 的一个更有效的算法.

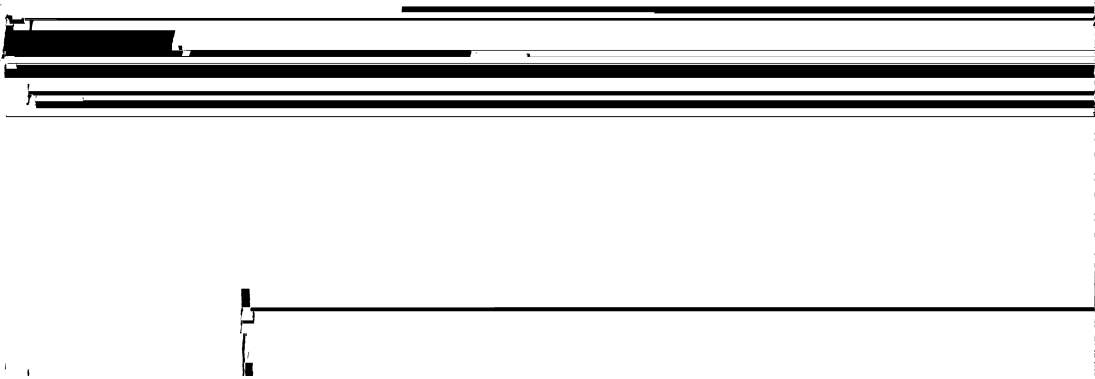


图 4 两种算法求解三类 SUKP 实例的标准差

6 结束语

SUKP 是强 NP 难问题,目前没有解决它的伪多项式时间算法,因此寻找一个有效的快速算法进行求解是非常必要的. 本文研究的是如何利用乌鸦算法求解 SUKP 问题,针对 SUKP 问题提出了一种新的二次贪心策略,并对乌鸦算法的跟踪策略进行变异,引入莱维飞行. 仿真实验表明:QGMCSA 比 A-SUKP 和 BABC 算法更适合求解 SUKP 问题. 在未来的研究中,我们将继续沿这个方向利用其他 EAs 求解 SUKP,以期找到最佳表现的 EA.

参考文献:

[1] Goldschmidt O, Nehme D, Yu G. Note: on the set-union knapsack problem[J]. Naval Research Logistics, 2015, 41(6):833-842.

[2] Arulselvan A. A note on the set union knapsack problem[J]. Discrete Applied Mathematics, 2014, 169(41):214-218.

[3] TU M, Xiao L. System resilience enhancement through modularization for large scale cyber systems [C]// IEEE/CIC International Conference on Communications. China,Qingdao, IEEE, 2016:1-6.

[4] Schneier B. Applied cryptography (2nd ed): proto-

- cols, algorithms, and source code in C[J]. Government Information Quarterly, 1995, 13(3):336.
- [5] He Y C, Xie H, Wong T L, et al. A novel binary artificial bee colony algorithm for the set-union knapsack problem[J]. Future Generation Computer Systems, 2017, 78(1): 77-86.
- [6] Askarzadeh A. A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm[J]. Computers & Structures, 2016(169):1-12.
- [7] Oliva D, Hinojosa S, Cuevas E, et al. Cross entropy based thresholding for magnetic resonance brain images using Crow Search Algorithm[J]. Expert Systems with Applications, 2017(79):164-180.
- [8] Satpathy A, Addya S K, Turuk A K, et al. A resource aware VM placement strategy in cloud data centers based on crow search algorithm[C]// International Conference on Advanced Computing and Communication Systems, , Los Angeles USA - 2017.
- [9] 刘雪静,贺毅朝,路凤佳,等. 基于差分策略的混沌乌鸦算法求解折扣 $\{0-1\}$ 背包问题[J]. 计算机应用, 2018, 38(1):137-145.
- [10] Michalewicz Z. Genetic algorithm+data structure=evolution programs[M]. Berlin, Springer-Verlag, 1996.
- [11] 刘长平,叶春明. 一种新颖的仿生群智能优化算法:萤火虫算法[J]. 计算机应用研究, 2011, 28(9): 3295-3297.
- [12] Reynolds A M. Cooperative random Lévy flight searches and the flight patterns of honeybees [J]. Physics Letters A, 2006, 354(5):384-388.
- [13] Yang X S, Deb S. Cuckoo search via lévy flights[C]// Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on. Pietermaritzburg, South Africa, 2009:210-214.

### 作者简介:

**刘雪静** 女,(1980-),硕士,讲师.研究方向为演化计算、机器学习. E-mail: lxjpass@163.com.

**贺毅朝** 男,(1969-),硕士,教授.研究方向为智能计算、计算复杂性理论.

**吴聪聪** 女,(1975-),硕士,讲师.研究方向为智能计算、信息检索、机器学习.