文章编号：2095-2163(2023)12-0098-05 中图分类号：TP301 文献标志码：A

求解多维背包的改进差分进化算法

韩丽萍，潘大志
（西华师范大学 数学与信息学院，四川 南充 637009）

摘 要：针对多维背包问题，提出了一种改进的差分进化(IDE)算法。该算法保留了基本差分进化算法的交叉策略，同时将特定维数的 0-1 变异融入其中；为提高算法的收敛性，设计了最大和最小可装入背包的物品数量模型，作为对后续操作产生解的一个条件判断，从而缩小了搜索范围及时间；最后，通过对 10 个背包测试集进行测试，并与贪心二进制狮群优化(GBLSO)算法、混合粒子群(HPSO)算法进行比较。结果表明，该算法能较好的求得最优解，具有更快的收敛速度及更高的精度。

关键词：多维背包问题；差分进化算法；组合优化；最大、最小装入背包物品数

Improved differential evolution algorithm for solving multidimensional knapsacks

HAN Liping, PAN Dazhi
(School of Mathematics and Information, China West Normal University, Nanchong Sichuan 637009, China)

Abstract: An improved differential evolution (IDE) algorithm is proposed for the multidimensional backpack problem. The algorithm retains the crossover strategy of the basic differential evolution algorithm while incorporating the 0-1 variant of a specific number of dimensions. In order to improve the convergence of the algorithm, a model of the maximum and minimum number of items that can be loaded into the knapsack is designed to serve as a conditional judgment on the subsequent operations to produce a solution, thus narrowing down the scope of the search and its time. Finally, the algorithm is tested by examining a test set of 10 knapsacks and compared to the Greedy Binary Lion Swarm Optimization (GBLSO) algorithm, and Hybrid Particle Swarm (HPSO) algorithm for comparison. The results show that the algorithm can find the optimal solution better, with faster convergence speed and higher accuracy.

Key words: multidimensional knapsack problem; differential evolution algorithm; combinatorial optimization; maximum and minimum number of knapsack items

0 引 言

多维背包问题[1](Multidimensional Knapsack Problem，MKP)是一个典型的 NP 难问题，属于带约束的组合优化问题，在实际生活中具有广泛的应用，如金融、工业、货物装载等方面。目前，精确式算法和启发式算法被用于解决此类问题。随着大规模问题的不断出现，精确算法已经不足以解决此类问题，而启发式算法可取得较好的解。其中包括基于 PSO 的求解算法[2-3]，分布估计算法[4]、遗传算法[5]等等。为更有效的求解背包问题，诸多研究人员对算法进行了改进。如：杨艳等[6]通过引入反置移动算子更新位置，利用贪心思想修复不可行解，提高了算法的收敛速度；刘雅文等[7]利用伯努利生成初始群体，且提出了一种多维加权价值密度对个体进行修复，有效提高了该算法的收敛速度；吴聪聪等[8]设计了一种有效的价值密度方式，并利用反向搜索策略来提高算法的搜索能力，使得算法的求解效果提高；YUAN 等[9]提出使用加权的编码方式，提高了算法的求解性能；张晶等[10]为利用种群多样性对鸟巢进行自变异，提出新的混合修复方式有效降低了早熟的概率；王凌等[11]通过建立一个概率模型产生新个体，使得算法的求解性能提高；杨广益等[12]基于分布估计凭借自然界互补的机制，提出松弛互补分

基金项目：国家自然科学基金(11871059)；四川省教育厅自然科学基金(18ZA0469)。
作者简介：韩丽萍(1998-)，女，硕士研究生，主要研究方向：组合优化、智能计算。
通讯作者：潘大志(1974-)，男，博士，教授，主要研究方向：智能计算、算法设计。Email：pdzzj@126.com
收稿日期：2022-12-18

布估计算法，提高了求解效率；Martins 等[13]根据效率组提出了启发式修复的随机策略，有效提高了所求解；薛俊杰等[14]通过离散化各个算子构成二进制烟花算法，并设计了一个不完全反向算子，可有效增强算法跳出局部最优的能力。

差分进化算法 DE[15]最初由 Storn 和 Price 提出，用于解决连续优化问题。进化算法是通过选择、交叉及突变来模拟个体的进化，DE 在连续空间的搜索能力良好，但存在易早熟、后期收敛速度慢等缺点。因此，本文提出了一种解决多维背包的改进差分进化算法（IDE）。在启发式算法的应用中，不可避免地会产生不可行的解，因此本文做出了以下改进：首先，在多约束条件下，随机生成可行的初始解；其次，为提高算法的收敛速度，提出了可装入背包的最大及最小物品数，即解的上下界，用于判断交叉产生的解，对不在此范围内的解则用当前的解替代，有效减少了解的搜索时间；为验证算法的有效性，测试了 10 组测试集，并与其他启发式算法进行最优值、最差值及平均值的比较，结果表明，此算法的求解精度较优。

1　多维背包问题

多维背包问题就是如何选择物品装入背包，并使得装入背包的物品价值之和最大，并且满足多个约束条件，其数学模型如下：

$$\max \sum_{j=1}^{n} a_j x_j \tag{1}$$

$$\text{s.t.} \sum_{j=1}^{n} w_{ij} x_j \leqslant c_i, i = 1,2,\cdots,m \tag{2}$$

$$x_j \in \{0,1\}, j = 1,2,\cdots,n \tag{3}$$

其中，n 为物品数量；m 为资源约束数量且 $m \geqslant 2$；a_j 为物品 j 的价值；c_i 为第 i 种资源的最大容量；w_{ij} 为物品 j 消耗第 i 种资源的大小。一般情况，a_j, c_i, w_{ij} 均为非负，$w_{ij} \leqslant c_i$ 且 $\sum_{j=1}^{n} w_{ij} > c_i$，$x_j = 1$ 表示物品 j 放入背包，$x_j = 0$ 表示物品 j 未放入背包。

2　差分进化算法

为解决多维背包问题，DE 算法选择采用 0-1 编码，该算法同 GA 算法一样，包含变异、交叉、选择等操作步骤。

（1）二进制编码

个体 $x_i = [x_{i1}, x_{i2}, \cdots, x_{in}]$ 为一个解，用 n 维二进制串表示，每个二进制位为 0 或 1。每个个体具有对应的概率向量，则 x_i 对应的概率向量可表示为 $p(x_i) = [p(x_{i1}), p(x_{i2}), \cdots, p(x_{in})]$，每个个体的概率向量在（0，1）之间随机产生，初始种群的产生则通过公式（4）产生：

$$x_i = \begin{cases} 1, p(x_i) > 0.5 \\ 0, p(x_i) \leqslant 0.5 \end{cases} \tag{4}$$

（2）变异

变异操作采用概率差分向量进行，变异公式如式（5）所示：

$$\boldsymbol{p}(u) = \boldsymbol{p}(\boldsymbol{r}_1) + F(\boldsymbol{p}(\boldsymbol{r}_2) - \boldsymbol{p}(\boldsymbol{r}_3)) \tag{5}$$

其中，u 为变异产生的个体；$\boldsymbol{p}(u)$ 为变异个体对应的概率向量；$\boldsymbol{p}(\boldsymbol{r}_1)$，$\boldsymbol{p}(\boldsymbol{r}_2)$，$\boldsymbol{p}(\boldsymbol{r}_3)$ 为随机选择的 3 个个体所对应的概率向量；F 为 [0,1] 间的变异因子，基于式（5）生成变异个体 u。

（3）交叉

交叉采用二项式交叉方式，生成交叉个体，交叉公式如式（6）所示：

$$v_i = \begin{cases} x_i, rand < cr \\ u_i, rand \geqslant cr \end{cases} \tag{6}$$

其中，v_i 为交叉个体，cr 为交叉概率。

（4）选择

将交叉后产生的个体与当前个体进行适应度比较，选择较优的个体进入下一代，选择方式如式（7）所示。

$$x_i = \begin{cases} x_i, fitness(v_i) < fitness(x_i) \\ v_i, \text{ otherwise} \end{cases} \tag{7}$$

其中，$fitness(x_i)$，$fitness(v_i)$ 分别表示个体 x_i，v_i 的适应度值。

3　差分进化算法的改进

基于传统的 DE，本文融入了 GA 的变异操作，并引入了特定维数变异及最大、最小可装入背包的物品数目的思想，用来提高算法的收敛速度及精度。

3.1　价值密度

由于 MKP 中物品的选取会受到多种资源的限制，为考虑物品在每一维的资源消耗[8]，故价值密度采用物品价值与物品在每一维的资源消耗占比之比如式（8）所示：

$$d_j = p_j / \sum_{i=1}^{m} (w_{ij} / c_i) \tag{8}$$

对于不可行解的修复和优化，采用贪心策略，对价值密度进行降序排列；对已放入背包的物品，依据价值密度排列从右至左逐一判断物品是否满足约束

条件,不满足则置为 0;对未放入背包的物品,依据价值密度排列从左至右判断,若满足则置为 1。

3.2 最小、最大可装入背包物品数

(1)计算最小可装入背包的物品数[1],数学模型如下:

$$k_{\min} = \min_i \left\{ k_i \,\middle|\, k_i = \sum_{j=1}^{n} x_j, i = 1,2,\cdots,m \right\} \quad (9)$$

$$\text{s.t.} \sum_{j=1}^{n} w_{ij} x_j \leqslant c_i, i = 1,2,\cdots,m \quad (10)$$

$$x_j \in \{0,1\}, j = 1,2,\cdots,n \quad (11)$$

通过对物品每一维的资源消耗量进行降序排列,依次选择装入背包的物品,在满足每一维资源约束的条件下,计算每一维约束下可装入背包的物品数量,最后计算出装入背包的最小物品数。

(2)计算最大可装入背包的物品数,数学模型如下:

$$k_{\max} = \max_i \left\{ k_i \,\middle|\, k_i = \sum_{j=1}^{n} x_j, i = 1,2,\cdots,m \right\} \quad (12)$$

$$\text{s.t.} \sum_{j=1}^{n} w_{ij} x_j \leqslant c_i, i = 1,2,\cdots,m \quad (13)$$

$$x_j \in \{0,1\}, j = 1,2,\cdots,n \quad (14)$$

通过对物品每一维的资源消耗量进行升序排列,依次选择装入背包的物品,在满足每一维资源约束的条件下,计算每一维约束限制下可装入背包的物品数,最后计算可装入背包的最大物品数。

针对大规模问题,在搜索解的过程中易出现搜索时间过长、搜索速度过慢等缺点,故引入最大、最小能够装入背包的物品数,即对解设置上下界,对于进行交叉操作产生的不在此范围内的解(可行或不可行),选择直接用对应的初始解(个体)替代,对于在此范围内的解,基于价值密度公式对其进行贪心修复优化操作,进而缩短了搜索时间。与原始算法相比,在收敛速度上也有一定效果。

3.3 特定维数变异

变异的过程采用 GA 中的变异策略,生成变异个体 u_i, 其操作如下:

$$u_{ij} = \begin{cases} 1 - x_{ij}, div < 0 \\ x_{ij}, div \geqslant 0 \end{cases} \quad (15)$$

其中, $div = fitness(i) - favg$, $fitness(i)$ 表示个体 i 的适应度值, $favg$ 表示种群的平均适应度值; $j \in d$,其 d 为 $\{1,2,\cdots,n\}$ 的随机数构成的集合。

3.4 IDE 算法求解 MKP

Step 1 初始化算法参数,设置最大迭代次数 T, 种群规模为 NP, 交叉概率 $cr = 0.5$, 最大、最小装入背包物品数 $k_{\max}, k_{\min}$;

Step 2 初始化生成全为可行解的初始种群,选择装入背包的物品数为 $k_{\min}$;

Step 3 基于公式(15)进行特定维数的 0-1 变异,对于适应度低于平均适应度的个体进行变异;

Step 4 基于公式(6)进行交叉,低于交叉概率则接受变异个体,否则接受当前个体;

Step 5 对交叉产生的解(个体)进行判断,对不在最小装入背包的物品数及最大物品数范围内的解,直接用当前个体替代,在此范围内的不可行解基于价值密度式(8)进行贪心修复优化,至满足约束条件;

Step 6 将修复优化后的个体与当前个体进行适应度比较,选择较优的个体进入下一代 ;

Step 7 满足迭代终止条件,则输出最优个体,否则转至 Step 3。

4 仿真实验与分析

为测试改进算法的有效性,对 10 个实例进行测试,并与贪心二进制狮群优化(GBLSO)算法[16]及混合粒子群算法(HPSO)算法[17]进行对比分析。

算法实例来自 ELIB 数据(http://elib.zib.de/pub/Packages/mp - testdata/ip/sac94 - suite/index.html),所有结果均由 Matlab R2021a 运行产生。

4.1 参数设置

为使实验结果进行有效对比,所有算法都设置相同的种群规模为 $NP = 30$, 最大迭代次数 $T = 10n$, n 为物品数量,变异维数为 $\lfloor n/5 \rfloor$, 算法的设置参数见表 1。

表 1 算法参数设置

Table 1 Settings of parameters of different algorithms

算法	参数
GBLSO	母狮数量比例 $\beta = 0.1$, 最小贪婪度指数 $Q_{\min} = 1$,
	最大贪婪度指数 $Q_{\max} = 12$
HPSO	惯性权重因子 $\omega = 1$, 学习因子 $c_1 = c_2 = 2$

4.2 实验结果分析

每个实例独立运行 20 次,记录各算法最优值、最差值及平均值,结果见表 2。

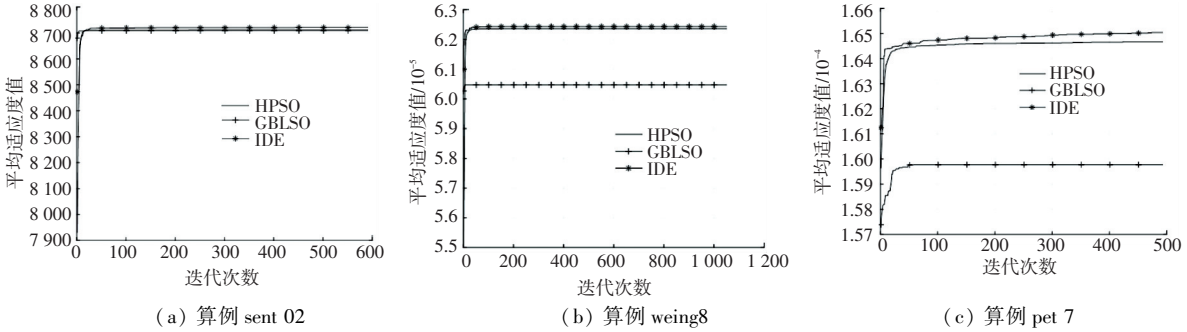
从表 2 可以看出,本文所提算法在大部分实例上均可达到最优解,且算法的求解精度较好。

IDE、GBLSO、HPSO 3 种算法部分算例独立运行 20 次的寻优曲线如图 1 所示。从图中可以看出,本文算法能较好的求得最优解。

表 2　算法结果对比

Table 2　Comparison of algorithm results

实例	物品-资源	参考最优解	算法	最优解	最差解	平均值
sent01	60-30	7 772	IDE	7 772	7 772	7 772
			GBLSO	7 772	7 707	7 764.4
			HPSO	7 772	7 761	7 771.4
sent02	60-30	8 722	IDE	8 722	8 713	8 721.1
			GBLSO	8 721	8 691	8 708.5
			HPSO	8 721	8 708	8 711.3
weing1	28-2	141 278	IDE	141 278	141 278	141 278
			GBLSO	141 278	141 278	141 278
			HPSO	141 278	140 778	141 252
weing2	28-2	130 883	IDE	130 883	130 883	130 883
			GBLSO	130 883	130 883	130 883
			HPSO	130 883	130 723	130 867
weing7	105-2	1 095 445	IDE	1 095 382	1 094 356	1 095 000
			GBLSO	1 093 780	1 091 460	1 091 600
			HPSO	1 095 382	1 094 356	1 094 900
weing8	105-2	624 319	IDE	624 319	624 319	624 319
			GBLSO	612 664	600 562	604 710
			HPSO	624 319	621 086	623 630
pet2	10-10	87 061	IDE	87 061	87 061	87 061
			GBLSO	87 061	87 061	87 061
			HPSO	87 061	87 061	87 061
pet3	15-10	4 015	IDE	4 015	4 015	4 015
			GBLSO	4 015	4 015	4 015
			HPSO	4 015	4 015	4 015
pet5	28-10	12 400	IDE	12 400	12 400	12 400
			GBLSO	12 400	12 380	12 392
			HPSO	12 400	12 400	12 400
pet7	50-5	16 537	IDE	16 537	16 496	16 504
			GBLSO	16 281	15 309	15 978
			HPSO	16 524	16 426	16 467



(a) 算例 sent 02　　(b) 算例 weing8　　(c) 算例 pet 7

图 1　测试算例寻优曲线

Fig. 1　Optimization curves of test examples

5　结束语

本文在差分进化算法的基础上，利用了与 GA 的相似性，将特定维数的 0-1 变异融入其中，并提出最大及最小装入背包的物品数模型，筛选出一些不在此范围内的解，并用对应的初始解替代，缩短了搜索时间，从而提高本文算法的收敛速度，较原始算法的收敛性有一定的效果。基于对 10 个算例的测试，表明了该算法的有效性以及良好的搜索能力。

（下转第 106 页）