Journal of Computer Applications
计算机应用,2020,40(5):1291－1294
ISSN 1001-9081
CODEN JYIIDU
2020-05-10
http://www.joca.cn

文章编号:1001-9081(2020)05-1291-04
DOI:10.11772/j.issn.1001-9081.2019091638

贪心二进制狮群优化算法求解多维背包问题

杨　艳[1*],刘生建[1],周永权[2]
(1.广州大学华软软件学院,广州 510990;　2.广西民族大学 信息科学与工程学院,南宁 530006)
(*通信作者电子邮箱yangyan_08@yeah.net)

摘　要:针对经典的多约束组合优化问题——多维背包问题(MKP),提出了一种贪心二进制狮群优化(GBLSO)算法。首先,采用二进制代码转换公式将狮群个体位置离散化,得到二进制的狮群算法;其次,引入反置移动算子对狮王位置进行更新,同时对母狮和幼狮位置重新定义;然后,充分利用贪心算法进行解的可行化处理,增强搜索能力并进一步提高收敛速度;最后,对10个MKP典型算例进行仿真实验,并把GBLSO算法与离散二进制粒子群(DPSO)算法和二进制蝙蝠算法(BBA)进行对比。实验结果表明,GBLSO算法是一种有效的求解MKP的新方法,在求解MKP时具有相对良好的收敛效率、较高的寻优精度和很好的鲁棒性。

关键词:智能算法;贪心算法;贪心二进制狮群优化算法;多维背包问题;组合优化
中图分类号:TP18　　**文献标志码**:A

Greedy binary lion swarm optimization algorithm for solving multidimensional knapsack problem

YANG Yan[1*], LIU Shengjian[1], ZHOU Yongquan[2]
(1. *South China Institute of Software Engineering. GU, Guangzhou Guangdong* 510990, *China*;
2. *School of Information Science and Engineering, Guangxi University for Nationalities, Nanning Guangxi* 530006, *China*)

Abstract: The Multidimensional Knapsack Problem (MKP) is a kind of typical multi-constraint combinatorial optimization problems. In order to solve this problem, a Greedy Binary Lion Swarm Optimization (GBLSO) algorithm was proposed. Firstly, with the help of binary code transform formula, the locations of lion individuals were discretized to obtain the binary lion swarm algorithm. Secondly, the inverse moving operator was introduced to update the location of lion king and redefine the locations of the lionesses and lion cubs. Thirdly, the greedy algorithm was fully utilized to make the solution feasible, so as to enhance the local search ability and speed up the convergence. Finally, Simulations on 10 typical MKP examples were carried out to compare GBLSO algorithm with Discrete binary Particle Swarm Optimization (DPSO) algorithm and Binary Bat Algorithm (BBA). The experimental results show that GBLSO algorithm is an effective new method for solving MKP and has good convergence efficiency, high optimization accuracy and good robustness in solving MKP.

Key words: intelligent algorithm; greedy algorithm; Greedy Binary Lion Swarm Optimization (GBLSO) algorithm; Multidimensional Knapsack Problem (MKP); combinatorial optimization

0　引言

多维背包问题(Multidimensional Knapsack Problem, MKP)是一类典型的组合优化问题,有着广泛的实际应用价值,如项目决策与规划、资源分配、资金预算、货物装载等,对其求解方法的研究无论是在理论上还是实践中都具有一定的意义[1]。求解MKP主要有精确算法和启发式算法两大类:精确算法的时间复杂性都是呈指数增长的,主要用于求解规模相对较小的问题,对大规模问题依赖智能优化算法解决,常见的算法有粒子群优化算法、烟花算法、狼群算法、布谷鸟搜索算法、蚁群算法等[2-9];进化算法多采用精英策略,在具有高进化效率的同时,存在易陷入局部最优解的局限性。文献[2-3]提出将连续的粒子群优化算法通过转换函数生成离散粒子群算法,并用于求解MKP,为求解离散问题提供了一种新方法。二进制反向烟花算法求解MKP具有良好的寻优效果,尤其是在背包维度高、物品数量多的问题中具有良好的寻优能力[6];二进制狼群算法求解MKP时减小了陷入局部极值的概率[7];二进制布谷鸟算法求解背包问题时提高了算法求解精度和收敛速度[8];二进制蚁群算法求解背包问题时提高了算法的全局搜索能力[9]。本文受以上算法思想的启发,通过改进狮群算法求解MKP。

受狮群协作捕猎的启发,文献[10]提出一种新的群智能优化算法——狮群算法,并验证其良好的计算鲁棒性和全局搜索能力,但其主要用于连续函数优化问题。本文在基本狮群算法的基础上,引入二进制编码,加入贪心算法增强局部搜索能力,提出一种基于贪心算法的二进制狮群算法并求解多维背包问题,通过经典算例仿真对比实验验证了算法的有效性。

1　多维背包问题描述

多维背包问题(KMP)可以看作多个0-1背包问题,一个m维0-1背包问题可以描述如下:

收稿日期:2019-09-25;**修回日期**:2019-10-28;**录用日期**:2019-10-30。
基金项目:广东省普通高校重点科研平台和科研项目(2018KQNCX392);广州大学华软软件学院科研项目(ky201823)。
作者简介:杨艳(1985—),女,湖北襄阳人,讲师,硕士,主要研究方向:计算智能、群智能算法;　刘生建(1972—),男,贵州遵义人,讲师,硕士,主要研究方向:群智能算法、深度学习;　周永权(1962—),男,陕西旬邑人,教授,博士,主要研究方向:计算智能、神经网络。

已知n个价值为$p_i(i=1,2,\cdots,n)$的物品，m个容量大小为$C_j(j=1,2,\cdots,m)$的容器，第i个物品所占第j个容器的容积大小为w_{ij}。KMP要解决的就是如何选择物品装入这m个容器，使得装入容器的物品总价值最大。问题的数学模型为：

$$\max f(x_1,x_2,\cdots,x_n)=\sum_{i=1}^{n}p_i x_i \tag{1}$$

$$\text{s.t.}\ \sum_{i=1}^{n}w_{ij}x_i\leqslant C_j, j=\{1,2,\cdots,m\}; x_i\in\{0,1\}, i=\{1,2,\cdots,n\}$$

其中：$f(x_1,x_2,\cdots,x_n)$为目标函数；n为物品的编号；m为容器的编号；p_i为第i个物品的价值；C_j为第j个容器的容积；w_{ij}为第i个物品所占第j个容器的容积大小；x_i为0-1决策变量，当物品i为选择装入时x_i=1，否则x_i=0。

通过引入惩罚系数将带约束的离散化的多维背包问题转化成无约束最优化问题，转化后的无约束优化问题模型如下：

$$\max f(x)=\sum_{i=1}^{n}p_i x_i-\mu\sum_{j=1}^{m}\left[\max\left(0,\sum_{i=1}^{n}w_{ij}x_i-C_j\right)\right] \tag{2}$$

其中惩罚系数μ是一个很大的常数，取值为10^{10}，以保证可行解的最差值都要优于不可行解的最优值。

2 贪心二进制狮群算法求解多维背包问题

2.1 狮群位置数据的离散化

将狮群领地抽象为一个$N\times m$的欧氏空间，N为人工狮的总数量，m为编码长度；第i个人工狮在第t代的位置为$\boldsymbol{x}_i^t=(x_{i1}^t,x_{i2}^t,\cdots,x_{ij}^t,\cdots,x_{im}^t)$，$x_{ij}^t$为位置$\boldsymbol{X}_i$在第$t$代时第$j$维的分量值，且只能取0或1；人工狮感受到的猎物优劣即为目标函数值，通过式(2)求得。

基本的狮群算法主要适用于求解连续空间优化问题，二进制狮群算法求解多维背包问题时需将狮子的位置设为1或0，而狮子个体进行位置更新后产生的位置分量x_{ij}^t不一定是1或0，使得狮群算法无法对其进行求解。由于狮子下一个位置$\boldsymbol{X}_i^{t+1}$由位置改变量$\Delta\boldsymbol{X}_i$决定，$\Delta\boldsymbol{X}_i=\boldsymbol{X}_i^{t+1}-\boldsymbol{X}_i^t$，$\Delta\boldsymbol{X}_i$反映了$\boldsymbol{X}_i^{t+1}$的各个分量取1的概率。为了使概率值在[0,1]区间，贪心二进制狮群优化(Greedy Binary Lion Swarm Optimization, GBLSO)算法采用式(3)的转换函数对$\Delta\boldsymbol{X}_i$进行处理，然后采用式(4)对狮子位置分量置1或0操作[2,4]：

$$T(\Delta x_{ij})=\left|\frac{2}{\pi}\arctan\left(\frac{\pi}{2}(\Delta x_{ij})\right)\right| \tag{3}$$

$$x_{ij}^{t+1}=\begin{cases}1, & rand()<T(\Delta x_{ij})\\ 0, & rand()\geqslant T(\Delta x_{ij})\end{cases} \tag{4}$$

其中：t为当前代数，$rand()$为[0,1]上均匀分布的随机数。

2.2 狮群位置更新规则

2.2.1 狮王位置更新规则

狮王位置是当前群体最优位置，其下一个位置通过反置移动算子计算。反置移动算子$\Theta(\boldsymbol{X}_i,M,r)$表示$\boldsymbol{X}_i$在集合$M$指定的位置上随机抽出$r$个位置进行反置运算。狮王通过在可行位置中随机找一个位置进行取反，从而实现位置的更新，位置更新如式(5)：

$$\boldsymbol{X}^{t+1}=\Theta(\boldsymbol{gBest}^t,\{1,2,\cdots,m\},1) \tag{5}$$

其中：t为当前代数；$\boldsymbol{gBest}^t$是当前狮群发现的最佳猎物的位置。

2.2.2 母狮位置更新规则

母狮之间协作捕猎，向下一个位置移动时需考虑在自身位置$\boldsymbol{X}_i^t$上移动还是以母狮个体最优位置$\boldsymbol{pBest}_i^t$为基础移动；本文通过一个随机值来确定母狮移动前的位置。母狮下一个位置通过移动前位置$\boldsymbol{Xold}_i^t$和协作狮位置$\boldsymbol{Xc}_i^t$求平均得到，更新公式如式(6)所示：

$$\boldsymbol{X}_i^{t+1}=\frac{1}{2}(\boldsymbol{Xold}_i^t+\boldsymbol{Xc}_i^t) \tag{6}$$

其中：$\boldsymbol{Xold}_i^t=\begin{cases}\boldsymbol{X}_i^t, & rand()<0.1\\ \boldsymbol{pBest}_i^t, & rand()\geqslant 0.1\end{cases}$，$\boldsymbol{Xc}_i^t=\boldsymbol{pBest}_1^t\ \&\ \boldsymbol{pBest}_2^t\ \&\cdots\&\ \boldsymbol{pBest}_s^t$，$\boldsymbol{X}\ \&\ \boldsymbol{Y}=\boldsymbol{X}\times\boldsymbol{Y}$表示只有同为1时结果为1。

2.2.3 幼狮位置更新规则

幼狮向下一个位置移动的方向由一个随机值$rand()$决定：若$rand()$<0.5，幼狮在狮王附近移动进食，下一个位置由当前位置和狮王位置共同决定；若$0.5\leqslant rand()<0.8$，幼狮跟随母狮学习捕猎，移动的位置由当前位置和所跟随母狮的历史最优位置决定，而所跟随母狮由幼狮编号与母狮数量求模的方法确定；否则幼狮将被逐出狮群，即幼狮向狮群当前最优位置的相反方向移动。幼狮位置更新如式(7)所示：

$$\boldsymbol{X}_i^{t+1}=\begin{cases}\boldsymbol{X}_i^t+\boldsymbol{gBest}^t, & rand()<0.5\\ \boldsymbol{X}_i^t+\boldsymbol{pBest}_i^t, & 0.5\leqslant rand()<0.8\\ \boldsymbol{X}_i^t+\overline{\boldsymbol{gBest}^t}, & rand()\geqslant 0.8\end{cases} \tag{7}$$

其中：$\overline{\boldsymbol{gBest}^t}$表示取反。

2.3 贪心算法修正处理

由于狮子的初始位置是随机产生的，在解空间会产生一部分不满足约束条件的无效解，若直接使用式(2)计算适应度，其结果必然会小于零，从而造成大量的无效试探。为了缩短寻优时间，可对任意一个潜在解使用贪心算法进行修正，使其转换为一个可行解。使用贪心算法修正会增加算法的计算量，但能够使搜索有一定的导向而不再盲目，反而能在总体上缩短处理时间。设第i个潜在解的位置为$\boldsymbol{X}_i=(x_{i1},x_{i2},\cdots,x_{ij},\cdots,x_{im})(1\leqslant i\leqslant N,1\leqslant j\leqslant m)$，贪心算法修正潜在解的实现步骤如下：

步骤1 对任意一个容器j，用式(1)检测输入值$\boldsymbol{X}_i$表示的物品体积是否超过C_j，若否，则直接转入步骤3。

步骤2 循环处理(总体积>C_j时)：

1)从袋中取出价值最小的物品x_{ij}；

2)实际总体积减去该项物品体积，置$x_{ij}=0$。

步骤3 循环处理(总体积<C_j时)：

1)从候选物品中选择价值最大的物品x_{ij}；

2)若该物品的体积加上袋内物品总体积>C_j则结束循环；

3)把该物品加入袋中，并更新总体积，置$x_{ij}=1$。

步骤4 返回修正后的$\boldsymbol{X}_i$。

步骤5 检测下一个容器，直到m个容器中的物品全部检测完成。

3 GBLSO算法

GBLSO算法步骤如下：

步骤1 初始化。设定狮群数目N，最大迭代次数T，维度空间m，母狮占狮群比例因子β，随机产生狮子的初始位置$\boldsymbol{X}_i$，贪婪因子Q_i。

步骤2 计算适应度值。根据适应度值排序狮王、母狮及幼狮的位置，适应度最佳的位置设为群最优位置，即狮王位置。

步骤3 判断终止条件：若达到终止条件，则输出问题最优解，即狮王的位置；否则，转步骤4。

步骤4 根据式(5)、(6)、(7)更新狮王、母狮及幼狮的

位置。

步骤5　根据狮子的位置计算适应度值,并对无效位置进行纠正处理。

步骤6　更新狮子自身历史最优位置及狮群历史最优位置。当有狮子发现最优解时,其他狮子调整自己的贪婪因子为自身因子和最佳因子的均值。

步骤7　每隔一定迭代次数(10次)重新排序狮王、母狮及幼狮的位置。

步骤8　计算适应度值,并判断算法是否达到终止条件,若达到则输出问题最优解,否则转步骤4。

4　仿真实验与分析

本文选取经典的离散二进制粒子群优化(Discrete binary Particle Swarm Optimization, DPSO)算法[11]和二进制蝙蝠算法(Binary Bat Algorithm, BBA)[12],并按照本文中的修复机制进行改进,和本文算法进行比较。本文算法的实验硬件环境为Intel Xeon E3-1230V2@3. 30 GHz和16 GB DDR3内存,操作系统为Win10,使用python3. 7,NumPy进行实验仿真。

4. 1　常用算例测试

为了验证二进制狮群群算法求解多维背包问题的有效性,本文选用参考文献[11]中的5类10个算例,每个算例的背包维度和物品数量如表1所示。

表1　选用算例测试表

Tab. 1　Test table of selected examples

背包维度	物品数量	测试算例	背包维度	物品数量	测试算例
低	少	weing1, weing2	中	多	pet5, pet7
低	多	weing7, weing8	高	多	sent01, sent02
中	少	pet2, pet3			

4. 2　算法参数设置

实验中种群规模N=50,最大迭代次数T=200。各算法的特定参数设置如表2所示。

表2　算法参数设置表

Tab. 2　Settings of parameters of different algorithms

算法	参数
GBLSO	母狮数量比例β=0. 1,最小贪婪度指数$Q_{\min}$=1,最大贪婪度指数$Q_{\max}$=12
BBA	初始音量A_i^0 = 0.25,初始脉冲发生率r_i^0 = 0.5,音量衰减系数α = 0.9,脉冲发生率系数γ = 0.9
DPSO	惯性权重因子ω= 0 . 729,学习因子c_1=c_2=1. 5

4. 3　仿真实验及结果分析

对每个算例独立运行20次,记录20次实验中获优次数、最优解、最差解、平均解和寻优成功迭代次数,如表3所示。

表3　三种算法性能比较

Tab. 3　Performance comparison of three algorithms

算例	物品-资源	参考最优解	算法	最优值	最差值	平均值	获优次数	寻优成功平均迭代数
weing1	28-2	141 278	GBLSO	141 278	141 258	141 276	20	88. 45
			BBA	141 278	140 618	141 208	20	111. 65
			DPSO	141 278	140 786	140 786	20	174. 25
weing2	28-2	130 883	GBLSO	130 883	130 723	130 859	20	86. 9
			BBA	130 883	130 723	130 819	20	121. 55
			DPSO	130 883	130 883	130 883	20	79. 55
weing7	105-2	1 095 445	GBLSO	1 095 382	1 094 785	1 095 142. 8	20	200
			BBA	1 095 382	1 095 206	1 095 371. 7	20	200
			DPSO	1 095 112	1 094 267	1 094 840. 65	20	200
weing8	105-2	624 319	GBLSO	623 459	621 086	622 164. 25	20	200
			BBA	623 612	621 086	621 686. 9	20	200
			DPSO	624 319	621 086	622 598. 3	20	184. 1
pet2	10-10	87 061	GBLSO	87 061	87 061	87 061	20	4. 3
			BBA	87 061	83 369	86 251. 4	9	8. 11
			DPSO	87 061	85 943	86 977. 1	19	38. 32
pet3	15-10	4 015	GBLSO	4 015	4 015	4 015	20	21. 45
			BBA	4 015	4 005	4 012	20	82. 25
			DPSO	4 015	3 940	3 998	17	153. 71
pet5	28-10	12 400	GBLSO	12 400	12 400	12 400	20	6. 05
			BBA	12 400	12 400	12 400	20	6. 1
			DPSO	12 400	12 070	12 382	19	24. 89
pet7	50-5	16 537	GBLSO	16 499	16 427	16 459. 25	20	200
			BBA	16 520	16 422	16 474	20	200
			DPSO	16 425	16 253	16 345	6	200
sent01	60-30	7 772	GBLSO	7 772	7 758	7 769. 5	20	84. 7
			BBA	7 772	7 758	7 766. 4	20	116. 5
			DPSO	7 772	7 690	7 737. 15	18	183. 94
sent02	60-30	8 722	GBLSO	8 722	8 721	8 721. 4	20	174. 3
			BBA	8 722	8 721	8 721. 5	20	136. 6
			DPSO	8 722	8 721	8 721. 1	20	190. 3

从表3的对比数据可以看出,本文算法在大多数算例上都要优于BBA和DPSO。本文算法每次迭代都能获得最优解,且寻优平均迭代数较参考文献算法少;说明本文算法收敛速度快,效率高。

图 1 是 GBLSO、BBA 和 DPSO 三种算法独立运行 20 次实验的平均寻优曲线。限于篇幅，仅给出三种算法对不同维数的背包问题部分算例的平均寻优曲线图。从图中可以看出，本文算法的平均寻优收敛速度快，能更好地找到全局最优解。

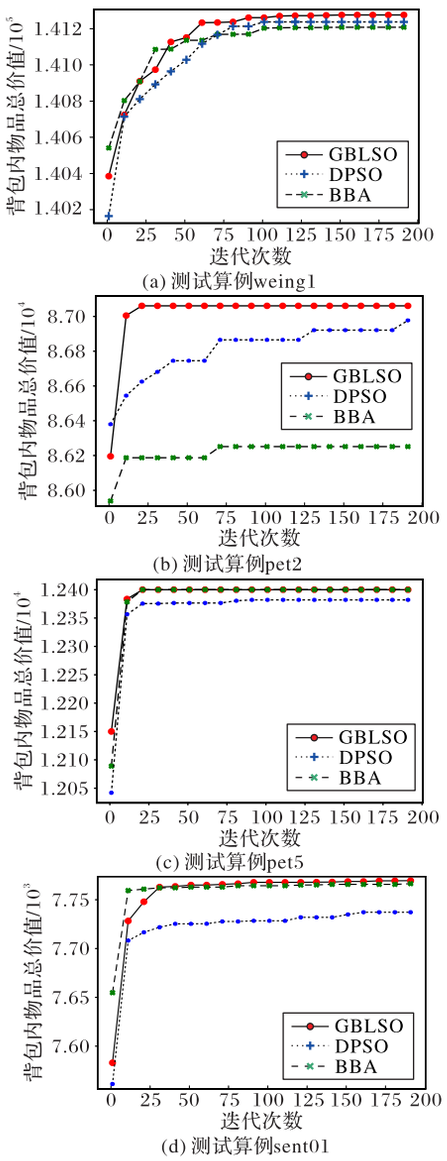

图 1 典型测试算例寻优曲线

Fig. 1 Optimization curves of typical test examples

5 结语

本文在狮群算法的基础上引入贪心算法的思想，利用二进制编码，提出了一种基于贪心算法的二进制狮群算法，并用于求解多维背包问题。通过 10 个经典多维背包算例的仿真对比实验表明本文算法具有较好的求解稳定性、收敛性和全局搜索能力，同时改进算法也为其他二进制的离散优化问题提供了一种有效的方法。

参考文献（References）

[1] 张晓霞，刘哲. 一种新的求解多维背包问题的分散算法[J]. 计算机应用研究，2012，29(5)：1716-1719.（ZHANG X X，LIU Z. New scatter search algorithm for multidimensional knapsack problem [J]. Application Research of Computers，2012，29(5)：1716-1719.）

[2] MIRJALILI S，LEWISA. S-shaped versus V-shaped transfer function for binary particle swarm optimization[J]. Swarm and Evolutionary Computation，2013，9:1-14.

[3] CHIL M，LIN C J，CHEERN M S，et al. Particle swarm optimization with time-varying acceleration coefficients for the multidimensional knapsack problem [J]. Applied Mathematical Modelling，2014，38(4):1338-1350.

[4] 钟培华，吴志远，缪建群. 区域分割粒子群算法及多维背包问题求解[J]. 计算机工程与应用，2011，47(36)：73-75.（ZHONG P H，WU Z Y，MIAO J Q. Partition particle optimization for multi-dimensional knapsack problem[J]. Computer Engineering and Applications，2011，47(36)：73-75.）

[5] 王会颖，倪志伟，吴昊. 求解多维背包问题的 MapReduce 蚁群优化算法[J]. 计算机工程，2013，39(4)：248-253.（WANG H Y，NI Z W，WU H. MapReduce-based ant colony optimization algrithm for multi-dimensional knapsack problem [J]. Computer Engineering，2013，39(4)：248-253.）

[6] 薛俊杰，王瑛，孟祥飞，等. 二进制反向学习烟花算法求解多维背包问题[J]. 系统工程与电子技术，2017，39(2)：451-458.（XUE J J，WANG Y，MENG X F，et al. Binary opposite backward learning fireworks algorithm for multidimensional knapsack problem[J]. Systems Engineering and Electronics，2017，39(2)：451-458.）

[7] 吴虎胜，张凤鸣，战仁军，等. 利用改进的二进制狼群算法求解多维背包问题[J]. 系统工程与电子技术，2015，37(5)：1084-1091.（WU H S，ZHANG F M，ZHAN R J，et al. Improved binary wolf pack algorithm for solving multidimensional knapsack problem [J]. Systems Engineering and Electronics，2015，37(5)：1084-1091.）

[8] 张晶，吴虎胜. 改进二进制布谷鸟搜索算法求解多维背包问题[J]. 计算机应用，2015，35(1)：183-188.（ZHANG J，WU H S. Modified binary cuckoo search algorithm for multidimensional knapsack problem[J]. Journal of Computer Applications，2015，35(1)：183-188.）

[9] 王小彤，侯立刚，苏成利. 一种改进的蚁群算法求解多维背包问题[J]. 辽宁石油化工大学学报，2015，35(4)：53-57.（WANG X T，HOU L G，SU C L. An improved ant colony algorithm solving multi-dimension knapsack problem [J]. Journal of Liaoning Shihua University，2015，35(4)：53-57.

[10] 刘生建，杨艳，周永权. 一种群体智能算法——狮群算法[J]. 模式识别与人工智能，2018，31(5)：431-441.（LIU S J，YANG Y，ZHOU Y Q. A swarm intelligence algorithm-lion swarm optimization [J]. Pattern Recognition and Artificial Intelligence，2018，31(5)：431-441.）

[11] KENNEDY J，EBERHART R C. A discrete binary version of the particle swarm algorithm[C]// Proceedings of the 1997 IEEE International Conference on Systemics，Man，and Cybernetics：Computational Cybernetics and Simulation. Piscataway：IEEE，1997：4104-4109.

[12] MIRJALILI S，MIRJALILI S M，YANG X S. Binary bat algorithm [J]. Neural Computing and Applications，2014，25(3/4)：663-681.

[13] WU H，ZHANG F，ZHAN R，et al. A binary wolf pack algorithm for solving 0-1 knapsack problem [J]. Systems Engineering and Electronics，2014，36(8)：1660-1667.

This work is partially supported by the Key Platform and Major Program of Higher Education Institutions of Guangdong (2018KQNCX392)，the Scientific Research Project of South China Institute of Software Engineering. GU (ky201823).

YANG Yan，born in 1985，M. S.，lecturer. Her research interests include computation intelligence，swarm intelligence algorithm.

LIU Shengjian，born in 1972，M. S.，lecturer. His research interests include swarm intelligence algorithm，deep learning.

ZHOU Yongquan，born in 1962，Ph. D.，professor. His research interests include computation intelligence，neural networks.