改进二进制人工蜂群算法求解多维背包问题

王志刚，夏慧明
（南京师范大学泰州学院数学科学与应用学院，江苏泰州 225300）

[摘要] 针对二进制人工蜂群算法收敛速度慢、易陷入局部最优的缺点，提出一种改进的二进制人工蜂群算法。新算法对人工蜂群算法中的邻域搜索公式进行了重新设计，并通过Bayes公式来决定食物源的取值概率。将改进后的算法应用于求解多维背包问题，在求解过程中利用贪婪算法对进化过程中的不可行解进行修复，对背包资源利用不足的可行解进行修正。通过对典型多维背包问题的仿真实验，表明了本文算法在解决多维背包问题上的可行性和有效性。

[关键词] 人工蜂群算法；多维背包问题；贪婪算法；组合优化
[中图分类号] TP301.6 [文献标识码] A [文章编号] 1009-1742(2014)08-0106-07

1 前言

人工蜂群[1~4](artificial bee colony，ABC)算法是由Karaboga于2005年提出的一种基于群体智能的仿生优化算法。实验表明，ABC算法比遗传算法(GA)、差分进化算法(DE)、粒子群优化算法(PSO)具有更好的优化性能。由于其具有收敛速度快、控制参数少、易于实现等优点，已引起越来越多的学者关注，并在很多优化问题中取得了成功的应用[5~9]。传统的ABC算法主要用于求解连续空间的优化问题，对于一些采用二进制编码的0～1整数规划问题却难以处理，这已成为限制其发展的一个瓶颈。为此，Marinakis等在2009年提出了一种二进制人工蜂群(binary artificial bee colony，BABC)算法[10]，然而其邻域搜索方式存在一些不足：a. 邻域搜索方式是随机的，导致算法的开采能力较弱，收敛速度较慢；b. 邻域搜索公式是单维的，使得候选食物源与原食物源极易相同，导致邻域搜索失败，影响算法性能。Pampará等[11]又提出了3种BABC算法，重点在于连续域到离散域的映射，但未研究新的邻域搜索方式。

在前人研究的基础上，本文提出一种改进的二进制人工蜂群(modified binary artificial bee colony，MBABC)算法，并将其应用于多维背包问题的求解。MBABC算法对ABC算法中引领蜂和跟随蜂的邻域搜索公式进行重新定义，并利用Bayes公式决定蜜蜂食物源的取值概率。在求解多维背包问题时，加入了贪婪算法这个有效的局部搜索方法，利用启发式信息，加快算法的收敛速度。通过对多维背包问题标准测试库的仿真实验和与其他算法的比较，表明了本文算法在迭代相同次数的情况下更容易找到最优解，体现了算法的可行性和高效性，为多维背包问题的求解提供了一种新的解决办法，拓展了ABC算法的应用领域。

2 人工蜂群算法

2.1 传统人工蜂群算法

ABC算法主要模拟蜜蜂种群的智能采蜜行

[收稿日期] 2013-11-11
[基金项目] 南京师范大学泰州学院资助项目(Q201232)
[作者简介] 王志刚，1978年出生，男，山东潍坊市人，讲师，主要研究方向为组合优化与算法研究；E-mail：wzg19.scut@163.com

为。蜂群主要由引领蜂(employed bees)、跟随蜂(onlookers)和侦察蜂(scouts)三个部分组成。人工蜂群算法在求解优化问题时,食物源代表优化问题的一个可能解,蜂群采蜜(食物源)的过程也就是搜寻优化问题最优解的过程。食物源的优劣取决于优化问题的适应值(函数值),适应值高的食物源较优。人工蜂群算法中解的个数(SN)等于引领蜂或跟随蜂的个数。用 $x_i=(x_{i1}, x_{i2}, \cdots, x_{iD})$ 表示第 i 个食物源($i=1, 2, \cdots, SN$,D 为搜索空间的维数)。首先,人工蜂群算法随机产生 SN 个解(食物源),然后蜂群对所有的食物源进行循环搜索,循环次数为 MCN 。引领蜂首先在食物源的邻域生成一个候选食物源,并比较候选食物源与先前食物源的优劣,如果候选食物源的适应值优于先前食物源的适应值,则用候选食物源代替先前的食物源,否则保持先前的食物源不变。结束之后,引领蜂回到舞蹈区把食物源优劣的信息通过跳摇摆舞传达给跟随蜂,跟随蜂根据所得到的信息按照一定概率选择食物源,适应值越高的食物源被选择的概率越大。跟随蜂选中食物源后,也在食物源的邻域生成一个候选食物源,并比较候选食物源与选中食物源的优劣,保留较优的食物源。人工蜂群算法就是通过上述反复循环来最终找到最优解的。当某个蜜蜂个体经过 $\lim it$ 次循环食物源没有更新时,个体放弃该食物源变成侦察蜂,寻找新的食物源。

引领蜂和跟随蜂根据式(1)在食物源的邻域生成一个候选食物源

$$v_{ij}=x_{ij}+r_{ij}(x_{ij}-x_{kj}) \tag{1}$$

式(1)中,v_{ij} 是生成的候选食物源;$k\in\{1, 2, \cdots, SN\}$,$j\in\{1, 2, \cdots, D\}$,这两个数都是随机选取的,但 $k\neq i$;r_{ij} 是 $[-1, 1]$ 上均匀分布的随机数,它控制 x_{ij} 邻域的生成范围。随着迭代次数的增加,$(x_{ij}-x_{kj})$ 之间的距离缩小,搜索的空间也缩小,即搜索的步长缩小,动态地调整步长,有助于算法提高精度,并最终获得最优解。式(1)称为ABC算法的邻域搜索公式。

跟随蜂通过如下概率来选择食物源

$$q_i=\frac{fit_i}{\sum_{n=1}^{SN} fit_n} \tag{2}$$

式(2)中,fit_i 为第 i 个食物源的适应值,即食物源的适应值越优,其被选择的概率就越大。在最大化问题中,fit_i 与优化问题目标函数值 f_i 的对应关系为

$$fit_i=\begin{cases}\dfrac{1}{1+f_i}, & f_i\geq 0\\ 1+|f_i|, & f_i<0\end{cases} \tag{3}$$

在ABC算法中,某个蜜蜂个体如果连续经过 $\lim it$ 次循环之后食物源仍然没有得到更新,个体就要放弃食物源,转变为侦察蜂。侦察蜂通过式(4)产生一个新的食物源

$$x_i^j=x_{\min}^j+\text{rand1}\,(x_{\max}^j-x_{\min}^j) \tag{4}$$

式(4)中,x_i^j 为新的食物源的第 j 维分量,$j\in\{1, 2, \cdots, D\}$;rand1 为 (0, 1) 之间均匀分布的随机数;$x_{\min}^j$、$x_{\max}^j$ 分别为第 j 维变量的最小值和最大值。

2.2 二进制人工蜂群算法

Marinakis 等人提出的BABC算法仿照二进制粒子群优化(BPSO)算法[12]和二进制差分进化(BDE)算法[13]的思想,将初始化得到的 x_i 和邻域搜索公式(1)得到的 v_i 通过logistic函数转换为二进制的 x_i' 和 v_i',转换公式为

$$\text{sig}(x_{ij})=\frac{1}{1+\exp(-x_{ij})} \tag{5}$$

$$x_i'=\begin{cases}1, \text{sig}(x_{ij})>\text{rand2}\\ 0, \text{sig}(x_{ij})\leq\text{rand2}\end{cases} \tag{6}$$

$$\text{sig}(v_{ij})=\frac{1}{1+\exp(-v_{ij})} \tag{7}$$

$$v_i'=\begin{cases}1, \text{sig}(v_{ij})>\text{rand3}\\ 0, \text{sig}(v_{ij})\leq\text{rand3}\end{cases} \tag{8}$$

式(6)、式(8)中,rand2 和 rand3 均为 (0, 1) 之间均匀分布的随机数。

算法的其他流程与传统的ABC算法相同,在此不再赘述。

3 改进二进制人工蜂群算法

BABC算法继续沿用传统ABC算法中的邻域搜索公式(1)得到的 v_i 转换成二进制的 v_i' 的必要性并不是很强,而且采用的logistic函数的主要功能是限界,并没用充分的理由。此外,在BABC算法中采用单维更新,这使得候选食物源与原食物源极易相同,导致邻域搜索失败。为此,本文对邻域搜索公式进行重新定义。首先,在生成候选食物源 v_{ij} 时,不采取随机选取一个 $j\in\{1, 2, \cdots, D\}$,而是使 j 取遍 1, 2, …, D,这样可以避免候选食物源与原食物源极易相同的缺陷;其次,从式(1)可以看出,v_{ij} 的取值主要由 x_{ij} 和 x_{kj} 决定,当 x_{ij} 和 x_{kj} 取0或1

后，可以通过设定 x_{ij} 和 x_{kj} 判断取0和1的可信度，然后借助Bayes公式来决定 v_{ij} 到底取0还是取1。因此，先作如下两个假定。

1）由于优化过程中 v_{ij} 的真值是未知的，故假定先验概率 $P(v_{ij}=0)=P(v_{ij}=1)=0.5$ 。

2）在寻找最优值的过程中，假定 x_{ij} 和 x_{kj} 对最优值的判定是相互独立的。

记 x_{ij} 作出正确判定的概率为 P_1 ，x_{kj} 作出正确判定的概率为 P_2 ，由Bayes公式可得

$$\begin{cases} P(v_{ij}=1|x_{ij}=1,\ x_{kj}=1)=\dfrac{P_1P_2}{P_1P_2+(1-P_1)(1-P_2)}=\alpha \\ P(v_{ij}=1|x_{ij}=0,\ x_{kj}=0)=\dfrac{(1-P_1)(1-P_2)}{P_1P_2+(1-P_1)(1-P_2)}=1-\alpha \\ P(v_{ij}=1|x_{ij}=1,\ x_{kj}=0)=\dfrac{P_1(1-P_2)}{P_1(1-P_2)+(1-P_1)P_2}=\beta \\ P(v_{ij}=1|x_{ij}=0,\ x_{kj}=1)=\dfrac{P_2(1-P_1)}{P_2(1-P_1)+P_1(1-P_2)}=1-\beta \end{cases} \tag{9}$$

x_{ij} 和 x_{kj} 发现最优值的概率应与优化问题的适应值成正比，因此不妨假设 $P_2=\dfrac{fit_k}{fit_i}P_1$ ，由于发生某个事件的概率应小于等于1，因此当 $P_2>1$ 时，就令 $P_2=1$ 。此外，由于 x_{ij} 和 x_{kj} 是迭代过程中的历史最好值，它们发现最优值的概率应随迭代次数的增加而逐渐变大，因此不妨令

$$P_1=(p_1-p_2)\frac{\text{iter}}{\text{MCN}}+p_2 \tag{10}$$

式(10)中，p_1、p_2 为 x_{ij} 发现最优值的概率的终值和初始值；iter为当前迭代次数；MCN为最大迭代次数。

4 求解多维背包问题的改进二进制人工蜂群算法

多维背包问题是组合优化领域中一个经典的NP难题，在很多实际工程问题中有着广泛的应用。因而，寻找可以有效解决该问题的算法具有重要的研究意义。目前，许多学者利用不同的思想提出了各种各样的算法来对其进行求解。贺一等[14]借鉴认知心理学有关记忆系统的表述，在禁忌搜索算法中引入长时记忆，构造了基于双禁忌表的禁忌搜索算法，在求解多维背包问题的仿真实验中取得了不错的效果。俞学才等[15]通过定义新的选择概率的规则和基于背包项的一种序的启发式信息，提出了求解多维背包问题的蚁群优化算法。孔民等[16]在蚁群优化系统高维立方体结构的基础上，提出一种二进制蚂蚁算法来求解多维背包问题。冀俊忠等[17]提出基于变异和信息素扩散的多维背包问题的蚁群算法。刘勇等[18]基于元胞自动机的原理和离散粒子群算法，提出一种元胞粒子群算法，该算法具有较好的全局优化能力。除了上述算法外，还有小世界算法[19]、DNA计算[20]、人工鱼群算法[21]等。

多维背包问题可描述为：有 n 个价值为 $w_j(j=1,\ 2,\ \cdots,\ n)$ 的物品，m 个容积为 $c_i(i=1,\ 2,\ \cdots,\ m)$ 的背包，第 j 个物品占用第 i 个背包的体积大小为 a_{ij} 。如何选择物品使其既可以装入这 m 个背包，又能使装入物品的总价值最大。其数学模型为

$$\begin{cases} \max f(x_1,\ x_2,\ \cdots,\ x_n)=\sum\limits_{j=1}^{n}x_jw_j \\ \text{s.t}\sum\limits_{j=1}^{n}x_ja_{ij}\leqslant c_i\ ,\quad i=1,\ 2,\ \cdots,\ m \\ x_j\in\{0,\ 1\},\ j=1,\ 2,\ \cdots,\ n \end{cases} \tag{11}$$

式(11)中，$f(x_1,\ x_2,\ \cdots x_n)$ 为目标函数；n 为物品数量；m 为背包数量；w_j 为第 j 个物品的价值；c_i 为第 i 个背包的体积；a_{ij} 为第 j 个物品占用第 i 个背包的体积大小；x_j 为0~1变量，$x_j=1$ 表示第 j 件物品被装入背包，$x_j=0$ 表示第 j 件物品没有被装入背包。

采用MBABC算法在求解多维背包问题时通过邻域搜索得到的解不一定可行，为此，引入贪婪算法来修正不可行解，同时在保证解可行的前提下，尽量增加其适应值。若解不可行，则按物品 j 的价值密度 $\rho_j=c_j/w_j\ (j=1,\ 2,\ \cdots,\ n)$ 由小到大的方向将 $x_j=1$ 变为 $x_j=0$ ，直到将不可行的解变成可行解；若解可行，则在保证解可行的前提下，按 ρ_j 由大到小的方向将 $x_j=0$ 变成 $x_j=1$ ，尽量增加其适应值。

求解多维背包问题的MBABC算法步骤如下。

1）设置迭代次数 iter ，群体规模 SN ，限定的循环次数 $\lim it$ ，最大迭代次数 MCN ，x_{ij} 发现最优值的概率的终值 p_1 和初始值 p_2 。

2）初始化种群 $X(\text{SN}\times n)$ ，对 X 中的每个向量 x_i（$i=1,\ 2,\ \cdots,\ \text{SN}$）用贪婪算法修正并计算适应值。

3）引领蜂按式(9)产生候选食物源，用贪婪算法修正并计算其适应值，如果候选食物源 v_i 的函数值优于原有食物源 x_i 的函数值，则用其替换 x_i ，否则保留 x_i 不变。

4）利用式(2)计算与 x_i 有关的概率值 q_i 。

5)跟随蜂根据 q_i 选择食物源，并按式(9)产生候选食物源，用贪婪算法修正计算其适应值，如果候选食物源 v_i 的函数值优于原有食物源 x_i 的函数值，则用其替换 x_i，否则保留 x_i 不变。

6)判断是否有要放弃的解，如果存在，则按照式(4)随机产生一个满足约束条件的新解来代替它。

7)记录迄今为止最好的解。

8)判断算法是否满足停止条件，如满足则输出最优结果，否则返回步骤3。

5 仿真实验

为了验证本文算法的性能，采用VC++编程，对文献[18]中的55个标准测试算例进行了仿真实验。限于篇幅，本文仅列出文献[18]和其他算法共同采用的较大规模背包问题的实验结果。表1为本文算法与文献[12]的BPSO算法和文献[18]的元胞粒子群算法(CPSO)以及文献[10]提出的BABC算法的实验结果对比，BPSO和CPSO的参数设置见文献[18]，BABC算法和本文算法中群体规模均为100(与BPSO和CPSO的群体规模一致)，$\lim it = 50$，在本文算法中的另外两个参数 p_1 和 p_2 分别取值为 $p_1 = 0.9$，$p_2 = 0.1$。仿真实验时，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、最劣解和平均解。

表1 BPSO、CPSO、BABC和MBABC的性能比较

Table 1 Comparison results of BPSO, CPSO, BABC and MBABC

测试数据				算法	最大迭代次数	获优次数	最优解	最劣解	平均解
名称	*m*	*n*	最优值						
Weing4	2	28	119 337	BPSO	80	10/20	119 337	115 831	118 110
				CPSO	80	20/20	119 337	119 337	119 337
				BABC	80	20/20	119 337	119 337	119 337
				MBABC	80	20/20	119 337	119 337	119 337
Weing6	2	28	130 623	BPSO	100	7/20	130 623	130 223	130 370
				CPSO	100	20/20	130 623	130 623	130 623
				BABC	100	10/20	130 623	130 233	130 428
				MBABC	100	16/20	130 623	130 233	130 543
Weing8	2	28	624 319	BPSO	500	0/20	596 790	561 228	579 930
				CPSO	500	15/20	624 319	606 029	623 220
				BABC	500	3//20	624 319	621 086	622 704.75
				MBABC	500	20/20	624 319	624 319	624 319
Weish09	5	40	5 246	BPSO	100	10/20	5 246	5 168	5 223.5
				CPSO	100	20/20	5 246	5 246	5 246
				BABC	100	20/20	5 246	5 246	5 246
				MBABC	100	20/20	5 246	5 246	5 246
Pet7	5	50	16 537	BPSO	100	0/20	16 458	16 188	16 324
				CPSO	100	18/20	16 537	16 465	16 533
				BABC	100	0/20	16 499	16 315	16 405.35
				MBABC	100	20/20	16 537	16 537	16 537
Weish11	5	50	5 643	BPSO	100	0/20	5 624	5 425	5 517.8
				CPSO	100	19/20	5 643	5 639	5 642.8
				BABC	100	20/20	5 643	5 643	5 643
				MBABC	100	20/20	5 643	5 643	5 643
Weish12	5	50	6 339	BPSO	100	0/20	6 318	6 072	6 240.7
				CPSO	100	20/20	6 339	6 339	6 339
				BABC	100	20/20	6 339	6 339	6 339
				MBABC	100	20/20	6 339	6 339	6 339
Weish13	5	50	6 159	BPSO	100	0/20	6 140	5 925	6 052.6
				CPSO	100	19/20	6 159	6 050	6 153.6
				BABC	100	20/20	6 159	6 159	6 159
				MBABC	100	20/20	6 159	6 159	6 159
Weish14	5	60	6 954	BPSO	150	0/20	6 923	6 732	6 819.9
				CPSO	150	18/20	6 954	6 923	6 950.9
				BABC	150	20/20	6 954	6 954	6 954
				MBABC	150	20/20	6 954	6 954	6 954
Weish15	5	60	7 486	BPSO	100	0/20	7 449	7 128	7 318.9
				CPSO	100	20/20	7 486	7 486	7 486
				BABC	100	9/20	7 486	7 442	7 467.1
				MBABC	100	20/20	7 486	7 486	7 486

续表

测试数据				算法	最大迭代次数	获优次数	最优解	最劣解	平均解
名称	m	n	最优值						
Weish16	5	60	7 289	BPSO	100	0/20	7 269	7 021	7 124.5
				CPSO	100	19/20	7 289	7 288	7 288.9
				BABC	100	4/20	7 289	7 281	7 287.35
				MBABC	100	19/20	7 289	7 288	7 288.9
Weish17	5	60	8 633	BPSO	150	0/20	8 618	8 519	8 577.6
				CPSO	150	19/20	8 633	8 624	8 632.55
				BABC	150	12/20	8 633	8 608	8 627.65
				MBABC	150	20/20	8 633	8 633	8 633
Weish18	5	70	9 580	BPSO	200	0/20	9 549	9 364	9 454.9
				CPSO	200	17/20	9 580	9 565	9 577.8
				BABC	200	0/20	9 573	9 518	9 556.7
				MBABC	200	8/20	9 580	9 573	9 575.8
Weish19	5	70	7 698	BPSO	200	0/20	7 683	7 340	7 546.4
				CPSO	200	17/20	7 698	7 683	7 695.8
				BABC	200	12/20	7 698	7 674	7 690.65
				MBABC	200	20/20	7 698	7 698	7 698
Weish20	5	70	9 450	BPSO	200	0/20	9 408	9 154	9 308.9
				CPSO	200	18/20	9 450	9 433	9 448.3
				BABC	200	3/20	9 450	9 418	9 436
				MBABC	200	12/20	9 450	9 433	9 447.4
Weish25	5	80	9 939	BPSO	200	0/20	9 801	9 553	9 678.5
				CPSO	200	16/20	9 939	9 923	9 937.1
				BABC	200	4/20	9 939	9 881	9 913.8
				MBABC	200	16/20	9 939	9 936	9 938
Weish26	5	90	9 584	BPSO	200	0/20	9 283	8 870	9 083.1
				CPSO	200	15/20	9 584	9 543	9 577.8
				BABC	200	4/20	9 584	9 527	9 574.6
				MBABC	200	17/20	9 584	9 578	9 583.4
Weish27	5	90	9 819	BPSO	200	0/20	9 551	9 071	9 307.3
				CPSO	200	20/20	9 819	9 819	9 819
				BABC	200	4/20	9 819	9 688	9 808.15
				MBABC	200	20/20	9 819	9 819	9 819
Weish28	5	90	9 492	BPSO	200	0/20	9 345	8 877	9 048.3
				CPSO	200	15/20	9 492	9 438	9 482.7
				BABC	200	20/20	9 492	9 492	9 492
				MBABC	200	20/20	9 492	9 492	9 492
Weish29	5	90	9 410	BPSO	200	0/20	9 287	8 720	9 026
				CPSO	200	19/20	9 410	9 394	9 409.2
				BABC	200	13/20	9 410	9 369	9 399.15
				MBABC	200	20/20	9 410	9 410	9 410
Weish30	5	90	11 191	BPSO	400	0/20	11 098	10 787	10 941
				CPSO	400	13/20	11 191	11 182	11 190
				BABC	400	0/20	11 182	11 144	11 162.3
				MBABC	400	20/20	11 191	11 191	11 191
Pet3	10	15	4 015	BPSO	30	14/20	4 015	4 005	4 012
				CPSO	30	20/20	4 015	4 015	4 015
				BABC	30	20/20	4 015	4 015	4 015
				MBABC	30	20/20	4 015	4 015	4 015
Pet5	10	28	12 400	BPSO	100	3/20	12 400	12 330	12 376
				CPSO	100	19/20	12 400	12 380	12 399
				BABC	100	20/20	12 400	12 400	12 400
				MBABC	100	20/20	12 400	12 400	12 400
Pb7	30	37	1 035	BPSO	100	0/20	1 034	936	1 009.5
				CPSO	100	19/20	1 035	1 034	1 034.95
				BABC	100	14/20	1 035	1 027	1 034.3
				MBABC	100	14/20	1 035	1 034	1 034.7
Pb6	30	40	776	BPSO	100	7/20	776	679	724.5
				CPSO	100	20/20	776	776	776
				BABC	100	13/20	776	765	772.15
				MBABC	100	20/20	776	776	776

从表1的对比数据可以看出，BPSO在很多情况下没有搜索到最优解或者获优次数很少，极易陷入局部最优而停滞不前；BABC凭借人工蜂群算法良好的优化性能，在最终优化结果上相比于BPSO有显著的改善。但由于BABC采用单维更新，这使得候选食物源与原食物源极易相同，容易导致邻域搜索失败，因而其求解效率不是很高。而MBABC克服了BABC容易陷入局部极值的缺陷，能以较快的速度达到全局最优，有较高的全局寻优性能。在对测试算例进行的仿真实验中要优于BABC以及BPSO和CPSO两种优化算法，表明了本文算法在解决多维背包问题上的可行性和有效性。

6 结语

二进制人工蜂群算法具有收敛速度慢、容易陷入局部最优等问题，针对这一不足，本文提出一种改进的二进制人工蜂群算法，并将其与贪婪算法相结合，应用于多维背包问题的求解。仿真实验结果表明，改进后算法的优化性能明显优于二进制人工蜂群算法和其他一些进化算法，为多维背包问题的求解提供了一个新的思路，并拓展了人工蜂群算法的应用范围。

参考文献

[1] Karaboga D. An idea based on honey bee swarm for numerical optimization [R]. Technical Report-TR06, Kayseri: Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[2] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm [J]. Journal of Global Optimization, 2007, 39(3): 459-471.

[3] Karaboga D, Basturk B. Artificial bee colony (ABC) optimization algorithm for solving constrained optimization [J]. Foundations of Fuzzy Logic and Soft Computing, 2007,4529: 789-798.

[4] Karaboga D, Basturk B. On the performance of artificial bee colony (ABC) algorithm [J]. Applied Soft Computing, 2008, 8(1): 687-697.

[5] Karaboga D. A new design method based on artificial bee colony algorithm for digital IIR filters [J]. Journal of the Franklin Institute, 2009, 346(4): 328-348.

[6] Singh A. An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem [J]. Applied Soft Computing, 2009, 9(2): 625-631.

[7] 胡中华，赵　敏. 基于人工蜂群算法的机器人路径规划[J]. 电焊机，2009，39(4)：93-96.

[8] 胡中华，赵　敏. 基于人工蜂群算法的TSP仿真[J]. 北京理工大学学报，2009，29(11)：978-982.

[9] 孙晓雅，林　焰. 改进的人工蜂群算法求解任务指派问题[J]. 微电子学与计算机，2012，29(1)：23-26.

[10] Marinakis Y, Marinaki M, Matsatsinis N. A hybrid discrete artificial bee colony-GRASP algorithm for clustering [C]//International Conference on Computers Industrial Engineering. Troyes, France: [s.n.], 2009: 548-553.

[11] Pampará G, Engelbrecht A P. Binary artificial bee colony optimization [C]//IEEE Symposium on Swarm Intelligence. Paris: IEEE, 2011:1-8.

[12] Kennedy J, Eberhart R C. A discrete binary version of the particle swarm algorithm [C]//Proceedings of IEEE Conference on Systems, Man, and Cybernetics. Orlando, 1997, 5: 4104-4108.

[13] Pampará G, Engelbrecht A P, Franken N. Binary differential evolution [C]//IEEE Congress on Evolutionary Computation. Vancouver: IEEE, 2006: 1873-1879.

[14] 贺　一，邱玉辉，刘光远，等.多维背包问题的禁忌搜索求解[J]. 计算机科学，2006，33(9) ：169-172.

[15] 喻学才，张田文. 多维背包问题的一个蚁群优化算法[J]. 计算机学报，2008，31(5)：810-819.

[16] 孔　民，田　澎，李相勇. 多维背包问题的二进制蚂蚁算法[J]. 管理科学学报，2009，12(2)：44-53.

[17] 冀俊忠，黄　振，刘椿年. 基于变异和信息素扩散的多维背包问题的蚁群算法[J]. 计算机研究与发展，2009，46(4)：644-654.

[18] 刘　勇，马　良. 元胞微粒群算法及其在多维背包问题中的应用[J]. 管理科学学报，2011，14(1)：86-96.

[19] 杜　巍，李树茁，陈煜聪. 一种求解多维背包问题的小世界算法[J]. 西安交通大学学报，2009，43(2)：10-14.

[20] 刘　毅，宋玉阶. 多维背包问题的DNA计算[J]. 生物数学学报，2008，23(1)：180-186.

[21] 李春梅，马　良. 求解多维0-1背包问题的人工鱼群算法[J]. 数学的实践与认识，2010，40(17)：195-199.

Modified binary artificial bee colony algorithm for multidimensional knapsack problem

Wang Zhigang, Xia Huiming
(School of Mathematics, Nanjing Normal University Taizhou College, Taizhou, Jiangsu 225300, China)

[**Abstract**] The binary artificial bee colony algorithm has the shortcomings of slower convergence speed and falling into local optimum easily. According to the defects, a modified binary artificial bee colony algorithm is proposed. The algorithm redesign neighborhood search formula in artificial bee colony algorithm, the probability of the food position depends on the Bayes formula. The modified algorithm was used for solving multidimensional knapsack problem. During the evolution process, it used the greedy algorithm to repair the infeasible solution and rectify feasible solution with insufficient use. The simulation results showed the feasibility and effectiveness of the proposed algorithm.

[**Key words**] artificial bee colony algorithm; multidimensional knapsack problem; greedy algorithm; combinatorial optimization