文章编号: 1001-0920(2012)09-1387-06

求解整数非线性规划结合正交杂交的离散PSO算法

张 莉[a,b], 冯大政[a], 李 宏[b]

(西安电子科技大学 a. 雷达信号处理国家重点实验室，b. 数学系，西安 710071)

摘 要: 针对整数非线性规划问题, 提出一种结合正交杂交的离散粒子群优化(PSO)算法. 首先采用舍入取整方法, 为了减少舍入误差, 对PSO中的每个粒子到目前为止的最好位置进行随机修正, 将基于正交实验设计的正交杂交算子引入离散PSO算法, 以增强搜索性能; 然后对PSO算法中的惯性权重和收缩因子采用动态调整策略, 以提高算法的搜索效率; 最后对一些不同维数的整数非线性规划问题进行数值仿真实验, 实验结果表明了所提出算法的有效性.

关键词: 整数非线性规划；粒子群优化算法；正交杂交；离散粒子群优化算法

中图分类号: O221.4 **文献标志码:** A

Discrete PSO combined with the orthogonal crossover for solving integer nonlinear programming

ZHANG Li[a,b], *FENG Da-zheng*[a], *LI Hong*[b]

(a. National Lab of Radar Signal Processing，b. Department of Mathematics，Xidian University，Xi'an 710071，China. Correspondent：ZHANG Li，E-mail：lzhang@xidian.edu.cn)

Abstract: For solving the integer nonlinear programming problems, a discrete particle swarm optimization(PSO) algorithm combined with the orthogonal crossover is proposed. In the PSO algorithm, each particle of the swarm is truncated to the nearest integer after the determination of its new position, and then each particle's best position till now is repaired by using a stochastic method to reduce the rounding error. The orthogonal crossover operator based on the orthogonal experimental design is integrated into discrete PSO algorithm to make a systematic and rational exploration. The inertia weight and constriction factor are dynamically adjusted to improve the efficiency of PSO algorithm. Some numerical examples with different dimensions are carried out and the experimental results show the effectiveness of the proposed algorithm.

Key words: integer nonlinear programming; particle swarm optimization; orthogonal crossover; discrete particle swarm optimization

1 引 言

整数规划是运筹学、组合数学、金融学、管理学、电子与信息科学等领域经常遇到的数学规划问题, 如商品配送、生产及人员调度、机车分配、资产预算、投资组合分析、神经网络及大规模集成电路设计等[1-7]. 由于变量在空间内离散分布, 处理连续变量的优化算法不能直接使用, 而且搜索方向与搜索步长比连续优化问题更难以寻找和控制, 使得搜索离散最优解变得比较困难.

粒子群优化(PSO)最初由Kennedy等[8]提出, 是模拟鸟群、蜂群等生物群体的社会行为而设计的求解连续搜索空间的智能算法, 具有原理简单、控制参数少、运行速度快等优点, 受到了人们的普遍关注, 并得到了广泛应用[1,3]. 此后, Kennedy等[9]又针对0-1整数变量问题引入了离散二进制PSO算法; Luo等[4-5]对其作了改进, 使其适用于求解不同的离散问题. 对于一般的整数问题, 即变量取搜索空间中的整数点, 文献[1-2]将PSO算法得到的实数解截断到最近的整数值(即舍入取整), 数值结果表明这种取整方法是有效的. 本文也采用这种取整方法, 并对其进行了改进. 在粒子群中的每个粒子的新位置确定后, 对每个粒子采用舍入取整, 然后求出每个粒子当前的最好位置, 再对其进行随机修正, 以减少舍入误差.

与其他进化算法一样, PSO也存在早熟收敛问

收稿日期: 2011-01-11；**修回日期:** 2011-04-14.

基金项目: 国家自然科学基金项目(60971111).

作者简介: 张莉(1976−), 女, 博士生, 从事盲信号处理、神经网络、智能计算与智能信息处理等研究；冯大政(1959−), 男, 教授, 博士生导师, 从事信号处理、神经网络、雷达信号处理等研究.

题, 即可能只收敛到问题的局部最优解. 大量文献表明, 将其他知识加入PSO构成混合算法, 可改善PSO的搜索性能, 提高优化效率. 正交设计是一种有效且应用较为广泛的统计实验方法, 它是一种利用正交表进行科学安排与分析多因素实验的方法. 正交表是运用组合数学理论在正交拉丁方的基础上构造的一种规格化的表格或矩阵, 其主要优点是能在很多实验方案中挑选出代表性强的少数几个实验方案, 并通过对这些少数实验方案的实验结果进行分析推断出最优方案[10-14]. 文献[11]首次将正交设计引入遗传算法, 构造了正交杂交, 提出了正交遗传算法. 许多文献对正交杂交进行了发展, 设计了许多基于正交杂交的进化算法, 使其能够求解不同的优化问题[12-15]. 本文则将正交杂交用于离散PSO, 利用正交杂交的系统推理能力来提高离散PSO的全局搜索能力和效率.

2 整数非线性规划问题

本文考虑的整数非线性规划是指变量取某一给定超长方体搜索空间中的整数点, 其数学形式如下:

$$\begin{cases} \min f(\boldsymbol{x}), \\ \text{s.t. } \boldsymbol{x} \in I. \end{cases} \tag{1}$$

其中: $\boldsymbol{x}=(x_1,\cdots,x_d,\cdots,x_D)$ 为整数向量, 第 d 个分量 x_d 为搜索区间 $[l_d,u_d]$ 中的整数; 问题(1)的搜索空间是 D 维空间上的整数点集 $I=\{\boldsymbol{x}\in Z^D | l_d\leqslant x_d\leqslant u_d;\ l_d,u_d\in Z;\ d=1,2,\cdots,D\}$, Z 表示整数集合; $f(\boldsymbol{x})$ 为非线性目标函数, 可以不连续或者不可微.

3 改进的离散型粒子群优化算法

3.1 粒子群的初始化

因为需要保证每个粒子是搜索空间的一个整数点, 所以按下面的方式随机产生第 i 个粒子的位置 $\boldsymbol{x}_i$ $(\boldsymbol{x}_i=(x_{i1},\cdots,x_{id},\cdots,x_{iD}))$:

$$x_{id}=l_d+\text{round}[r_d\cdot(u_d-l_d)],$$
$$d=1,2,\cdots,D. \tag{2}$$

其中: r_d 为 $(0,1)$ 内的随机数; round$[\cdot]$ 表示四舍五入取整, 以下相同; l_d和 u_d分别为 x_{id}的整数下界和上界.

初始速度 $\boldsymbol{v}_i=(v_{i1},\cdots,v_{id},\cdots,v_{iD})$ 随机设置, 其中 $v_{id}\in(0,v_{\max})$, $v_{\max}$ 表示设置的最大速度.

令 $i=1,2,\cdots,NP$, 产生 NP 个粒子的初始位置和初始速度.

3.2 连续变量的取整设计及最好位置更新

PSO算法是一种处理连续变量的群体智能算法, 通过对速度和位置两个更新公式进行不断调整, 并根据对环境的适应程度, 使群体中的粒子移动到较好区域. 因为连续型PSO算法的位置更新公式不能直接产生整数变量, 所以在位置更新时, 需对其进行四舍五入取整, 使其满足整数要求. 于是, 离散PSO算法中速度 $\boldsymbol{v}_i^{t+1}=(v_{i1}^{t+1},\cdots,v_{id}^{t+1},\cdots,v_{iD}^{t+1})$ 的更新公式和位置 $\boldsymbol{x}_i^{t+1}=(x_{i1}^{t+1},\cdots,x_{id}^{t+1},\cdots,x_{iD}^{t+1})$ 的更新公式分别如下:

$$v_{id}^{t+1}=\omega v_{id}^t+c_1r_1(p_{id}^t-x_{id}^t)+c_2r_2(p_{gd}^t-x_{id}^t), \tag{3}$$

$$x_{id}^{t+1}=\text{round}[x_{id}^t+\lambda v_{id}^{t+1}]. \tag{4}$$

其中: 第 i 个粒子 $\boldsymbol{x}_i^t$ 到目前为止经历过的最好位置为 $\boldsymbol{p}_i^t=(p_{i1}^t,\cdots,p_{id}^t,\cdots,p_{iD}^t)$, 粒子群中全局最好位置为 $\boldsymbol{p}_g^t=(p_{g1}^t,\cdots,p_{gd}^t,\cdots,p_{gD}^t)$, $g\in\{1,2,\cdots,NP\}$; ω 为惯性权重; c_1 和 c_2 为两个正常数, c_1称为认知常数, c_2 称为社会常数; r_1 和 r_2 为 $[0,1]$ 之间的随机数; λ 为收缩因子; 上标 $t(t=1,2,\cdots)$ 为迭代次数, t 表示当前代, $t+1$ 表示下一代; $d=1,2,\cdots,D$, $i=1,2,\cdots,NP$.

为防止由式(4)得到的 $x_{id}^{t+1}(d=1,2,\cdots,D)$ 落在搜索区间 $[l_d,u_d]$ 外, 进行如下修正:

$$x_{id}^{t+1}=\begin{cases} l_d,\ x_{id}^{t+1}<l_d; \\ x_{id}^{t+1},\ l_d\leqslant x_{id}^{t+1}\leqslant u_d; \\ u_d,\ x_{id}^{t+1}>u_d. \end{cases} \tag{5}$$

比较 $\boldsymbol{x}_i^{t+1}$和 $\boldsymbol{p}_i^t$ 的目标函数值, 较小者记为第 i 个粒子的最好位置 $\boldsymbol{p}_i^{t+1}=(p_{i1}^{t+1},\cdots,p_{id}^{t+1},\cdots,p_{iD}^{t+1})$. 考虑到四舍五入取整过程中可能存在一定误差, 为减小误差, 对 p_{id}^{t+1} 进行随机处理, 得到

$$p'^{t+1}_{id}=p_{id}^{t+1}+\text{round}\,[-1+2\cdot r_d],$$
$$d=1,2,\cdots,D, \tag{6}$$

其中 r_d 为 $(0,1)$ 之间的随机数.

为防止由式(6)得到的 $p'^{t+1}_{id}(d=1,2,\cdots,D)$ 落在搜索区间 $[l_d,u_d]$ 外, 按式(5)的方法修正 p'^{t+1}_{id}. 最后比较 $\boldsymbol{p}_i^{t+1}$ 与 $\boldsymbol{p}'^{t+1}_i$ 的目标函数值, 将二者中较好的作为第 i 个粒子在下一代的最好位置 $\boldsymbol{p}_i^{t+1}$.

3.3 基于正交实验设计的正交杂交算子

为增强离散PSO的搜索能力, 增加了基于正交设计的正交杂交算子. 首先随机选择两个不同粒子的最好位置 $\boldsymbol{p}_{k_1}^{t+1}$ 和 $\boldsymbol{p}_{k_2}^{t+1}$, 其中指标 $k_1,k_2\in\{1,2,\cdots,NP\}$, 且 $k_1\neq k_2$; 然后对它们进行正交杂交, 产生一个新的最好位置 $\boldsymbol{z}_b^{t+1}$; 最后用该最好位置替换群体中最差粒子的最好位置. 正交杂交的具体步骤如下.

Step 1: 选择合适的正交表. 通常标准两水平正交表的符号为 $L_N(2^{N-1})$, 其中 N 为实验次数, $N-1$ 表示因素个数. 两水平正交表是将数字1和2按一定规则排列成 N 行和 $N-1$ 列的矩阵, 正交表的每一行表示因素水平的一个组合, 每一列对应一个因素, 每一列的数字1和2表示某一因素的水平1和水平2. 根据问题的维数 D 和已知标准两水平正交表 $L_N(2^{N-1})$ 的列数 $(N-1)$, 选择合适的两水平正交表 $L_N(2^D)$. 如果 $D=N-1$, 则直接使用标准两水平正交表

$L_N(2^{N-1})$; 如果 $D<N-1$, 则使用标准两水平正交表 $L_N(2^{N-1})$ 前 D 列, 后 $(N-1-D)$ 列忽略[13].

Step 2: 由得到的正交表 $L_N(2^D)$ 产生 N 个临时位置. 首先, 将一个粒子的最好位置 $\boldsymbol{p}_{k_1}^{t+1}$ 和另一个粒子的最好位置 $\boldsymbol{p}_{k_2}^{t+1}$ 的 D 个分量看作 D 个因素, 第 d 个因素的水平 1 对应 $\boldsymbol{p}_{k_1}^{t+1}$ 的第 d 个分量值, 同时第 d 个因素的水平 2 对应 $\boldsymbol{p}_{k_2}^{t+1}$ 的第 d 个分量值; 然后, 将第 d 个因素的水平 1 和水平 2 对应的分量值分别填入正交表 $L_N(2^D)$ 中第 d 列的水平 1 和水平 2 处, 其中 $d=1,2,\cdots,D$, 从而得到一个新的 $N\times(N-1)$ 矩阵. 该矩阵的每一行就是一个水平组合, 它表示一个位置, 因此共产生 N 个临时位置, 记为 $\boldsymbol{y}_j^{t+1}, j=1,2,\cdots,N$.

Step 3: 由因素分析法[14]生成一个具有较好因素水平组合的新位置 $\boldsymbol{z}_b^{t+1}$. 对每个位置 $\boldsymbol{y}_j^{t+1}$ 计算目标函数值, 即 $f_j=f(\boldsymbol{y}_j^{t+1}), j=1,2,\cdots,N$. 根据式 (7) 对各因素对应的水平进行分析, 找出对该因素影响最大的是水平 1 还是水平 2. 设因素 d 的第 l 个水平的影响度[14]为

$$E_d(l)=\sum_{j=1}^{N} f_j\cdot\chi_j,\ d=1,2,\cdots,D \text{ 且 } l=1,2. \quad (7)$$

若在第 j 次实验中因素 d 的水平为 l, 则 $\chi_j=1$; 否则, $\chi_j=0$. 对于最小化问题 (1), 若 $E_d(1)<E_d(2)$, 则因素 d 选择水平 1; 否则, 选择水平 2. 在所有因素都选择了较好的水平后, 便得到新位置 $\boldsymbol{z}_b^{t+1}$.

3.4 动态参数设置

速度更新公式 (3) 中的 3 个参数惯性权重 ω, 加速常数 c_1 和 c_2 都是确定常数. 本文将惯性权重设置为动态参数, 即 ω 为 $(\omega_{\min},\omega_{\max})$ 之间的动态随机数, 它随迭代次数的变化而取不同值. 惯性权重 ω 设置如下:

若迭代次数 $t\leqslant 0.75G_{\max}$, 则

$$\omega=\omega_{\max}-(t-1)\cdot\left[\frac{\omega_{\max}-\omega_{\min}}{0.75G_{\max}}\right];$$

否则

$$\omega=\omega_{\max}-r\cdot(\omega_{\max}-\omega_{\min}).$$

其中: $\omega_{\max}$ 为 ω 的最大值, $\omega_{\min}$ 为其最小值, $r\in(0,1)$ 为随机数, $G_{\max}$ 为最大迭代次数.

3.5 结合正交杂交的离散 PSO 算法步骤

下面给出改进的离散 PSO 算法的具体实现步骤:

Step 1: 设定粒子群规模 NP, 最大和最小惯性权重 $\omega_{\max}$ 和 $\omega_{\min}$, 加速常数 c_1 和 c_2, 收缩因子 λ, 最大初始速度 $v_{\max}$, 合适的两水平正交表 $L_N(2^D)$, 最大迭代次数 $G_{\max}$.

Step 2: 令迭代次数计数器 $t=0$, 在搜索空间 I 中随机产生 NP 个初始位置 $\boldsymbol{x}_i^t$, 在 $(0,v_{\max})$ 中随机产生 NP 个初始速度 $\boldsymbol{v}_i^t$. 评价每个粒子 $\boldsymbol{x}_i^t, i=1,2,\cdots,NP$.

Step 3: 将每个粒子 $\boldsymbol{x}_i^t$ 与它所经历过的最好位置进行比较, 保留最好结果作为当前最好位置 $\boldsymbol{p}_i^t, i=1,2,\cdots,NP$.

Step 4: 将每个粒子 $\boldsymbol{x}_i^t\ (i=1,2,\cdots,NP)$ 与保留的粒子群经历过的全局最好位置进行比较, 从中选择最好结果作为全局最好位置 $\boldsymbol{p}_g^t$.

Step 5: 根据式 (3) 和 (4) 调整每个粒子的速度 $\boldsymbol{v}_i^{t+1}$ 和位置 $\boldsymbol{x}_i^{t+1}$; 检查和修正位置 $\boldsymbol{x}_i^{t+1}$, 使其位于搜索空间; 评价每个粒子 $\boldsymbol{x}_i^{t+1}$; 比较每个粒子 $\boldsymbol{x}_i^{t+1}$ 与它所经历的当前代最好位置 $\boldsymbol{p}_i^t$, 较好者记为第 i 个粒子的最好位置 $\boldsymbol{p}_i^{t+1}, i=1,2,\cdots,NP$.

Step 6: 由式 (6) 产生 $\boldsymbol{p'}_i^{t+1}$, 并检查和修正位置 $\boldsymbol{p'}_i^{t+1}$, 使其也位于搜索空间; 比较 $\boldsymbol{p}_i^{t+1}$ 与 $\boldsymbol{p'}_i^{t+1}$ 的目标函数值, 二者中较好者作为第 i 个粒子的下一代最好位置 $\boldsymbol{p}_i^{t+1}, i=1,2,\cdots,NP$.

Step 7: 随机选择两个粒子的最好位置 $\boldsymbol{p}_{k_1}^{t+1}$ 和 $\boldsymbol{p}_{k_2}^{t+1}$, 其中指标 $k_1,k_2\in\{1,2,\cdots,NP\}$, 且 $k_1\neq k_2$; 然后对其进行正交杂交, 产生一个新位置 $\boldsymbol{z}_b^{t+1}$; 最后用该位置替换群体中的最差粒子的最好位置, 求其目标函数值.

Step 8: 如果停机条件不满足, 则令 $t=t+1$, 返回 Step 3; 否则, 输出结果.

4 数值实验及比较

4.1 测试函数

为了评估求解整数非线性规划的结合正交杂交的离散 PSO 算法 (简记为 OXIPSO) 的性能, 对如下 15 个函数进行测试:

$$f_1(\boldsymbol{x})=\sum_{i=1}^{D}|x_i|,\ \boldsymbol{x}\in[-100,100]^D;$$

$$f_2(\boldsymbol{x})=\sum_{i=1}^{D}x_i^2,\ \boldsymbol{x}\in[-100,100]^D;$$

$$f_3(\boldsymbol{x})=$$

$$-20\exp\Big(-0.02\sqrt{D^{-1}\sum_{i=1}^{D}x_i^2}\Big)-$$

$$\exp\Big(D^{-1}\sum_{i=1}^{D}\cos(2\pi x_i)\Big)+20+e,\ \boldsymbol{x}\in[-30,30]^D;$$

$$f_4(\boldsymbol{x})=\Big(\frac{\pi}{D}\Big)\Big\{10\sin^2(\pi y_1)+\sum_{i=1}^{D-1}(y_i-1)^2[1+$$

$$10\sin^2(\pi y_{i+1})]+(y_D-1)^2\Big\},$$

$$y_i=1+\frac{1}{4}(x_i+1),\ \boldsymbol{x}\in[-10,10]^D;$$

$$f_5(\boldsymbol{x})=10D+\sum_{i=1}^{D}[x_i^2-10\cos(2\pi x_i)],\ \boldsymbol{x}\in[-5,5]^D;$$

$f_6(\boldsymbol{x}) = -a^{\top}\boldsymbol{x} + \boldsymbol{x}^{\top}A\boldsymbol{x},\ \boldsymbol{x} \in [-100, 100]^5,$

$$a = \begin{bmatrix} 15 \\ 27 \\ 36 \\ 18 \\ 12 \end{bmatrix},\ A = \begin{bmatrix} 35 & -20 & -10 & 32 & -10 \\ -20 & 40 & -6 & -31 & 32 \\ -10 & -6 & 11 & -6 & -10 \\ 32 & -31 & -6 & 38 & -20 \\ -10 & 32 & -10 & -20 & 31 \end{bmatrix};$$

$f_7(\boldsymbol{x}) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2,$

$\boldsymbol{x} \in [-100, 100]^2;$

$f_8(\boldsymbol{x}) = (9x_1^2 + 2x_2^2 - 11)^2 + (3x_1 + 4x_2^2 - 7)^2,$

$\boldsymbol{x} \in [-100, 100]^2;$

$f_9(\boldsymbol{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2,\ \boldsymbol{x} \in [-100, 100]^2;$

$f_{10}(\boldsymbol{x}) =$

$(x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 +$

$10(x_1 - x_4)^4,\ \boldsymbol{x} \in [-100, 100]^4;$

$$f_{11}(\boldsymbol{x}) = 1 - \exp\Big[\Big(-\frac{1}{60}\Big)\sum_{i=1}^{30} x_i^2\Big],\ \boldsymbol{x} \in [0, 5]^{30};$$

$f_{12}(\boldsymbol{x}) =$

$x_1^2 + x_1x_2 - x_2^2 + x_3x_1 - x_3^2 + 8x_4^2 - 17x_5^2 +$

$6x_6^3 + x_4x_5x_6x_7 + x_8^3 + x_9^4 - x_{10}^5 - x_{10}x_5 +$

$18x_3x_7x_6,\ \boldsymbol{x} \in [0, 99]^{10};$

$f_{13}(\boldsymbol{x}) =$

$100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 +$

$(1 - x_3)^2 + 10.1[(x_2 - 1)^2 + (x_4 - 1)^2] +$

$19.8(x_2 - 1)(x_4 - 1),\ \boldsymbol{x} \in [-10, 10]^4;$

$f_{14}(\boldsymbol{x}) =$

$[1.5 - x_1(1 - x_2)]^2 + [2.25 - x_1(1 - x_2^2)]^2 + [2.625 -$

$x_1(1 - x_2^3)]^2,\ x_i = 0.001j,\ -10^4 \leqslant j \leqslant 10^4;$

$f_{15}(\boldsymbol{x}) =$

$(x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4,$

$x_i = 0.001j,\ -10^4 \leqslant j \leqslant 10^4,\ i = 1, 2, 3, 4.$

其中: 对函数 $f_1 \sim f_5$ 测试了 25 维, 50 维和 100 维问题; 函数 f_{12} 为最大化问题, 其余函数都是最小化问题, 所有函数变量都取给定区间的整数; 函数 $f_1, f_2, f_6 \sim f_{10}$ 选自文献 [1], 函数 $f_3 \sim f_5$ 选自文献 [16], 函数 f_{11} 选自文献 [17], 其他函数选自文献 [6]; 已知函数 $f_1 \sim f_5, f_7 \sim f_{11}, f_{13} \sim f_{15}$ 的全局最小值为 0, 函数 f_6 的全局最小值为 -737, f_{12} 的全局最大值为 216 300 719.

4.2 实验结果与比较

为了说明结合正交杂交的离散 PSO (OXIPSO) 的有效性, 首先将 OXIPSO 中的正交杂交去掉, 得到不含正交杂交的离散 PSO, 记为 IPSO; 然后将 OXIPSO 与 IPSO 进行比较, 以说明正交杂交的作用. 标准 PSO 算法 (PSO) 的取整方法与文献 [1-2] 相同, 即将群体中的每个粒子四舍五入到最近的整数. 将 IPSO 与 PSO 进行比较的目的是说明四舍五入取整后再对舍入误差进行随机修正的必要性. OXIPSO, IPSO 和 PSO 这 3 种算法所使用的参数和停机条件全部相同.

3 种算法的参数设置如下: 最大初始速度 $v_{\max} = 4$; 粒子群规模

$$NP = \begin{cases} 30,\ D \leqslant 5, \\ 5D,\ D \geqslant 10, \end{cases}$$

其中 D 是问题维数; 最大和最小惯性权重 $\omega_{\max} = 0.9$, $\omega_{\min} = 0.1$; 加速参数 $c_1 = c_2 = 2$; 收缩因子 $\lambda = 0.45 + r \cdot (0.729 - 0.45)$, 其中 $r \in (0, 1)$ 为随机数; 最大迭代次数 $G_{\max} = 1\,000$.

正交杂交中用到的两水平正交表分别为 $L_4(2^2)$, $L_8(2^4), L_8(2^5), L_{12}(2^{10}), L_{32}(2^{25}), L_{32}(2^{30}), L_{64}(2^{50})$, $L_{200}(2^{100})$.

3 种算法的停机条件如下: 在最大迭代次数内, 当种群中全局最好解的目标函数值与已知的全局最优值的误差小于给定值 ($\varepsilon = 10^{-6}$) 时, 算法停止, 输出结果.

用函数计算次数评价算法的计算代价, 3 种算法每迭代一次所需函数计算次数分别如下: PSO 需 NP 次, NP 是粒子数规模; IPSO 需 $2NP$ 次; OXIPSO 需 $(2NP + N + 1)$ 次, N 表示正交表的行数.

对不同维数的函数分别独立运行 50次, 统计成功率、函数平均计算次数和平均 CPU 时间, 结果如表 1 所示. 图 1 ~图 6 为部分具有代表性的函数 (100 维的函数 $f_1 \sim f_5$ 和 30 维的函数 f_{11}) 在第 25 次运行中的收敛曲线, 其中横轴表示迭代次数, 纵轴表示目标函数值.

由表 1 可以看出: 对于不同维数的函数 $f_1 \sim f_5$, 尤其对于 100 维函数, OXIPSO 都能够以 100 % 的成功率找到其全局最优解, 而 IPSO 和 PSO 均不能找到所有 100 维函数的全局最优解; 对于 50 维函数, IPSO 和 PSO 都不能以 100 % 的成功率找到其全局最优解, 但 IPSO 的成功率高于 PSO 的成功率; 对于 25 维函数, IPSO 能够以 100 % 的成功率找到其全局最优解, 除函数 f_1 和 f_4 外, PSO 能以 100 % 的成功率求出其他问题的全局最优解. 因此, 在算法的稳健性方面, OXIPSO 优于其他两种算法, IPSO 优于 PSO. 在计算量方面, 总体而言, OXIPSO 的函数计算次数少于 IPSO 和 PSO.

对于维数较低的函数 $f_6 \sim f_{15}$, OXIPSO 和 IPSO 都能够以 100 % 的成功率找到所有函数的全局最优解, PSO 对这些函数也有较高的成功率. 从计算量看, 除函数 f_7, f_9 和 f_{13} 外, OXIPSO 所需的计算量比 IPSO

表 1　PSO, IPSO 和 OXIPSO对不同维数的函数 $f_1 \sim f_{15}$ 的统计结果

函数	维数	成功率/%			函数平均计算次数			平均CPU时间		
		PSO	IPSO	OXIPSO	PSO	IPSO	OXIPSO	PSO	IPSO	OXIPSO
f_1	25	82	100	100	39 090	40 925	36 309	0.95	0.94	0.83
f_1	50	10	52	100	230 495	295 820	79 644	8.67	9.33	2.43
f_1	100	0	0	100	500 500	1 000 500	191 507	30.86	51.28	30.03
f_2	25	100	100	100	21 678	42 975	43 005	0.42	0.76	0.82
f_2	50	24	72	100	203 690	220 030	91 509	5.89	5.48	2.01
f_2	100	0	0	100	500 500	1 000 500	221 148	21.10	33.51	21.64
f_3	25	100	100	100	19 608	40 475	39 179	1.08	2.28	2.09
f_3	50	36	78	100	175 135	200 900	85 588	17.61	20.36	8.11
f_3	100	0	0	100	500 500	1 000 500	205 271	94.73	185.98	38.59
f_4	25	52	100	100	70 510	42 130	36 581	5.28	3.07	2.71
f_4	50	2	16	100	246 360	438 470	85 101	33.63	60.27	11.44
f_4	100	0	0	100	500 500	1 000 500	208 801	137.18	264.13	54.61
f_5	25	100	100	100	16 405	35 075	23 874	0.62	1.31	0.89
f_5	50	42	94	100	165 140	117 950	51 609	9.94	7.28	2.15
f_5	100	0	0	100	500 500	1 000 500	162 179	59.13	115.46	19.07
f_6	5	80	100	100	9 416	7 398	8 966	0.17	0.13	0.18
f_7	2	100	100	100	1 123	778	715	0.02	0.01	0.03
f_8	2	100	100	100	645	590	594	0.01	0.01	0.03
f_9	2	86	100	100	5 821	2 032	1 519	0.07	0.03	0.07
f_{10}	4	90	100	100	5 799	5 484	6 357	0.08	0.08	0.30
f_{11}	30	100	100	100	1 176	2 244	2 288	0.04	0.08	0.27
f_{12}	10	16	100	100	42 146	2 342	2 509	0.62	0.04	0.13
f_{13}	4	98	100	100	1 855	1 537	1 151	0.03	0.02	0.05
f_{14}	2	100	100	100	3 400	5 882	6 295	0.05	0.13	0.28
f_{15}	4	100	100	100	14 489	17 473	19 321	0.20	0.23	0.89

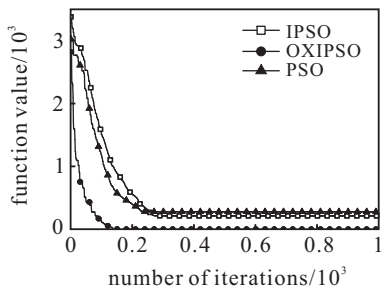

图 1　100维函数 f_1 的收敛曲线

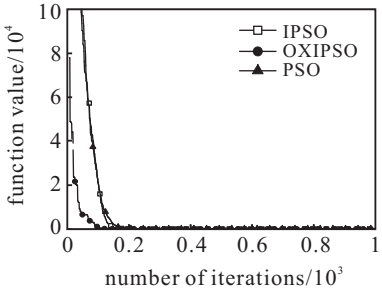

图 2　100维函数 f_2 的收敛曲线

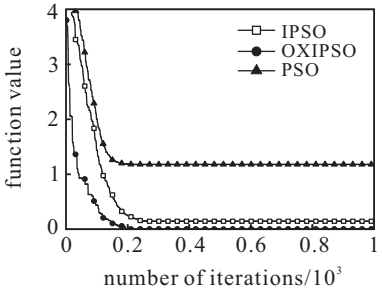

图 3　100维函数 f_3 的收敛曲线

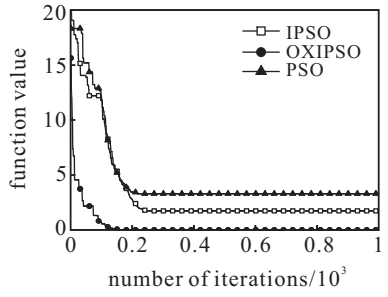

图 4　100维函数 f_4 的收敛曲线

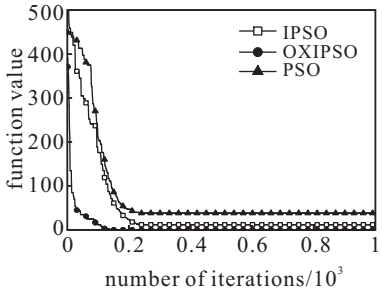

图 5　100维函数 f_5 的收敛曲线

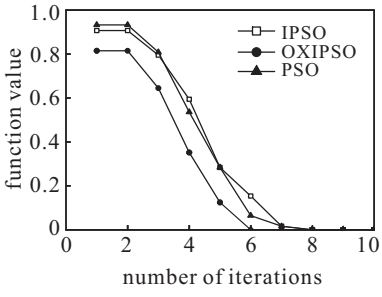

图 6　30维函数 f_{11} 的收敛曲线

稍大. 除函数 f_{10}, f_{11}, f_{14}和 f_{15} 外, OXIPSO 和IPSO 所需要的计算量均比PSO 小; 因此, 对于低维函数, OXIPSO 和 IPSO 都具有较好的稳健性, 而且计算量相差不多, 这两种算法都优于 PSO.

从图 1～图 6 可以看出, OXIPSO 的收敛性明显好于 IPSO, 而IPSO 的收敛性优于 PSO. 因此, 本文对取整方法的修正是必要和有效的, 在粒子群算法中嵌入正交杂交的方法也是有效的, 它提高了算法的搜索性能. 文献 [1] 中的离散 PSO 算法与本文给出的离散 PSO 算法类似, 只是参数选取不同. [1] 对函数 f_1 和 f_2 提供了最大 30 维的结果, 而本文 OXIPSO 测试的函数达到了 100 维. [16] 是一种离散的动态凸化方法, [17] 是嵌入模拟退火接受准则的可控随机搜索方法, [6] 是一种离散填充函数法. 本文 OXIPSO 的计算结果与

其他文献中的结果比较如表2所示.

表 2 OXIPSO与文献中算法对所有函数的结果比较

函数	维数	成功率/%		函数平均计算次数	
		文献结果	OXIPSO	文献结果	OXIPSO
$f_1^{[1]}$	25	100	100	171 110	36 309
$f_1^{[1]}$	50	–	100	–	79 644
$f_1^{[1]}$	100	–	100	–	191 507
$f_2^{[1]}$	25	100	100	171 610	43 005
$f_2^{[1]}$	50	–	100	–	91 509
$f_2^{[1]}$	100	–	100	–	221 148
$f_3^{[16]}$	25	100	100	36 868	39 179
$f_3^{[16]}$	50	100	100	143 952	85 588
$f_3^{[16]}$	100	100	100	567 984	205 271
$f_4^{[16]}$	25	100	100	17 878	36 581
$f_4^{[16]}$	50	100	100	72 640	85 101
$f_4^{[16]}$	100	100	100	255 312	208 801
$f_5^{[16]}$	25	100	100	5 146	23 874
$f_5^{[16]}$	50	100	100	19 960	51 609
$f_5^{[16]}$	100	100	100	78 416	162 179
$f_6^{[1]}$	5	80	100	75 060	8 966
$f_7^{[1]}$	2	100	100	8 066	715
$f_8^{[1]}$	2	100	100	8 322	594
$f_9^{[1]}$	2	100	100	8 388	1 519
$f_{10}^{[1]}$	4	92	100	18 477	6 357
$f_{11}^{[17]}$	30	100	100	26 042	2 288
$f_{12}^{[6]}$	10	100	100	122 497	2 509
$f_{13}^{[6]}$	4	100	100	4 263	1 151
$f_{14}^{[6]}$	2	100	100	939 210	6 295
$f_{15}^{[6]}$	4	100	100	7 337 208	19 321

由表2可以看出, 除25维函数 f_3, 25维和50维函数 f_4, 以及函数 f_5 外, 对于其他函数, OXIPSO所得结果均优于以往文献的结果, 表明所提出的OXIPSO具有良好的稳健性和较快的收敛速度.

5 结 论

整数非线性规划属于离散优化问题, 其求解比连续非线性规划问题困难. 由于其变量限制为整数, 需考虑离散空间中的整数点的搜索方法. 为此, 本文对已有的取整方法进行改进, 提出了一种离散PSO算法. 在此基础上, 结合基于正交实验设计的正交杂交算子, 提出一种基于正交杂交的离散PSO算法, 提高了离散PSO算法的搜索性能. 对15个不同维数的函数进行了测试, 其结果表明了所提出算法的有效性. 进一步的工作是采用本文算法优化带整数权值的神经网络, 这种神经网络的传递函数和权值被限定为某一范围内的整数, 这是因为带整数权值的神经网络比分数(或小数)权值的神经网络更便于硬件实现[7].

参考文献(References)

[1] Parsopoulos K E, Vrahatis M N. Recent approaches to global optimization problems through particle swarm optimization[J]. Natural Computing, 2002, 1(2/3): 235-306.

[2] Laskari E C, Parsopoulos K E, Vrahatis M N. Particle swarm optimization for integer programming[C]. Proc of the 2002 Congress on Evolutionary Computation. Honolulu: IEEE Conf Publications, 2002: 1582-1587.

[3] Poli R, Kennedy J, Blackwell T. Particle swarm optimization an overview[J]. Swarm Intelligence, 2007, 1(1): 33-57.

[4] Luo Y, Yuan X, Liu Y. An improved PSO algorithm for solving non-convex NLP/MINLP problems with equality constraints[J]. Computers and Chemical Engineering, 2007, 31(3): 153-162.

[5] Chen W N, Zhang J, Chung H S H, et al. A novel set-based particle swarm optimization method for discrete optimization problems[J]. IEEE Trans on Evolutionary Computation, 2010, 14(2): 278-300.

[6] Ng C K, Zhang L S, Li D, et al. Discrete filled function method for discrete global optimization[J]. Computational Optimization and Applications, 2005, 31(1): 87-115.

[7] Plagianakos V P, Vrahatis M N. Parallel evolutionary training algorithms for "hardware-friendly" neural networks[J]. Natural Computing, 2002, 1(2/3): 307-322.

[8] Kennedy J, Eberhart R C. Particle swarm optimization[C]. IEEE Int Conf on Neural Networks. Perth: IEEE Conf Publications, 1995, 4: 1942-1948.

[9] Kennedy J, Eberhart R C. A discrete binary version of the particle swarm algorithm[C]. IEEE Int Conf on Systems, Man, and Cybernetics. Orlando: IEEE Conf Publications, 1997, 5: 4104-4108.

[10] 方开泰, 马长兴. 正交与均匀试验设计[M]. 北京: 科学出版社, 2001: 35-77, 233-235.
(Fang K T, Ma C X. Orthogonal and uniform experimental design[M]. Beijing: Science Press, 2001: 35-77, 233-235.)

[11] Zhang Q, Leung Y W. An orthogonal genetic algorithm for multimedia multicast routing[J]. IEEE Trans on Evolutionary Computation, 1999, 3(1): 53-62.

[12] Leung Y W, Wang Y. An orthogonal genetic algorithm with quantization for global numerical optimization[J]. IEEE Trans on Evolutionary Computation, 2001, 5(1): 41-53.

[13] Tsai J T, Liu T K, Chou J H. Hybrid taguchi-genetic algorithm for global numerical optimization[J]. IEEE Trans on Evolutionary Computation, 2004, 8(4): 365-377.

[14] Ho S Y, Shu L S, Chen J H. Intelligent evolutionary algorithms for large parameter optimization problems[J]. IEEE Trans on Evolutionary Computation, 2004, 8(6): 522-540.

[15] Wang Y, Liu H, Cai Z, et al. An orthogonal design based constrained evolutionary optimization algorithm[J]. Engineering Optimization, 2007, 39(6): 715-736.

(下转第1401页)