文章编号: 1000-5862(2012)04-0364-06

基于区域分割的差分进化算法求解 0-1 背包问题

钟培华，吴志远，胡建根，朱　丽

(江西农业大学理学院，江西 南昌 330045)

摘要：为了更有效地求解 0-1 背包问题，提出了基于区域分割的差分进化算法(PDE). 为保证变异算子的封闭性，对传统差分进化算法(DE)的变异算子进行了修改. 引入区域分割算法以后，解空间中一些没有希望的点被移除，缩小了最优解的搜索范围，增加了找到最优解的概率. 将区域分割和贪婪算法相结合，用搜索到的最好解替换了种群中目标函数值最差的个体，保证了种群的多样性. 数值实验表明：该算法比文献中的 DE 算法更稳健，全局搜索能力更强，能以更大的概率找到背包问题的最优解.

关键词：背包问题；差分进化算法；分割

中图分类号: TP 18　　文献标志码: A

0　引言

背包问题是运筹学中的 NP 难问题，有极其广泛的应用背景. 如选址问题、资金预算问题、投资决策等经济管理中的问题都可以建立背包问题的数学模型. 背包问题可以描述为：有一个最大承重为 b 的背包和 n 种物品，物品 j 的重量为 w_j，价值为 c_j，在背包承重允许条件下应如何选择一组物品装入背包，才能使装入背包的物品价值最大.

若用 $x_j=1$ 表示物品 j 装入背包，$x_j=0$ 表示物品 j 不装入背包，则背包问题可用下列 0-1 整数规划表示:

$$\max \quad f(x)=\sum_{j=1}^{n} c_j x_j ,$$

$$\text{s. t.} \quad g(x)=\sum_{j=1}^{n} w_j x_j \leqslant b ,$$

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

其中 x_j 称为决策变量，当 $x=(x_1,x_2,\cdots,x_n)\in\{0,1\}^n$ 且满足(1)式时，称为背包问题的可行解，否则，称 x 为不可行解.

背包问题的算法有精确算法和近似算法. 精确算法的优点是能求到问题的最优解，缺点是求解时间和内存消耗会随着问题规模的增加而急剧增加. 近似算法的优点是时间和内存消耗相对稳定，能在合理的时间内求到大规模问题的有效解，因而在背包问题中得到广泛应用. 其中近似算法主要有差分进化算法[1-3]、粒子群算法[4]、遗传算法[5]、蜂群算法[6]等. 这些算法经过多次运算能找到问题的最优解，但是找到最优解概率的大小因问题而异，也和问题的规模有关. 可见算法的稳健性有待于进一步提高. 由于差分进化算法具有原理简单、控制参数少、易于实现、全局搜索能力强等优点，它在连续空间的函数优化中得到广泛的应用[7-10]. 本文结合背包问题适合二进制编码的特点，在差分进化算法的框架下，对其变异算子进行了改进，保证了变异、选择、交叉等算子的封闭性. 根据背包问题的特性引入区域分割算法，提出了基于区域分割的差分进化算法. 对经典背包问题的测试表明，该算法找到最优解的概率更大，运行更稳健，所需的种群规模和迭代次数更少.

1　差分进化算法及其改进

差分进化算法(DE)是基于群体差异的演化算法，是在 1995 年由 Rainer Storn 和 Kennth Prince 提出的直接搜索算法. 它主要由变异、交叉、选择 3 个算子组成. 下面就以最大化问题为例，简单介绍 DE/best/1/bin 模式的差分进化算法.

收稿日期：2012-05-20
基金项目：江西省教育厅省级教改基金(JXJG-10-4-22)和江西农业大学青年基金(1574, 2971)资助项目.
作者简介：钟培华(1974-)，男，江西遂川人，讲师，硕士，主要从事最优化算法和数学建模研究.

设优化问题为

$$\max \quad f(x_1,x_2,\cdots,x_n),\ (x_1,x_2,\cdots,x_n)\in S,$$

其中 $S=\{(x_1,x_2,\cdots,x_n)\,|\,l_j\leqslant x_j\leqslant u_j,\ j=1,2,\cdots,n\}$，$l_j$ 和 u_j 为变量 x_j 的下界和上界. 在 DE 算法中，种群的个体对应 S 中的一个点. 若差分进化算法第 t 代的第 i 个个体表示为 $X_i^t=\left(x_{i1}^t,x_{i2}^t,\cdots,x_{in}^t\right)$ $(i=1,2,\cdots,N)$，N 称为种群规模，则 DE 算法的步骤如下.

Step 1 初始化：置 $t=0$，按(1)式产生初始种群的第 i 个个体 $X_i^0=\left(x_{i1}^0,x_{i2}^0,\cdots,x_{in}^0\right)$，其中 r_{ij} 是 [0,1] 上服从均匀分布的随机数.

$$x_{ij}^0=l_j+r_{ij}(u_j-l_j),\ i=1,2,\cdots,N,\ j=1,2,\cdots,n. \tag{1}$$

Step 2 变异操作：求出当前最优解 $Z=(z_1,z_2,\cdots,z_n)$，对种群中的每一个目标个体 X_i^t，按(2)式产生中间个体 $V_i^{t+1}=\left(v_{i1}^{t+1},v_{i2}^{t+1},\cdots,v_{in}^{t+1}\right)$，

$$v_{ij}^{t+1}=z_j+F\left(x_{r_1j}^t-x_{r_2j}^t\right), \tag{2}$$

其中 $F\in[0,2]$，互不相等的随机数 r_1，$r_2\in\{1,2,\cdots,N\}$.

Step 3 交叉操作：为增加种群多样性，将目标个体 X_i^t 和中间个体 V_i^{t+1} 按(3)式杂交产生实验个体 $U_i^{t+1}=\left(u_{i1}^{t+1},u_{i2}^{t+1},\cdots,u_{in}^{t+1}\right)$，其中 P_c 是交叉概率，r_{ij} 是 [0,1] 上服从均匀分布的随机数，均匀分布随机整数 $Rand(i)\in\{1,2,\cdots,n\}$，

$$u_{ij}^{t+1}=\begin{cases}v_{ij}^{t+1}, r_{ij}\leqslant P_c\text{或者 } j=Rand(i),\\ x_{ij}^t, \text{否则.}\end{cases} \tag{3}$$

Step 4 选择操作：按(4)式选择相应的个体进入下一代，

$$X_i^{t+1}=\begin{cases}U_i^{t+1}, f\left(U_i^{t+1}\right)\geqslant f\left(X_i^{t+1}\right),\\ X_i^{t+1}, \text{否则.}\end{cases} \tag{4}$$

Step 5 算法收敛性判断：若算法达到最大迭代次数，输出当前最优解作为最优解；否则，转 Step 2.

由于背包问题的决策变量 x_j 只取 0 和 1，因此，种群的个体更适合二进制编码. 但是，注意到按(2)式产生的中间个体将不再是二进制编码. 因此，须对变异操作进行改进. 为解决变异操作问题，文献[2]产生中间个体 $\hat{V}_i^{t+1}=\left(\hat{v}_{i1}^{t+1},\hat{v}_{i2}^{t+1},\cdots,\hat{v}_{in}^{t+1}\right)$ 方法是“少数服从多数”(见表 1). 即当 z_j，x_{r_1j} 和 x_{r_2j} 中有 2 个取值为 1(或 0)，则中间个体分量 $\hat{v}_{ij}^{t+1}$ 取 1(或 0)，否则 $\hat{v}_{ij}^{t+1}$ 取 0(或 1). 其数值实验表明，该方法是有效的. 本文保留“少数服从多数的思想”，提出一种分 2 步的、以概率变异的变异操作. 如表 1 所示，在第 1 步变异中取 $F=1$ 计算中间个体 V_i^{t+1}，其分量 v_{ij}^{t+1} 按(2)式计算. 显然，$v_{ij}^{t+1}\in\{-1,0,1,2\}$. 由于第 1 步变异已经破坏二进制编码结构，为了保证变异算子的封闭性，须进行第 2 步变异，以便将 v_{ij}^{t+1} 映射到 $\{0,1\}$ 上去，从而产生二进制的临时中间个体 $Q_i^{t+1}=\left(q_{i1}^{t+1},q_{i2}^{t+1},\cdots,q_{in}^{t+1}\right)$. 第 2 步变异的操作是：若第 1 步变异后 $v_{ij}^{t+1}=1$，则 q_{ij}^{t+1} 以概率 0.333 3 取 1，以概率为 0.666 7 取 0. 类似地，若第 1 步变异后 $v_{ij}^{t+1}=0$，则 q_{ij}^{t+1} 以概率 0.666 7 取 1，以概率为 0.333 3 取 0. 若 $v_{ij}^{t+1}=2$，则 q_{ij}^{t+1} 以概率 1 取 1. 若第 1 步变异后 $v_{ij}^{t+1}=-1$，则 q_{ij}^{t+1} 以概率 1 取 0.

从表 1 可知，本文分 2 步、依概率变异的操作本质上还是按“少数服从多数”的变异操作，而且更能保持种群多样性.

表 1　临时中间个体分量的取值概率

$\left(z_j,x_{r_1j},x_{r_2j}\right)$ 取值的情形	“少数服从多数”原则 $\hat{v}_{ij}^{t+1}$ 取值	第 1 步变异：按(2)式计算 v_{ij}^{t+1}	第 2 步变异：已知 v_{ij}^{t+1} 取值条件下临时中间个体分量 q_{ij}^{t+1} 的取值概率
(1,0,0)	0	1	$P\left(q_{ij}^{t+1}=1\right)=P\left(\hat{v}_{ij}^{t+1}=1\middle\|v_{ij}^{t+1}=1\right)=0.333\,3$
(0,1,0)	0	1	
(1,1,1)	1	1	$P\left(q_{ij}^{t+1}=0\right)=P\left(\hat{v}_{ij}^{t+1}=0\middle\|v_{ij}^{t+1}=1\right)=0.666\,7$
(1,0,1)	1	0	$P\left(q_{ij}^{t+1}=1\right)=P\left(\hat{v}_{ij}^{t+1}=1\middle\|v_{ij}^{t+1}=0\right)=0.666\,7$
(0,0,0)	0	0	
(0,1,1)	1	0	$P\left(q_{ij}^{t+1}=0\right)=P\left(\hat{v}_{ij}^{t+1}=0\middle\|v_{ij}^{t+1}=0\right)=0.333\,3$
(1,1,0)	1	2	$P\left(q_{ij}^{t+1}=1\right)=P\left(\hat{v}_{ij}^{t+1}=1\middle\|v_{ij}^{t+1}=2\right)=1$
(0,0,1)	0	–1	$P\left(q_{ij}^{t+1}=0\right)=P\left(\hat{v}_{ij}^{t+1}=0\middle\|v_{ij}^{t+1}=-1\right)=1$

交叉操作相应地改为用临时中间个体 Q_i^{t+1} 按(5)式交叉产生实验个体 U_i^{t+1}，其中 P_c 是交叉概率. 然后，按(6)式选择个体进入下一代种群，

$$u_{ij}^{t+1}=\begin{cases}q_{ij}^{t+1}, r_{ij}\leqslant P_c \text{ 或者 } j=Rand(i),\\ x_{ij}^{t}, \text{否则}.\end{cases} \tag{5}$$

2 区域分割算法

为了增加找到最优解的概率，本文引进区域分割算法，将一些没有希望的点移除，缩小最优解的搜索范围，并用搜索到的最好可行解替换种群中适应值最差的点.

设 $l=(l_1,l_2,\cdots,l_n)$，$u=(u_1,u_2,\cdots,u_n)\in\mathbf{Z}^n$，其中 l_j，$u_j\in\mathbf{Z}$，且 $l_j\leqslant u_j$ $(j=1,2,\cdots,n)$. 称

$$\langle l,u\rangle=\{(x_1,x_2,\cdots,x_n)\,|\,l_j\leqslant x_j\leqslant u_j,$$
$$x_j\in\mathbf{Z}, j=1,2,\cdots,n\}=\prod_{j=1}^{n}\langle l_j,u_j\rangle$$

为 $\mathbf{Z}^n$ 上的超长方体整数箱子. l 和 u 分别称为箱子的下界和上界. 若$\exists\, l_j>u_j$，则称 $\langle l,u\rangle$ 为空箱子. 如果 $\alpha=(\alpha_1,\alpha_2,\cdots,\alpha_n)$，$\beta=(\beta_1,\beta_2,\cdots,\beta_n)\in\mathbf{Z}^n$，且满足 $l\leqslant\alpha\leqslant\beta\leqslant u$，则称 $\langle\alpha,\beta\rangle$ 是 $\langle l,u\rangle$ 中的子箱子，其中 $l\leqslant\alpha\leqslant\beta\leqslant u$ 是指 $l_j\leqslant\alpha_j\leqslant\beta_j\leqslant u_j$ $(j=1,2,\cdots,n)$ 成立. 特别地，$\langle l,\alpha\rangle$ 和 $\langle\beta,u\rangle$ 也是 $\langle l,u\rangle$ 中的子箱子. 从 $\langle l,u\rangle$ 中移除 $\langle l,\alpha\rangle$ 和 $\langle\beta,u\rangle$ 中的整数点可以借助(6)式和(7)式实现，且剩下的整数点仍然可以表示成互不相交的超长方体箱子(不超过 $2n$ 个). 证明见文献[11]. 称(6)式和(7)式为区域分割算法，

$$\langle l,u\rangle\backslash\langle l,\alpha\rangle=$$
$$\bigcup_{j=1}^{n}\left(\prod_{k=1}^{j-1}\langle l_k,\alpha_k\rangle\times\langle\alpha_j+1,u_j\rangle\times\prod_{k=j+1}^{n}\langle l_k,u_k\rangle\right), \tag{6}$$
$$\langle l,u\rangle\backslash\langle\beta,u\rangle=$$
$$\bigcup_{j=1}^{n}\left(\prod_{k=1}^{j-1}\langle\beta_k,u_k\rangle\times\langle l_j,\beta_j-1\rangle\times\prod_{k=j+1}^{n}\langle l_k,u_k\rangle\right). \tag{7}$$

设 $l=(0,0,\cdots,0)$，$u=(1,1,\cdots,1)$，$x,y\in\{0,1\}^n=\langle l,u\rangle$，且 x 是背包问题的可行解，y 是背包问题的不可行解. 由于背包问题的目标函数 $f(x)$ 和约束函数 $g(x)$ 都是单调增加的，不难理解，$\forall\, z\in\langle l,x\rangle$ 都是背包问题的可行解. $\forall\, z\in\langle y,u\rangle$ 都是背包问题的不可行解. 因此，可以借助区域分割算法将子箱子 $\langle l,x\rangle$ 和 $\langle y,u\rangle$ 从 $\langle l,u\rangle$ 中移除，且不会遗失比 x 更好的可行解. 然后在更小的范围内搜索最优解，就可以增加找到最优解的概率.

3 基于区域分割的差分进化算法

由差分进化算法的变异和交叉算子产生的实验个体可能是可行解，也可能是不可行解. 为尽快找到最优解，须将其中的可行解改进为充分利用背包承重的、更好的可行解，同时将不可行解修复成可行解.

3.1 对可行解的改进算法

设 $x=(x_1,x_2,\cdots,x_n)$ 是背包问题的可行解，为了充分利用背包的承重，尽快找到最优解. 对 x 进行改进的方法是：对未装入背包的物品按价值密度(p_j 与 w_j 的比值)从大到小排序. 首先，考虑其中价值密度最大的物品，如果该物品未装入背包；且装入背包不会超重，则将该物品装入背包. 否则，该物品不装入背包. 然后，类似考虑其中价值密度第 2 大的物品，直到未装入背包的物品都不能装入背包为止. 设将 x 改进后得到的可行解为 $x^*=(x_1^*,x_2^*,\cdots,x_n^*)$，则 x^* 是充分利用背包承重的可行解. 若 x^* 中存在分量 x_j 满足 $x_j^*=0$，则 $\hat{x}^*=(x_1^*,\cdots,x_j^*+1,\cdots,x_n^*)$ 一定是不可行解.

3.2 修复不可行解的算法

设 $y=(y_1,y_2,\cdots,y_n)$ 是不可行解，对其进行修复的过程分 2 步执行，第 1 步对已装入背包的物品按价值密度从小到大排序，将背包中价值密度最小的物品取出，如果背包中的物品仍然超重，再将背包中价值密度最小的物品取出，直到背包中的物品不超重为止. 这样将得到 1 个可行解. 第 2 步对第 1 步得到的可行解用改进可行解的算法对其进行改进.

3.3 基于区域分割的差分进化算法

为便于叙述，以下假设 $l_j=0$，$u_j=1$ $(j=1,2,\cdots,n)$，即 $\langle l,u\rangle=\{0,1\}^n$.

Step 1 初始化：随机产生含 N 个二进制个体的种群，将其中的可行个体改进，将不可行个体修复. 求出当前最优解 $X^g=(x_1^g,x_2^g,\cdots,x_n^g)$. 设置最大的允许迭代次数，设置交叉概率，令进化代数 $t=0$.

Step 2 第 1 步变异：令 $F=1$，按(2)式计算中间

个体 V_i^{t+1} $(i=1,2,\cdots,N)$.

Step 3 第 2 步变异：根据 V_i^{t+1} 以概率计算临时中间个体 Q_i^{t+1}.

Step 4 按(5)式交叉产生实验个体 U_i^{t+1} $(i=1,2,\cdots,N)$，对其中的不可行解进行修复，对可行解进行改进.

step 5 按(4)式选择种群的个体和实验个体进入下一代新种群. 如果新种群存在个体比当前最优解更好，则更新当前最优解，令迭代次数 $t=t+1$.

Step 6 判断当前最优解是否已经连续迭代若干代都未得到更新，若是，转 Step 7；否则，转 Step 2.

Step 7 判断是否达到最大迭代次数，若是，转 Step 10；否则，转 Step 8.

Step 8 由于 X^g 是充分利用背包承重的可行解，随机选一个取值为 0 的变量($x_j^g=0$)，将其取值改为 1，则得到不可行解 $\hat{X}=(x_1^g,\cdots,x_j^g+1,\cdots,x_n^g)$. 借助(8)式和(7)式从 $\langle l,u\rangle$ 中先移除子箱子 $\langle l,X^g\rangle$，再移除 $\langle \hat{X},u\rangle$，丢弃其中下界不可行的箱子和空箱子后，设剩余的 K 个超长方体箱子表示为 $\langle S_i,T_i\rangle$ $(i=1,2,\cdots,K)$.

Step 9 若 $K=0$，转 Step 10；否则，第 i 个箱子的下界 $S_i=(s_{i1},s_{i2},\cdots,s_{in})$ 是可行解. 将其改进成充分利用背包问题的可行解. 第 i 个箱子的上界 $T_i=(t_{i1},t_{i2},\cdots,t_{in})$ 可能是可行解，也可能是不可行解. 将其中的可行解改进为充分利用背包承重的可行解，并将不可行解修复. 这样即可得到 $2K$ 个可行解，用其中最好的可行解替换种群中适应值最差的个体，当该可行解比当前最优解好时，更新当前最优解，转 Step 2.

Step 10 算法结束，输出当前最优解.

4 数值实验

为了测试本文算法的性能，对来源于文献的 7 个经典算例和 1 个随机产生的大规模算例进行了 2 组数值测试. 其中随机产生的 150 维背包问题 KP8 数据为：问题规模 n=150，背包最大承重 b=2 116，物品价值系数 C={325, 44, 47, 507, 260, 43, 208, 72, 176, 312, 219, 159, 19, 457, 18, 220, 130, 39, 202, 151, 221, 458, 95, 148, 400, 445, 292, 302, 47, 410, 325, 272, 125, 193, 10, 41, 27, 170, 331, 297, 197, 328, 360, 226, 100, 435, 408, 403, 236, 232, 146, 457, 249, 429, 369, 192, 124, 259, 21, 413, 104, 164, 125, 360, 410, 179, 255, 346, 169, 169, 420, 210, 243, 456, 156, 77, 339, 476, 494, 217, 191, 173, 197, 80, 462, 420, 393, 426, 221, 489, 301, 239, 326, 61, 232, 385, 72, 506, 485, 260, 423, 253, 427, 405, 286, 263, 367, 414, 392, 475, 252, 260, 216, 172, 406, 148, 126, 351, 184, 344, 317, 236, 175, 433, 221, 320, 79, 196, 395, 458, 443, 431, 124, 162, 376, 296, 319, 203, 147, 109, 384, 30, 185, 178, 419, 263, 105, 72, 220, 356}；物品重量系数 W={56, 51, 47, 37, 58, 28, 42, 25, 43, 44, 11, 14, 56, 14, 24, 45, 55, 40, 40, 38, 21, 23, 13, 58, 23, 40, 48, 11, 54, 53, 56, 10, 59, 33, 57, 24, 32, 40, 30, 32, 41, 25, 25, 17, 22, 53, 45, 12, 14, 50, 45, 12, 17, 57, 27, 22, 29, 43, 47, 21, 51, 58, 31, 58, 22, 30, 55, 30, 10, 26, 46, 16, 39, 44, 21, 51, 58, 52, 56, 17, 20, 53, 33, 56, 21, 53, 12, 48, 14, 35, 17, 53, 36, 11, 13, 41, 41, 20, 43, 11, 18, 32, 59, 52, 11, 18, 47, 49, 59, 57, 18, 49, 34, 24, 47, 36, 25, 49, 18, 45, 36, 45, 24, 43, 14, 33, 43, 43, 17, 30, 20, 40, 48, 40, 42, 14, 30, 13, 43, 33, 37, 38, 33, 19, 55, 54, 32, 19, 45, 51}，最优解 *xbest*={0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0}，最优值为 25 903，最优解的背包承重 2 115. 另外，需要指出的是：KP3 的重量数据的第 29 个取值为 50, KP6 的重量数据第 29 个取值为 25，而它们其它数据的取值相同.

第 1 组实验：比较本文基于区域分割的差分进化算法(PDE)和文献[2]的二进制差分进化算法(BDE1)以及文献[3]的离散差分进化算法(GDDEA)的全局搜索能力和稳健性. 3 种算法对 8 个实例独立运行 100 次，比较它们找到最优解的次数. 8 个算例的来源和 3 种算法的计算结果如表 2 所示.

将所有算法用 MATLAB7.01 编程，在 CPU 双核 2.9 GHz，内存 2.00 GHz 的微机(XP 环境)上运行. 文献[2]的 BDE1 算法采用“少数服从多数”的变异规则，且参数设置同原文献(种群规模为 50，最大的允许迭代次数为 200 代，交叉概率 0.1). 与文献[3]相同，GDDEA 算法连续进化 10 代重新初始化种群(种群规模为 50，最大的允许迭代次数为 200 代，交叉概率 0.1). 本文 PDE 算法种群规模为 50，最大的允许迭代次数为 200 代，交叉概率 0.1，若连续 10 次迭代当前最优解未得到更新，则启用区域分割算法，区域分割算法最多使用 20 次.

从表 2 可知，对 8 个算例的 100 次独立计算中，PDE 算法找到每个实例最优解的次数都多于 GDDEA 算法和和 BDE1 算法. PDE 算法找到最优解次数最少的都达到 98 次(KP7). 而其它 2 种算法就没那么稳健，其中 BDE1 算法找到 KP5 和 KP8 最优解的次数分别只有 29 次和 2 次，GDDEA 算法找到 KP7 和 KP8 最优解的次数分别只有 1 次和 8 次，. 因此，在种群规模相同条件下，本文算法更稳健，找到最优解的概率更大，全局寻优能力更强.

第 2 组实验：比较本文算法和文献[1]基于混合编码的差分进化算法(MCDE)以及文献[5]基于模式替代的遗传算法(GASR)找到最好解时所需的最少迭代次数. 3 种算法对 KP5、KP6、KP7 独立求解 10 次的计算结果见表 3. 本文 PDE 算法种群规模为 40，最大的允许迭代次数为 150 代，其它参数设置同第一组实验. MCDE 算法种群规模 50，变异概率 0.5，交叉概率 0.5，最大允许的迭代次数 150 代. GASR 算法种群规模 200，变异概率 0.06，交叉概率 0.618，最大允许的迭代次数 1 000 代.

在 10 次计算中，本文算法找到 KP5 和 KP6 的最优解 10 次，找到 KP7 的最优解 9 次. GASR 算法和 MCDE 算法 10 次计算均未找到 KP6 的最优解. 本文算法找到 3 个实例最优解的进化代数远小于 GASR 算法和 MCDE 算法的进化代数，其中本文算法找到 KP7 的最优解最少需进化 36 代，平均需进化代数为 78 代，其平均进化代数都小于 GASR 算法和 MCDE 算法的最少迭代次数(分别为 147 代和 137 代). 3 个实例表明，本文算法能以更少的种群、更少的迭代次数、更大的概率找到背包问题的最优解. 它比 GASR 算法和 MCDE 算法更稳健，全局寻优能力更强，所需种群规模和迭代次数更少.

5 结论

为了更有效地求解 0-1 背包问题，提出了基于区域分割的差异进化算法. 该算法将传统差分进化算法的变异操作改为分 2 步以概率变异的操作，保证了变异操作的封闭性. 将区域分割算法和背包问题的单调性相结合，割去了一些没有希望的点，缩小了最优解的搜索空间，因此增加了找到最优解的概率. 同时用区域分割算法找到的最好可行解替换了种群中目标函数值最差的解，增加了种群多样性，促进了种群的进化，增强了算法的全局搜索能力. 此外，对可行解的改进和对不可行解的修复也在一定程度上加速了算法的收敛速度. 数值实验表明，本文算法在求解背包问题时，比文献中的 DE 算法更稳健，求到最优解的概率更大，全局寻优能力更强，所需的种群规模和迭代次数更少. 因此，该算法更有效，为求解其它类似问题的提供了一个有效途径.

表 2　3 种差分进化算法求解 0-1 背包问题 100 次的结果

本文算例	来源	规模 n	最优解		找到最优解次数		
			最优值	背包承重	GDDEA	BDE1	PDE
KP1	文献[2]问题 1	10	295	269	100	100*	100
KP2	文献[2]问题 2	20	1 024	871	100	100*	100
KP3	文献[2]问题 3	50	3 103	1 000	100	100*	100
KP4	文献[2]问题 4	50	16 102	11 231	100	100*	100
KP5	文献[1]的 KP1	20	1 042	878	100	29	100
KP6	文献[1]的 KP2	50	3 119	1 000	87	100	100
KP7	文献[1]的 KP3	100	26 559	6 717	1	95	98
KP8	随机产生	150	25 903	2 115	8	2	100

注：带*的数据引自文献[2]

表 3　3 种算法找到的最好优解和所需的最少迭代次数对比

实例	GASR	MCDE	PDE
	最好结果(最少迭代次数)	最好结果(最少迭代次数)	最好结果(最少迭代次数)
KP5	1 042/878 (12)	1 042/878 (5)	1 042/878 (1)
KP6	3 103/1 000 (50)	3 105/997 (46)	3 119/1 000 (7)
KP7	26 559/6 717 (147)	26 559/6 717 (137)	26 559/6 717 (36)

注：MCDE 与 GASR 的计算结果直接引自文献[1].

6 参考文献

[1] 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.

[2] 蔡鸿英，郝志峰，王志刚，等．解 0-1 背包问题的二进制差异演化算法[J]．计算机工程与设计, 2009, 30(7): 1716-1721.

[3] 苗世清，高岳林．求解 0/1 背包问题的离散差分进化算法[J]．小微型计算机系统, 2009, 30(9): 1828-1830.

[4] 赵新超，杨婷婷．求背包问题的更贪心粒子群算法[J]．计算机工程与应用, 2009, 45(36): 32-34.

[5] 李康顺，贾玉珍，张文生．一种基于模式替代的遗传算法解 0/1 背包问题[J]．计算机应用研究, 2009, 26(2): 470-471.

[6] 樊小毛，马良．0-1 背包问题的蜂群优化算法[J]．数学的实践与认识, 2010, 40(6): 155-160.

[7] Stom R, Price K. Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous space [R]. Berkeley: University of California, 2006.

[8] Ali M M, Kajee-Bagdadi Z. A local exploration-based differential evolution algorithm for constrained global optimization [J]. Applied Mathematics and Computation, 2009, 208(1): 31-48.

[9] Mallipeddi R, Suganthan P N, Ensemble of constraint handling techniques [J]. IEEE Trans on Evolutionary Computation, 2010, 10(4): 311-338.

[10] 刘若辰，焦李成，雷峰，等．一种新的差分进化约束优化算法[J]．西安电子科技大学学报：自然科学版, 2011, 38(1): 47-53.

[11] Ruan Ning, Sun Xiaoling. An exact Algorithm for cost minimization in series reliability systems with multiple component choices[J]. Applied Mathematics and Computation, 2006,181(1): 732-741.

The Differential Evolution Algorithm Based on Domain Partition for Solving 0-1 Knapsack Problem

ZHONG Pei-hua, WU Zhi-yuan, HU Jian-gen, ZHU Li

(College of Science, Jiangxi Agricultural University, Nanchang Jiangxi 330045, China)

Abstract: In order to solve the 0-1 knapsack problems more effectively, a new differential evolution algorithm based on domain partition method is proposed. The mutation operator of traditional differential evolution algorithm (DE) is modified to guarantee the closure of mutation operation. With the domain partition method introduced in and some hopeless points removed off, the searching field of the optimal solution is reduced. Accordingly, the probability of finding the optimal solution is improved. To ensure the diversity of population, the solution with the least object function value is replaced by the best feasible solution found by the domain partition method combined with greedy method. Numerical experiments show that the new algorithm is of more robustness, of better global searching ability, and of greater probability to find the optimal solution of knapsack problem than those algorithms mentioned in the literatures.

Key words: knapsack problem; differential evolution; partition

(责任编辑：曾剑锋)