文章编号: 1001-0920 (2016) 05-0817-06 DOI: 10.13195/j.kzyjc.2015.0491

贪婪封装二进制差分进化算法求解高维背包问题

钱淑渠[1,2], 叶永强[1], 武慧虹[2]

(1. 南京航空航天大学 自动化学院，南京 210016；2. 安顺学院 数理学院，贵州 安顺 561000)

摘 要: 提出一种处理高维背包问题 (KP) 的贪婪封装二进制差分进化算法 (GPBDE), 并设计了一种贪婪封装的修补策略处理不可行解. 为了提高种群的多样性及算法的全局搜索能力, 对适应度较低的个体执行对偶变换. 数值实验选取 4 种 KP 对 GPBDE 的优化能力进行测试, 并将所提出的算法与 4 种同类算法进行比较, 结果表明, GPBDE 具有较强的寻优和约束处理能力, 且收敛速度较快.

关键词: 背包问题；贪婪封装；约束处理；二进制；差分进化

中图分类号: TP306.01 **文献标志码:** A

Binary differential evolution algorithm with greedy packaging to solve high-dimensional knapsack problem

QIAN Shu-qu[1,2], *YE Yong-qiang*[1], *WU Hui-hong*[2]

(1. College of Automation Engineering，Nanjing University of Aeronautics and Astronautics，Nanjing 210016，China; 2. School of Sciences，Anshun University，Anshun 561000，China. Correspondent：QIAN Shu-qu，E-mail: shuquqian@163.com)

Abstract: A binary differential evolution algorithm with greedy packaging(GPBDE) is proposed to solve high-dimensional knapsack problems(KPs). A repair strategy with greedy packaging is developed to handle infeasible individuals. To improve population diversity and the ability of global search, a dual transformation is applied to a small number of individuals with lower fitness. To verify the optimization ability of GPBDE, four high-dimensional KPs are used in numerical experiments, and four peer algorithms are also compared with GPBDE. Experimental results show that the GPBDE is superior to other algorithms in finding excellent solutions and handling constraints, and exhibits a fast convergence speed.

Keywords: knapsack problem；greedy packaging；constraint handling；binary；differential evolution

0 引 言

背包问题 (KP) 属于一类经典的 NP 完全问题, 广泛应用于项目选址、货物装载、投资决策和资源优化等领域[1-2]. 近年来, KP 的求解算法备受学者关注, 出现了许多传统的规划算法和启发式智能优化算法[3]. 由于传统的分支定界法、动态规划法等经典算法的计算量和存储量较大, 最坏的时间复杂性可达到 $O(2^n)$, 且难以有效解决高维 KP. 启发式智能优化方法是一种群体随机搜索算法, 能克服传统算法计算复杂度高等缺点, 已成为求解 KP 的研究热点. Wang 等[4]提出了一种改进的混沌 GA 求解 KP, 但该算法仅适用于低维 KP, 且算法在执行较多代数时才能获得一定的优化效果. 文献 [5] 针对 GA 求解高维 KP 时收敛速度慢、易于陷入局部最优等缺陷提出了一种二进制克隆选择免疫 GA (CSIGA) 算法, 实验通过 4 种高维 KP 验证了所提出算法的约束处理能力; 文献 [2] 提出了一种改进的二进制粒子群 (MBPSO) 算法, 并与基本粒子群 (PSO) 算法进行了比较, 验证了 MBPSO 的有效性; 文献 [6] 提出了一种学习型和声搜索 (LHS) 算法, 通过自适应调整和声记忆考虑概率, 动态调节基音调整概率, 增强了算法的收敛速度; 文献 [7] 设计了一种基于 KP 的信息修复机制, 并结合一种自适应局部搜索策略, 提出了一种基于分布估计算法的混合算法求解多维 KP. 综上可知, 大量求解 KP 的智能算法已被提出. 然而, 差分进化 (DE) 是一种新型进化算法, 其结构简单、易于实现, 且实践验证了 DE 在解决复杂

收稿日期: 2015-04-21；**修回日期:** 2015-09-28.

基金项目: 国家自然科学基金项目(61304146, 61473145)；贵州省教育厅优秀科技创新人才奖励计划项目(黔教合KY字[2014]255)；贵州省科学技术基金项目(20152002).

作者简介: 钱淑渠(1978−), 男, 副教授, 博士生, 从事智能优化算法理论与应用、系统建模与控制等研究；叶永强(1974−), 男, 教授, 博士生导师, 从事电力电子控制等研究.

的非约束优化问题时具有全局搜索能力, 并具有与GA、PSO相当的收敛速度[8]. 改进的DE对KP的求解也得到了一些应用. 文献[9]提出了一种无参数变异的二进制DE(BDEPM)算法, BDEPM不采用压缩因子, 而是根据个体间的差异直接在离散域内进行突变, 保持了算法的封闭性; 文献[10]引入辅助搜索空间和个体混合编码策略, 借助一种满射将连续域空间映射到离散域空间, 提出了一种混合编码的二进制DE算法来解决KP; 文献[11]对每个决策变量采用实数和二进制同时编码, 利用DE算子对实数进行进化操作, 然后通过定义一种映射获得对应的0、1变量, 同时结合一种丢弃算子对不可行解进行修正, 提出了一种混合编码DE(MCDE)算法, 实验表明, 该算法优越于GA和传统DE算法; 文献[12]提出了一种压缩因子和交叉概率自适应变化的DE(JADE)算法, 并通过求解13个30维和100维的标准测试函数与两种自适应DE算法(jDE、SaDE)进行了比较, 比较结果验证了JADE的优越性, 然而其对KP的求解有待进一步验证.

已有的求解KP的DE算法多数是通过一种近似映射将连续空间映射到离散空间. 然而, 真正意义上实现离散空间DE算子进化的成果较少, 且对不可行个体的修正策略简单, 因此在解决KP时, 需要执行较大的评价次数才能获得一定数目的可行解. 为此, 本文基于基本DE框架, 设计一种新的修补策略和突变算子, 提出贪婪封装的二进制DE(GPBDE)算法. 数值实验将GPBDE应用于4种高维的KP, 并与同类算法BDEPM[9]、MCDE[11]、JADE[12]、MBPSO[2]进行了仿真比较, 结果表明, GPBDE较其他算法在求解KP的效果和收敛速度上均具有一定的优越性.

1 KP问题模型描述

KP可描述为: 有n件物品, w_i和p_i $(i=1,2,\cdots,n)$分别为物品i的重量和价值, 背包的最大容量限制为C, 在满足最大容量约束下, 如何将物品装入背包, 使得被装入背包的物品总价值达到最大. 定义变量$x_i\in\{0,1\}$为物品i是否被装入背包, x_i为1表示物品i被装入, x_i为0表示物品i未装入, 则任一候选向量$\boldsymbol{x}=(x_1,x_2,\cdots,x_n)$表示一种封装组合, 根据此组合即可得到背包内所装物品总重量$\sum\limits_{i=1}^{n}x_iw_i$, 以及被封装物品的总价值$\sum\limits_{i=1}^{n}p_ix_i$.

求解KP即如何确定最优的组合向量$\boldsymbol{x}$, 使其满足在一定约束下所装物品的总价值最大, 其数学模型可描述为

$$\begin{aligned}&\max\ \ f(\boldsymbol{x})=\sum_{i=1}^{n}p_ix_i;\\&\text{s.t. }\ g(\boldsymbol{x})=\sum_{i=1}^{n}x_iw_i-C\leqslant 0,\\&\boldsymbol{x}=(x_1,x_2,\cdots,x_n)^{\mathrm{T}}.\end{aligned}\tag{1}$$

其中: $x_i\in\{0,1\}$, $i=1,2,\cdots,n$.

2 DE算法基本算子

DE算法是一种基于个体差异重组的进化算法, 该算法在连续优化问题中得到了广泛的应用. DE算法结构简单、参数少, 易于实现和理解, 主要包括变异、交叉和选择. 为了便于表述, 假设当前代为g, 进化种群为A.

2.1 变 异

DE算法的变异策略较多[8], 在此仅给出DE/rand/1策略. 父体$\boldsymbol{x}_{i,g}\in A$按如下方式突变为子体$\boldsymbol{v}_{i,g+1}$:

$$\boldsymbol{v}_{i,g+1}=\boldsymbol{x}_{r_0,g}+F_i(\boldsymbol{x}_{r_1,g}-\boldsymbol{x}_{r_2,g}).\tag{2}$$

其中: $F_i\in(0,1]$为压缩因子; $\boldsymbol{x}_{r_0}$、$\boldsymbol{x}_{r_1}$和$\boldsymbol{x}_{r_2}$为3个不同的随机个体, 即$r_0\neq r_1\neq r_2$; $\boldsymbol{x}_{r_1,g}-\boldsymbol{x}_{r_2,g}$为差异向量.

2.2 交 叉

根据给定的交叉概率CR, 个体$\boldsymbol{v}_{i,g+1}$经交叉算子作用后得到新个体$\boldsymbol{u}_{i,g+1}$, 具体如下:

$$u_{ij,g+1}=\begin{cases}v_{ij,g+1},\ \text{rand}\leqslant\text{CR or }j=j\text{rand};\\x_{ij,g},\ \text{otherwise.}\end{cases}\tag{3}$$

其中: rand$\in[0,1]$为随机数, $j$rand$\in[1,n]$为随机整数, $u_{ij,g+1}$、$x_{ij,g}$和$v_{ij,g+1}$分别为$\boldsymbol{u}_{i,g+1}$、$\boldsymbol{x}_{i,g}$和$\boldsymbol{v}_{i,g+1}$的第j分量.

2.3 选 择

选择算子基于优胜劣汰的原则对交叉后的个体$\boldsymbol{u}_{i,g+1}$和父体$\boldsymbol{x}_{i,g}$根据适应度值进行选择, 具体如下:

$$\boldsymbol{x}_{i,g+1}=\begin{cases}\boldsymbol{u}_{i,g+1},\ \text{fit}(\boldsymbol{u}_{i,g+1})>\text{fit}(\boldsymbol{x}_{i,g});\\\boldsymbol{x}_{i,g},\ \text{otherwise.}\end{cases}\tag{4}$$

其中: fit$(\cdot)$为个体的适应度值, 且式(4)表明DE选择是一种贪婪选择.

3 二进制DE算法

KP为一类0-1整数规划问题. 求解KP的关键是约束处理策略的设计, 其是否恰当将极大地影响算法的收敛速度和所获解的质量. 另一方面, 利用传统的DE算法求解KP时易于失去种群的多样性, 以致陷入局部最优. 为此, 本文提出的二进制DE算法重点考虑以下3个方面.

3.1 变异算子

传统DE算法是针对连续变量的优化问题, 要使其求解0-1整数规划问题, 必须对变量进行处理. 已有

的处理方法主要为映射变换和取整. 映射变换法是利用满射将连续的辅助空间上的搜索过程转换成离散域 $\{0,1\}^n$ 上的搜索过程, 以达到优化的目的[10]; 取整法即将 [0,1] 间的连续变量进行四舍五入取整得到 0-1 整数变量[11,13]. 以上两类常用方法实质上进行的仍然是连续变量的搜索, 没有在真正意义上实现二进制 DE. 近年来, 对变量直接进行二进制编码的 DE 求解 KP 成为该领域的研究热点. 孔祥勇等[9]借助幂函数设计了如下无变异参数的二进制变异算子:

$$v_{ij,g+1} = x_{r_0j,g} + (-1)^{x_{r_0j,g}}|x_{r_1j,g} - x_{r_2j,g}|. \quad (5)$$

式 (5) 表明, j 维变量能否变异完全由差量 $d^j_{r_1r_2} = x_{r_1j,g} - x_{r_2j,g}$ 决定. 当 $|d^j_{r_1r_2}| = 1$ 时, $x_{r_0j,g}$ 发生变异; 当 $|d^j_{r_1r_2}| = 0$ 时, $x_{r_0j,g}$ 不发生变异.

该策略真正实现了二进制变异, 且结构简单, 无其他参数. 然而, 随着算法的进化, 群体中的个体趋于相似, 因此, 相同基因位上的基因也趋于相同, 差量 $|d^j_{r_1r_2}| = 0$ 的概率增大, 即该变异算子易于失去作用, 对算法的收敛速度有一定的影响. 为了克服该缺点, 本文提出如下变异策略:

$$v_{ij,g+1} = x_{r_0j,g} + (-1)^{\mathrm{sig}}|x_{r_1j,g} - x_{r_2j,g}|; \quad (6)$$

$$\mathrm{sig} = \begin{cases} 0, \ \mathrm{rand} < 0.5; \\ 1, \ \mathrm{otherwise}. \end{cases}$$

其中 $\mathrm{rand} \in [0,1]$ 为均匀分布随机数. 该策略以一定概率使 $x_{r_0j,g}$ 发生变异, 克服了式 (5) 在算法进化后期陷入局部搜索的缺点. 需要注意的是: 若突变后变量超出界, 则使用截断法处理, 即: 若 $v_{ij,g+1} < 0$, 则置 $v_{ij,g+1} = 0$; 若 $v_{ij,g+1} > 1$, 则置 $v_{ij,g+1} = 1$.

3.2　贪婪封装的修补策略

KP 的最大容量受一定限制, 使 KP 成为一类约束优化问题. 在优化领域, 约束的处理策略主要是罚函数法. 对 KP 直接使用罚函数法很难获得高质量解. 为此, 学者们提出了价值密度贪婪修补策略[9-11, 13-15]. 本文在这些策略的基础上, 提出一种贪婪封装的修补策略, 过程如下.

对于每个候选解 $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)^{\mathrm{T}} \in \{0,1\}^n$, 若 $\boldsymbol{x}$ 为非可行解, 则执行如下 Step 1 ~ Step 7, 否则执行 Step 4 ~ Step 7.

Step 1: 确定集合 $I = \{j | x_j = 1, j = 1, 2, \cdots, n\}$, 假设 $|I| = u$, 计算价值密度 $\rho_j = \dfrac{p_j}{w_j}, j \in I$;

Step 2: 对 I 按价值密度 ρ_j 升序排列, 得到序列集合 $\bar{I} = \{j_1, j_2, \cdots, j_u\}$;

Step 3: 按 $\bar{I}$ 的排序依次拿出物品, 直到背包内剩下物品的总重量不超过最大约束, 即 $g(\boldsymbol{x}) = \sum_{j=1}^{n} x_j w_j - C \leqslant 0$;

Step 4: 计算背包内剩余容量 $\triangle W = g(\boldsymbol{x})$;

Step 5: 确定集合 $O = \{j | x_j = 0, j = 1, 2, \cdots, n\}$, 假定 $|O| = \eta$, 计算价值密度 $\sigma_j = \dfrac{p_j}{w_j}, j \in O$;

Step 6: 对 O 按价值密度 σ_j 降序排列, 得到序列集合 $\bar{O} = \{j_1, j_2, \cdots, j_\eta\}$;

Step 7: 在确保背包内物品的总重量不超过最大约束的情况下, 按 $\bar{O}$ 的排序依次将其对应的物品装入背包.

注 1　Step 1 ~ Step 3 对非可行解进行修补, 使其为可行解; Step 4 ~ Step 7 对可行解及修复后的解采用贪婪封装策略, 加速算法的收敛速度.

3.3　对偶变换

对于高维 KP, 算法极易失去种群的多样性而使其停滞向可行域搜索. 为了克服此缺点, 每代对一定比例的适应度较低的个体采用对偶变换, 增加种群的多样性, 具体实施如下.

Step 1: 选择 $\lceil \alpha N \rceil$ 个适应度值较低的个体构成集合 P.

Step 2: 对于每个 $\boldsymbol{x} \in P$, 执行对偶变换为 $\boldsymbol{x}'$, 则

$$x'_j = \begin{cases} 1, \ \mathrm{if}\ (x_j = 0) \bigwedge (\mathrm{rand} < p_r); \\ 0, \ \mathrm{if}\ (x_j = 1) \bigwedge (\mathrm{rand} < p_r). \end{cases}$$

其中: $\alpha \in (0,1]$ 为选择率, $\lceil \cdot \rceil$ 为取上整数, $\mathrm{rand} \in (0,1]$ 为随机数, $p_r \in (0,1]$ 为对偶变换概率.

3.4　算法流程

根据以上设计, GPBDE 流程如图 1 所示.

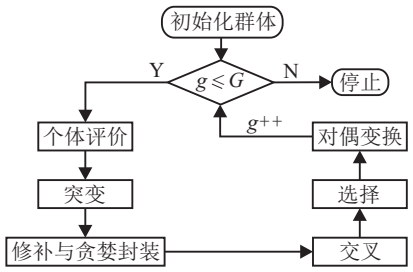

图 1　算法 GPBDE 流程

图 1 步骤可描述如下.

Step 1: 随机产生 N 个 n 维的 0-1 向量, 构成初始群 A, 置初始迭代数 $g = 1$;

Step 2: 判断是否达到最大代数 G, 若达到, 则停止, 否则执行 Step 3;

Step 3: 计算每个个体 $\boldsymbol{x} \in A$ 的适应度值

$$\mathrm{fit}(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}), \ g(\boldsymbol{x}) \leqslant 0; \\ f(\boldsymbol{x}) - 10^{10} g(\boldsymbol{x}), \ \mathrm{otherwise}; \end{cases}$$

Step 4: 对于每个 $\boldsymbol{x} \in A$, 采用 3.1 节突变算子进行突变, 获突变群 B;

Step 5: 采用 3.2 节的方法对每个个体执行贪婪封

装的修补策略, 获群体 C;

Step 6: 按 2.2 和 2.3 节的方式对 C 执行交叉 (交叉概率 $\mathrm{CR} = n^{-1}$) 和适应度值选择, 获得群体 D;

Step 7: 对群体 D 执行对偶变换, 获群体 E;

Step 8: 置 $g \leftarrow g+1$, $A \leftarrow E$, 转入 Step 2.

4 数值实验

4.1 实验设置

为了验证算法的性能, 将所提出的算法 GPBDE 与 BDEPM[9]、MCDE[11]、JADE[12]、MBPSO[2]进行对比, 分别应用于 50 维[10](KP-50)、60 维[5](KP-60)、100 维[10](KP-100) 和 200 维[5](KP-200) 的 KP, 以分析算法的性能.

通过 C++ 在计算机 (Intel(R)Core(TM) i5-4200 U, 500 G, 2.3 GHz) VC 平台上实现算法. 评价次数 (群体规模 × 最大迭代数) 设置为 100 × 100 和 400 × 400 两种情况. MBPSO 的参数设置如文献 [2], GPBDE 对偶变换中的选择率 $\alpha = 0.05$, 对偶变换概率 $p_r = 0.01$, 变异概率 $\mathrm{CR} = 1/n$, 其他 3 种改进的 DE 算法压缩因子和交叉概率按相应的文献设置, 如表 1 所示.

表 1 3 种算法的算子参数

参数	BDEPM	MCDE	JADE
F	无	0.5	自适应变化[12]
CR	0.9	0.5	自适应变化[12]

根据相应的参考文献 [5, 10], 4 个 KP 的背包最大容量和最优解如表 2 所示.

表 2 4 个 KP 的背包最大容量和最优值

参数	KP-50	KP-60	KP-100	KP-200
最大容量 C	1 000	600.2	4 995	1 918.8
最优值	3 119	1 563	5 375	5 164

4.2 实验结果及分析

为了减少随机性对算法性能评价的影响, 实验中各算法对每个问题独立执行 30 次, 根据所获得的 30 个结果分析其平均统计值, 并比较各算法的性能.

表 3 和表 4 为各算法在不同评价次数下对不同维数的 KP 独立执行 30 次所获得的统计结果. 在表 3 和表 4 中: 最好值、最差值指 30 次执行中所获得的最大目标值和最小目标值; 平均值指 30 个最好目标值的平均值; 标准差为 30 个最好目标值的标准差; 平均时间 (AvT) 为 30 次执行中每代的平均时间; 平均误差 (AvEr) 为

$$\mathrm{AvEr} = \sqrt{\sum_{k=1}^{30}(\mathrm{opt}_k - \mathrm{opt}_{\mathrm{best}})^2}. \tag{7}$$

其中: $\mathrm{opt}_{\mathrm{best}}$ 为最优值(见表 2), opt_k 为第 k 次执行所获得的最大目标值.

由表 3 可知: 评价次数为 10 000 时, 算法的最大迭代数为 100. 对于问题 KP-50, BDEPM 和 GPBDE 表现出了相同的优化效果; 对于问题 KP-60, BDEPM、MCDE、MBPSO 和 GPBDE 表现出相同的优化效果. 同时可以看出, MCDE 和 MBPSO 对问题 KP-60 的优化效果好于问题 KP-50, 这是由于问题 KP-60 的物品价值和重量均在 100 之内[5], 而问题 KP-50 的物品价值和重量比较大[10], 导致 KP-60 的优化难度低于 KP-50. 对于问题 KP-100 和 KP-200, GPBDE 表现出优越于其他 4 种算法的性能. 在这些算法中, JADE 表现出较差的效果, 这是由于 JADE 没有设置特定的修补策略, 其对 KP 的求解具有一定难度. 表 3 中"–"表示未获取可行解. 以上结果充分表明 GPBDE 对于高维 KP 表现出优于其他算法的性能.

表 3 评价次数为 10 000 时的统计结果

问题	算法	最好值	最差值	平均值	标准值	AvEr	AvT
KP-50	BDEPM	**3 119**	**3 119**	**3 119.00**	**0.00**	**0.00**	0.002
	MCDE	3 118	3 116	3 116.93	1.02	12.57	0.002
	JADE	3 112	2 838	2 900.80	61.28	1 239.8	0.002
	MBPSO	3 119	2 985	3 029.47	23.47	506.41	0.001
	GPBDE	**3 119**	**3 119**	**3 119.00**	**0.00**	**0.00**	0.002
KP-60	BDEPM	**1 563**	**1 563**	**1 563.00**	**0.00**	**0.00**	0.002
	MCDE	**1 563**	**1 563**	**1 563.00**	**0.00**	**0.00**	0.002
	JADE	1 563	869	1 029.37	178.28	3 076.5	0.002
	MBPSO	**1 563**	**1 563**	**1 563.00**	**0.00**	**0.00**	0.002
	GPBDE	**1 563**	**1 563**	**1 563.00**	**0.00**	**0.00**	0.002
KP-100	BDEPM	5 295	5 256	5 279.03	**10.37**	528.59	0.002
	MCDE	5 363	5 332	5 339.03	12.25	207.76	0.002
	JADE	5 288	5 063	5 103.87	53.01	1512.3	0.002
	MBPSO	5 353	5 305	5 324.97	12.28	281.91	0.002
	GPBDE	**5 371**	**5 332**	**5 351.73**	12.51	**144.14**	0.002
KP-200	BDEPM	5 127	5 061	5 089.23	22.26	426.70	0.002
	MCDE	5 125	5 079	5 099.63	15.25	361.99	0.002
	JADE	4 930	–	–	–	–	0.002
	MBPSO	5 085	4 999	5 042.07	25.86	682.22	0.002
	GPBDE	**5 164**	**5 143**	**5 156.20**	**4.61**	**49.42**	0.002

表 4 评价次数为 160 000 时的统计结果

问题	算法	最好值	最差值	平均值	标准值	AvEr	AvT
KP-50	BDEPM	**3 119**	**3 119**	**3 119.00**	**0.00**	**0.00**	0.004
	MCDE	3 119	3 117	3 118.80	0.48	2.83	0.004
	JADE	2 980	2 885	2 936.73	24.20	1 006.79	0.004
	MBPSO	3 118	2 988	3 027.23	25.38	520.88	0.005
	GPBDE	**3 119**	**3 119**	**3 119.00**	**0.00**	**0.00**	0.004
KP-60	BDEPM	**1 563**	**1 563**	**1 563.00**	**0.00**	**0.00**	0.004
	MCDE	**1 563**	**1 563**	**1 563.00**	**0.00**	**0.00**	0.004
	JADE	1 332	1 008	1 105.20	62.50	2 530.0	0.004
	MBPSO	**1 563**	**1 563**	**1 563.00**	**0.00**	**0.00**	0.004
	GPBDE	**1 563**	**1 563**	**1 563.00**	**0.00**	**0.00**	0.004
KP-100	BDEPM	5 349	5 309	5 326.23	10.63	273.17	0.004
	MCDE	5 370	5 350	5 354.00	7.43	121.78	0.004
	JADE	5 161	5 093	5 125.40	18.83	1 370.87	0.004
	MBPSO	5 356	5 305	5 321.47	10.98	299.11	0.004
	GPBDE	**5 375**	**5 372**	**5 374.70**	**0.84**	**4.80**	0.004
KP-200	BDEPM	**5 164**	**5 164**	**5 164.00**	**0.00**	**0.00**	0.004
	MCDE	**5 164**	**5 164**	**5 164.00**	**0.00**	**0.00**	0.004
	JADE	3 318	2 732	2 950.50	118.18	12 140.5	0.004
	MBPSO	5 090	4 988	5 034.00	25.32	724.98	0.004
	GPBDE	**5 164**	**5 164**	**5 164.00**	**0.00**	**0.00**	0.004

表 4 给出了评价次数为 160 000 时各算法所获得的统计结果, 当评价次数为 160 000 时, 群体规模和迭

代数均增大. 对于问题 KP-50, BDEPM 和 GPBDE 表现出了相同的优化效果; 对于问题 KP-60 和 KP-200, BDEPM、MCDE、GPBDE 表现出了相似的优化能力, MBPSO 对 KP-200 效果稍差, JADE 最差; 对于问题 KP-100, GPBDE 表现出最优性能. 同理, 由于 KP-100 物品的价值和重量均较大[10], 导致问题难度增大, 4 个算法在此问题上表现出较差的性能. 问题 KP-200 的物品价值和重量均在 100 内[5], 故即使其维数高于 KP-100, 在较大的评价次数下, BDEPM 和 MCDE 也能获得较好的优化效果. 由表 3 和表 4 中的 AvT 可知, 各算法在不同评价次数内的平均时间开销近似值相等, 这表明其计算复杂度较相似.

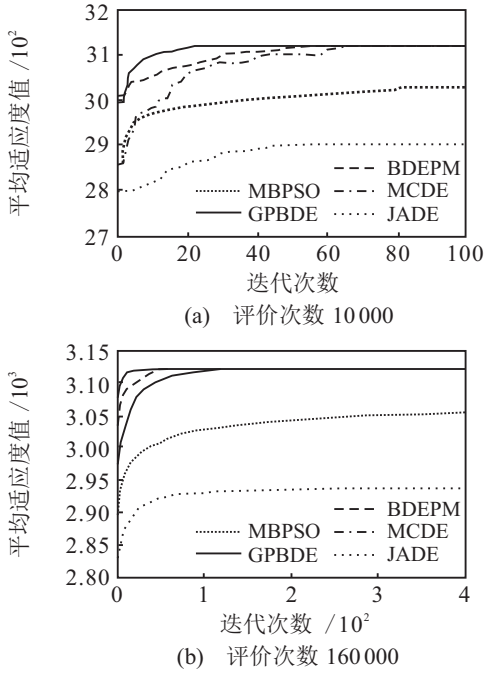

图 2　对于问题 KP-50 各算法的平均适应度值变化

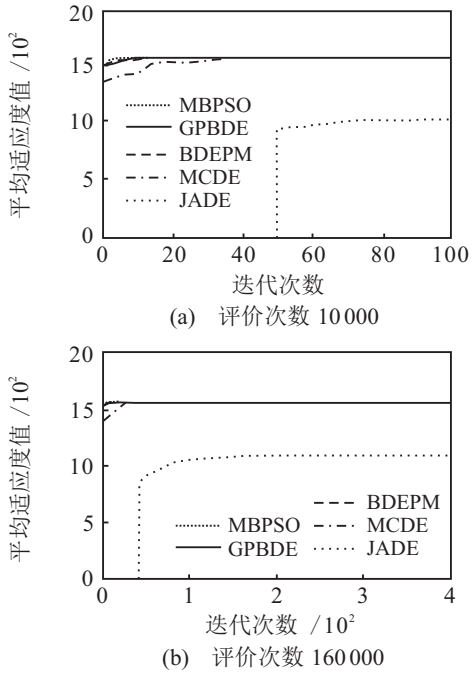

图 3　对于问题 KP-60 各算法的平均适应度值变化

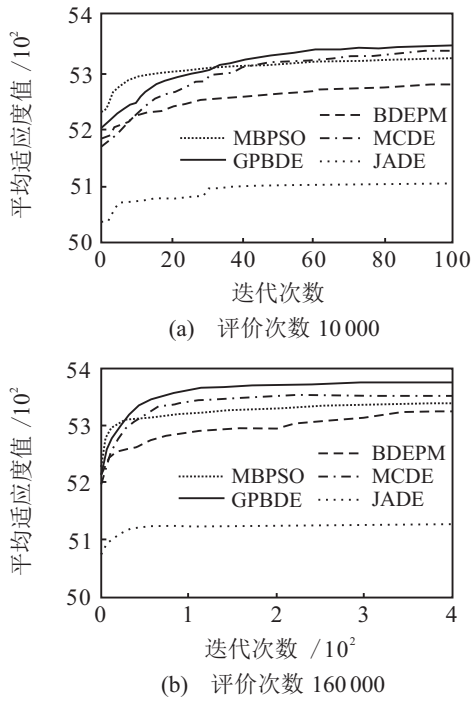

图 4　对于问题 KP-100 各算法的平均适应度值变化

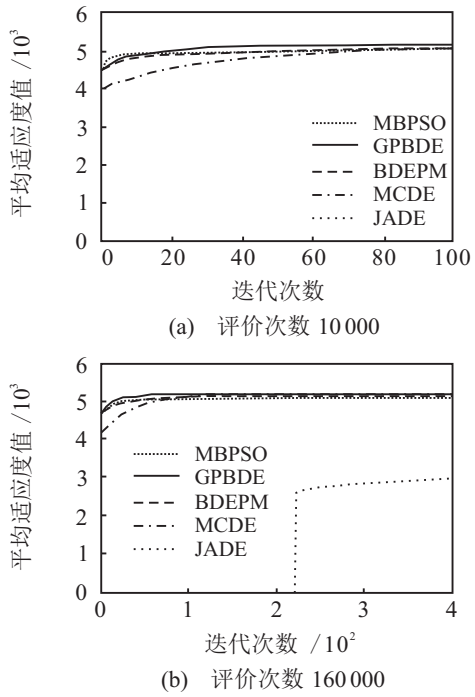

图 5　对于问题 KP-200 各算法的平均适应度值变化

图 2～图 5 分别为不同评价次数下各算法所获得的平均适应度值随迭代次数的变化曲线.

由图 2 可知: 对于问题 KP-50, GPBDE 收敛速度最快, 其他依次为 BDEPM、MCDE、MBPSO, 而 JADE 收敛速度最慢.

由图 3 可知: 对于问题 KP-60, GPBDE 和 BDEPM 表现出相同的收敛速度, JADE 在不同评价次数时, 约在 50 代才获得可行解 (图 3(a)).

由图 4 可知: 对于问题 KP-100, GPBDE 收敛速度最快, 其余依次为 MCDE、MBPSO、BDEPM、JADE.

由图5可知: 对于问题KP-200, 收敛速度快慢依次为GPBDE、MBPSO、BDEPM、MCDE, 而JADE在评价次数为10 000时仍没有获得可行解, 故此时未描绘出变化曲线.

通过以上分析可知: 本文提出的算法GPBDE在高维时表现出较好的优越性, 且收敛速度快于其他算法, 这是因为GPBDE的突变机制、对偶变换和约束处理策略极大提高了种群的多样性和可行解的质量.

5 结 论

本文基于贪婪封装的修补策略提出了一种二进制差分进化(GPBDE)算法, 用于高维背包问题的求解. 设计了一种无压缩因子的差分变异策略和对偶变换模块, 以及一种贪婪封装的修补策略, 提高了算法的收敛速度和约束处理能力. 数值实验将GPBDE用于4种高维背包问题, 通过不同的评价次数验证了所提出算法在求解高维背包问题上的有效性, 并与同类算法进行了仿真比较, 结果表明, 所提出的算法具有比其他算法更快的收敛速度. 本文算法对多维背包问题的应用将是下一步的研究内容.

参考文献(References)

[1] Gilmore P C, Gomory R E. The theory and computation of knapsack functions[J]. Operations Research, 1966, 14(6): 1045-1074.

[2] Bansal J C, Deep K. A modified binary particle swarm optimization for knapsack problems[J]. Applied Mathematics and Computation, 2012, 218(22): 11042-11061.

[3] Baykasolu A, Ozsoydan F B. An improved firefly algorithm for solving dynamic multidimensional knapsack problems[J]. Expert Systems with Applications, 2014, 41(8): 3712-3725.

[4] Wang R, Guo L. An improved quantum genetic algorithm with mutation and its application to 0-1 knapsack problem[C]. Proc of MIC'12. Harbin: IEEE Press, 2012: 484-488.

[5] 武慧虹, 钱淑渠, 徐志丹. 克隆选择免疫遗传算法对高维0/1背包问题应用[J]. 计算机应用, 2013, 33(3): 845-848.
(Wu H H, Qian S Q, Xu Z D. Immune genetic algorithm based on clonal selection and its application to 0/1 knapsack problem[J]. J of Computer Applications, 2013, 33(3): 845-848.)

[6] 李若平, 欧阳海滨, 高立群, 等. 学习型和声搜索算法及其在0-1背包问题中的应用[J]. 控制与决策, 2013, 28(2): 205-210.
(Li R P, Ouyang H B, Gao L Q, et al. Learned harmony search algorithm and its application to 0-1 knapsack problems[J]. Control and Decision, 2013, 28(2): 205-210.)

[7] 王凌, 王圣尧, 方晨. 一种求解多维背包问题的混合分布估计算法[J]. 控制与决策, 2011, 26(8): 1121-1125.
(Wang L, Wang S Y, Fang C. A hybrid distribution estimation algorithm for solving multidimensional knapsack problem[J]. Control and Decision, 2011, 26(8): 1121-1125.)

[8] Das S, Suganthan P N. Differential evolution: A survey of the state-of-the-art[J]. IEEE Trans on Evolutionary Computation, 2011, 15(1): 4-31.

[9] 孔祥勇, 高立群, 欧阳海滨, 等. 无参数变异的二进制差分进化算法[J]. 东北大学学报: 自然科学版, 2014, 35(4): 484-487.
(Kong X Y, Gao L Q, Ouyang H B, et al. Binary differential evolutionary algorithm based on parameterless mutation strategy[J]. J of Northeastern University: Natural Science, 2014, 35(4): 484-487.)

[10] 贺毅朝, 王熙照, 寇应展. 一种具有混合编码的二进制差分演化算法[J]. 计算机研究与发展, 2007, 44(9): 1476-1484.
(He Y C, Wang X Z, Kou Y Z. A binary differential evolution algorithm with hybrid encoding[J]. J of Computer Research and Development, 2007, 44(9): 1476-1484.)

[11] 邓长寿, 赵秉岩, 梁昌勇. 基于混合编码的差异演化算法解0-1背包问题[J]. 计算机应用研究, 2010, 27(6): 2031-2033.
(Deng C S, Zhao B Y, Liang C Y. Mixed-coding-based differential evolution algorithm for 0-1 knapsack problem[J]. Application Research of Computers, 2010, 27(6): 2031-2033.)

[12] Zhang J, Sanderson A C. JADE: Adaptive differential evolution with optional external archive[J]. IEEE Trans on Evolutionary Computation, 2008, 13(5): 945-958.

[13] 王丛佼, 王锡淮, 肖建梅. 具有参数自适应机制的改进离散差分进化算法[J]. 计算机科学, 2014, 41(1): 279-282.
(Wang C J, Wang X H, Xiao J M. Improved discrete differential evolution with parameter adaptive mechanism[J]. Computer Science, 2014, 41(1): 279-282.)

[14] Deng C, Zhao B, Yang Y, et al. Novel binary differential evolution algorithm for discrete optimization[C]. Proc of ICNC'09. Tianjin: IEEE Press, 2009: 346-349.

[15] Liang C, Deng A, Zhao B, et al. Hybrid-coding binary differential evolution algorithm with application to 0-1 knapsack problems[C]. Int Conf on Computer Science and Software Engineering. Wuhan: IEEE Press, 2008: 317-320.

(责任编辑: 闫 妍)