ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.6, June 2010, pp.1296−1307
doi: 10.3724/SP.J.1001.2010.03592

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

求解全局优化问题的混合自适应正交遗传算法*

江中央, 蔡自兴, 王 勇+

(中南大学 信息科学与工程学院,湖南 长沙 410083)

Hybrid Self-Adaptive Orthogonal Genetic Algorithm for Solving Global Optimization Problems

JIANG Zhong-Yang, CAI Zi-Xing, WANG Yong+

(School of Information Science and Engineering, Central South University, Changsha 410083, China)

+ Corresponding author: E-mail: ywang@csu.edu.cn

Jiang ZY, Cai ZX, Wang Y. Hybrid self-adaptive orthogonal genetic algorithm for solving global optimization problems. *Journal of Software*, 2010,21(6):1296−1307. http://www.jos.org.cn/1000-9825/3592.htm

Abstract: This paper presents a hybrid self-adaptive orthogonal genetic algorithm (HSOGA) based on orthogonal experimental design method for solving global optimization problems. In HSOGA, the orthogonal experimental design method is utilized to design crossover operator, and as a result, a self-adaptive orthogonal crossover operator is proposed. The self-adaptive orthogonal crossover operator self-adaptively adjusts the number of orthogonal array's factors and the location for dividing the parents into several sub-vectors according to the similarity of the two parents, in order to produce a small but representative set of points as the potential offspring. In addition, in HSOGA the self-adaptive orthogonal crossover operator is also adopted to generate an initial population that is scattered uniformly over the feasible solution space in order to maintain the diversity. Moreover, a local search scheme is incorporated into HSOGA in the purpose of enhancing the local search ability and speeding up the convergence of HSOGA. HSOGA is tested with fourteen benchmark functions. The experimental results suggest that HSOGA is generic and effective.

Key words: orthogonal genetic algorithm; local search; global optimization; orthogonal experimental design

摘 要: 提出了一种基于正交实验设计的混合自适应正交遗传算法(hybrid self-adaptive orthogonal genetic algorithm,简称 HSOGA)以求解全局优化问题,此算法利用正交实验设计方法设计交叉算子,并提出一种自适应正交交叉算子.该自适应正交交叉算子根据父代个体的相似度自适应地调整正交表的因素个数和对父代个体进行因素分割的位置,生成具有代表性的子代个体,以更好地搜索空间.此外,新算法利用自适应正交交叉算子生成均匀分布的初始种群,以保证初始群体的多样性.同时引入了局部搜索策略以提高算法局部搜索能力和收敛速度.通过 14 个

* Supported by the National Natural Science Foundation of China under Grant Nos.90820302, 60805027 (国家自然科学基金); the Specialized Research Foud for the Doctoral Program of Higher Education of China under Grant No.200805330005 (高等学校博士学科点专项科研基金); the Graduate Innovation Fund of Hu'nan Province of China under Grant No.CX2009B039 (湖南省研究生创新基金); the Graduate Degree Thesis Innovation Foundation of Central South University of China under Grant No.1373-74334000016 (中南大学研究生学位论文创新基金)

Received 2008-03-03; Revised 2008-08-07; Accepted 2009-02-16

高维的 Benchmark 函数验证了算法的通用性和有效性.

关键词: 正交遗传算法;局部搜索;全局优化;正交实验设计
中图法分类号: TP181 文献标识码: A

全局优化问题是工程应用领域经常遇到的一类数学规划问题,一般的全局优化问题可以描述为

$$\text{minimize } f(x),\ x=(x_1,...,x_N)\in S=\prod_{i=1}^{N}[l_i,u_i] \tag{1}$$

其中 N 表示决策变量个数, $f(x)$为目标函数, $l=(l_1,l_2,...,l_N)$, $u=(u_1,u_2,...,u_N)$, $[l,u]$为可行解空间.遗传算法是模仿生物遗传学和自然选择机理,通过人工方式构造的一类优化搜索算法,被广泛地用于求解全局优化问题.虽然遗传算法在许多优化问题中都有成功的应用,但大量的研究表明[1],传统的遗传算法也存在许多的不足和缺陷,如早熟收敛、计算量大和局部搜索能力差.为了有效地克服以上缺点,近年来有不少学者将正交实验设计方法引入到遗传算法中,来处理各种函数优化问题[2−5].Zhang 和 Leung[2]认为遗传算法的一些步骤可以从实验的角度去考虑,例如,交叉算子从选定的父代个体的附近随机产生新的子代个体,这个操作可以看作是采样实验,因此他们将正交实验设计的方法引入到遗传算法中,设计多媒体的多点传输路由.Leung 和 Wang[3]利用正交表初始化种群和安排交叉操作提出了一种量化的正交遗传算法(OGA/Q)来求解高维单目标的数值优化问题.曾三友等人[4]把正交实验方法和相应的统计优化方法相结合,提出了一种基于正交设计的多目标演化算法求解多目标优化问题.Wang 等人[5]把正交设计和约束处理技术相结合提出了基于正交设计的约束优化算法来处理约束优化问题.这些方法都取得了很好的效果.

本文根据父代个体的相似度,利用正交实验设计的方法设计交叉算子,提出了一种新的自适应正交交叉算子.为了提高种群的进化速度,还引入了基于种群分割和单形搜索的局部搜索策略.数值实验的结果验证了算法的有效性.

1 正交实验设计[2,3]

在实际的实验场合中,一般实验系统有 F 个因素,当每一个因素有 Q 个水平时,则有 Q^F 个组合,若进行全面组合实验,则要做 Q^F 组实验.但是当 Q 和 F 很大时,不可能做 Q^F 组实验.正交实验设计是一种解决多因素、多水平实验问题的有效方法,它利用正交表 $L_M(Q^F)$安排少数次实验,就能找到最好或者较好的实验条件,因此它被广泛地用于寻优.$L_M(Q^F)$表示一个具有 F 个因素和 Q 个水平的正交表,其中 L 表示拉丁方,M 表示水平组合数.$L_M(Q^F)$有 M 行,每一行表示一个水平的组合.应用正交表 $L_M(Q^F)$,只需要选择 M 个组合去做实验,这里 M 一般远小于 Q^F.以正交表 $L_9(3^4)$为例,如式(2),对 4 因素、3 水平的问题而言,若依据正交表 $L_9(3^4)$来进行正交实验设计,则需做 9 次实验,但若进行全面组合实验,则需 3^4=81 次实验.可见正交实验设计大大减少了实验次数,且因素和水平越大,该方法的优越性越明显.

$$L_9(3^4)=\begin{bmatrix}1&1&1&1\\1&2&2&2\\1&3&3&3\\2&1&2&3\\2&2&3&1\\2&3&1&2\\3&1&3&2\\3&2&1&3\\3&3&2&1\end{bmatrix} \tag{2}$$

为方便起见,记 $L_M(Q^F)=[a_{i,j}]_{M\times F}$,其中第 i 个组合的第 j 个因素的水平值为 $a_{i,j},a_{i,j}\in\{1,2,\ldots,Q\}$.设正交表 $[a_{i,j}]_{M\times F}$ 第 j 列为 a_j,若 $j=1,2,(Q^3-1)/(Q-1)+1,...,\ (Q^{J-1}-1)/(Q-1)+1$, 则 a_j 称为基本列,其他列称为非基本列.文献[3]给出了创建正交表 $L_M(Q^F)$的算法,它首先创建基本的列,然后再创建非基本的列,详细描述见算法 1.其中

Q 为素数,且 $M=Q^J$,J 满足式(3),因此,当 Q 给定时,正交表 $L_M(Q^F)$因素个数 F 越大,即正交表的列数越大,水平组合数 M 也越大.

$$F \le \frac{Q^J-1}{Q-1} \tag{3}$$

算法 **1**. 构建正交表 $L_M(Q^F)$.

Step 1. Select the smallest J fulfilling $(Q^J-1)/(Q-1) \ge F$

Step 2. If $(Q^J-1)/(Q-1)=F$,then $F'=F$ else $F'=(Q^J-1)/(Q-1)$

Step 3. Construct the basic columns as follows:

for k=1 to J **do**

$j=\dfrac{Q^{k-1}-1}{Q-1}+1$;

for i=1 to Q^J **do**

$a_{i,j}=\left\lfloor \dfrac{i-1}{Q^{J-1}} \right\rfloor \bmod Q$;

end for

end for

Step 4. Construct the non-basic columns as follows:

for k=2 to J **do**

$j=\dfrac{Q^{k-1}-1}{Q-1}+1$;

for s=1 to j−1**do**

for t=1 to Q−1 **do**

$a_{j+(s-1)(q-1)+t}=(a_s \times t+a_j) \bmod Q$;

end for

end for

end for

Step 5. Increment $a_{i,j}$ by one for all $1 \le i \le M$ and $1 \le j \le F'$

Step 6. Delete the last $F'-F$ columns of $L_{Q^J}(Q^{F'})$ to get $L_M(Q^F)$ where $M=Q^J$

2 混合自适应正交遗传算法

在群体进化的过程中,文献[3,5]使用固定的正交表来安排父代个体的交叉操作,且因素分割的位置是随机产生的,但当参与交叉的父代个体之间的空间距离很近即相似度很高时,若选用固定的正交表来安排交叉操作,则会产生很多冗余个体,使子代群体的多样性变差,这种现象在群体进化后期更为明显.而在对父代个体进行交叉操作的时候,应尽可能使新产生的子代个体均匀分布在父代个体所确定的可行解空间,并根据所确定的空间大小调节子代个体繁殖的数量,以维持群体的多样性和减少计算开销.为了尽可能实现这个目的,本文根据父代个体的相似度自适应地选择正交表的因素个数和调整对父代个体进行因素分割的位置,提出了自适应正交交叉算子(self-adaptive orthogonal crossover,简称 SOC).本文利用 SOC 算子在可行解空间进行全局搜索.为了进一步提高其学习能力和收敛速度,本文将 SOC 算子和局部搜索策略相结合,并提出了混合自适应正交遗传算法(hybrid self-adaptive orthogonal genetic algorithm,简称 HSOGA).

2.1 自适应正交交叉算子

设有两对父代个体,a 和 b,c 和 d,其中 $a=(3,0.5)$, $b=(0.5,3)$, $c=(3.5,3.5)$, $d=(4,4)$,个体 a 和 b 确定的空间

区域[(0.5,0.5),(3,3)]如图 1 中的区域 A 所示,c 和 d 确定的空间区域[(3.5,3.5),(4,4)]如图 1 中的区域 B 所示.若选用正交表 $L_{25}(5^2)$,即 Q=5,F=2,M=25,对这两对父代个体安排交叉操作,它们各自将产生 M 个后代个体,个体的分布如图 1 所示.由第 2 节分析可知,当正交表 $L_M(Q^F)$的水平数 Q 一定时,我们可以通过调节正交表的因素个数即 F 的大小,来调节水平组合数 M 的大小即调节子代个体的数目.从图 1 可以看出,个体 c 和 d 的距离近,它们所确定的空间区域 B 相对较小,若不减少用于安排交叉操作的正交表的因素个数,则产生后代个体的数目过多,且个体浓度过大.因此,应根据父代个体所确定的区域大小动态地选择合适的正交表,安排父代个体的交叉操作.

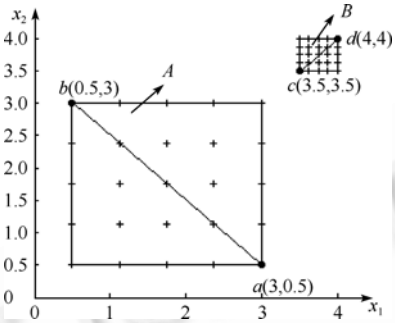

Fig.1 Distribution of the offspring generated by using orthogonal array to arrange the crossover operation of the parent chromosome

图 1 利用正交表安排父代个体交叉操作产生的子代个体分布图

父代个体的空间距离越近,其相似度就越高,由这两个父代个体所确定的空间区域则越小.在对两者利用正交表安排交叉操作时,如果所选择的正交表的因素个数不变,则群体的浓度会增加且群体的多样性会变差,这种现象在群体进化的后期更为明显.在群体进化的中后期,所有的个体不断地向适应度高的方向聚集,个体之间的空间距离越来越近.因此适当地减少相似父代个体繁殖子代个体的数目,不仅能够减少计算量,还可以抑制群体的浓度过高.

设 $p_1=(p_{1,1},p_{1,2},...,p_{1,N})$, $p_2=(p_{2,1},p_{2,2},...,p_{2,N})$ 为参与交叉操作的一对父代个体,由 p_1,p_2 所确定的可行解空间为 $[l_{parent},u_{parent}]$,其中 $\begin{cases} l_{parent}=[\min(p_{1,1},p_{2,1}),\min(p_{1,2},p_{2,2}),...,\min(p_{1,N},p_{2,N})] \\ u_{parent}=[\max(p_{1,1},p_{2,1}),\max(p_{1,2},p_{2,2}),...,\max(p_{1,N},p_{2,N})] \end{cases}$.

为了描述方便,我们给出如下定义:

定义 **1**. 令 $\delta_i=|p_{1i}-p_{2i}|$,其中 i=1,2,…,N.定义 δ_i 为 p_1,p_2 的第 i 维相似度值.

p_{1i} 与 p_{2i} 数值越接近,相似度值 δ_i 越小,则 p_{1i}, p_{2i} 相似度越高.在本文中,若 $\delta_i>\delta_0$ (δ_0 为给定的接近 0 的正实数),则认为 p_1,p_2 的第 i 维相似度低.

个体 p_1, p_2 之间的空间距离为

$$d=\sqrt{\sum_{i=1}^{N}(p_{1i}-p_{2i})^2} \tag{4}$$

由定义 1 可知,当 p_{1i} 和 p_{2i} 的相似度很高时,δ_i 接近于 0,即 $|p_{1i}-p_{2i}|$ 的数值很小.因此, p_{1i}, p_{2i} 对 d 的影响很弱,即个体 p_1,p_2 的第 i 维对区域 $[l_{parent},u_{parent}]$ 的影响很小.可以认为,解空间 $[l_{parent},u_{parent}]$ 的大小主要取决于 p_1,p_2 中那些相似度低的分量.令 J 为父代个体 p_1,p_2 中相似度低的分量所在的位置集合,则:

$$J=\{i \,||\, p_{1i}-p_{2i}\,|>\delta_0, i=1,2,...,N\} \tag{5}$$

不妨设 p_1,p_2 的前 t 个分量相似度低,后 N–t 个分量有较高的相似度.则 p_1,p_2 前 t 分量将组成一个 t 维的多面体,我们将它记为 Z.由前面的分析可以知道,p_1,p_2 所确定空间 $[l_{parent},u_{parent}]$ 的大小主要取决于 p_1,p_2 中相似度低的分量,即区域 Z 的体积大小.当 p_1,p_2 的后 N–t 个分量相似度值为 0,即后 N–t 个分量完全相等时,区域 $[l_{parent},u_{parent}]$ 的大小就等于区域 Z 的大小.所以通过对 p_1,p_2 相似度低的分量所确定的空间 Z 进行有效的搜索,可以减少盲目搜索,提高了对区域 $[l_{parent},u_{parent}]$ 的搜索能力.

例如,设参与交叉操作的两个父代个体 $p_1=(2,4,5,7), p_2=(1,3,5,7)$,则 p_1,p_2 所确定的可行解空间 $[l_{parents},u_{parents}]=[(1,3,5,7),(2,4,5,7)]$,取 $\delta_0=0.05$,则 $t=2$.p_1,p_2 前 t 维相似度低,它们确定的空间 $Z=[(1,3),(2,4)]$.取水平数 $Q=2$,构造正交表 $L_M(Q^F)$,即 $L_4(2^2)$.

$$L_4(2^2)=\begin{bmatrix}1 & 1\\ 1 & 2\\ 2 & 1\\ 2 & 2\end{bmatrix} \tag{6}$$

若将 p_1,p_2 的第 1 维和第 2 维分别作为一个因素,记为 factor_1 和 factor_2,将 p_1,p_2 的第 3 维和第 4 个维放入与之相邻的因素 factor_2 中,这样就将父代个体分割成 2 个因素,因素分割的位置如式(7)中的虚线所示.然后将 factor_1 和 factor_2 分割成 2 个水平.这样就把对 p_1,p_2 交叉操作转化为 2 因素、2 水平的实验问题.最后利用正交表 $L_4(2^2)$安排正交实验设计,产生子代种群 P^1,见式(7):

$$\begin{cases}p_1:(2,\vdots\ 4,\ 5,\ 7)\\ p_2:(1,\vdots\ 3,\ 5,\ 7)\end{cases}\longrightarrow P^1=\begin{cases}(2,\ 4,\ 5,\ 7)\\ (2,\ 3,\ 5,\ 7)\\ (1,\ 4,\ 5,\ 7)\\ (1,\ 3,\ 5,\ 7)\end{cases} \tag{7}$$

若将 p_1,p_2 的第 1 维和第 2 维一起作为一个因素,记为 factor_1,将 p_1,p_2 的第 3 维和第 4 维一起作为一个因素,记为 factor_2,因素分割的位置如式(8)中的虚线所示.然后将 factor_1 和 factor_2 分割成 2 个水平.这是同样把对 p_1,p_2 交叉操作转化为 2 因素、2 水平的实验问题.最后利用正交表 $L_4(2^2)$安排正交实验设计,产生子代种群 P^2,见式(8):

$$\begin{cases}p_1:(2,\ 4,\vdots\ 5,\ 7)\\ p_2:(1,\ 3,\vdots\ 5,\ 7)\end{cases}\longrightarrow P^2=\begin{cases}(2,\ 4,\ 5,\ 7)\\ (2,\ 4,\ 5,\ 7)\\ (1,\ 3,\ 5,\ 7)\\ (1,\ 3,\ 5,\ 7)\end{cases} \tag{8}$$

对种群 P^1 中的个体而言,这些个体的前 t 个分量在空间 $Z=[(1,3),(2,4)]$ 是均匀分布的.与种群 P^2 相比,种群 P^1 中的个体分布更均匀且群体的多样性更好.所以正交表的因素个数确定后,对父代个体进行因素分割的位置对子代种群的影响非常重要.由前面分析可知,p_1,p_2 中相似度低的维数是影响 p_1,p_2 所确定的可行解空间 $[l_{parent},u_{parent}]$ 大小的主要因素,因此应将这些维分别置于正交实验设计的不同因素中,使新产生的个体在空间 Z 中均匀分布,以充分利用这些维数所包含的空间信息,对空间 Z 进行有效地搜索,从而提高子代种群分布的均匀性、多样性,以提高对可行解空间 $[l_{parent},u_{parent}]$ 的搜索效率.

当 p_1,p_2 的所有相似度值大于 δ_0 的分量即相似度低的分量不相邻时,设相似度值大于 δ_0 的分量的个数为 t,将每一个相似度值大于 δ_0 的分量作为一个因素,那么这些分量在 p_1,p_2 中的位置就是进行因素分割的位置,p_1,p_2 中剩余的其他分量加入到与其位置相邻的因素中,因素分割的具体方法见算法 2 的 Step2,然后将每一个因素分割成 Q 个水平.因此,这样就将 N 维的两个父代个体的交叉操作转化为 t 因素、Q 水平实验问题,然后构建正交表 $L_M(Q^F)$安排正交设计实验,产生 M 子代个体,其中 $F=t$.自适应正交交叉算子的实现步骤见算法 2.

当水平个数 $Q=2$ 时,自适应正交交叉算子则变成多点交叉算子,对父代个体进行因素分割的位置就是进行交叉操作的位置,且当 $t=2$ 时,自适应交叉算子则变成单点交叉算子.在多点交叉中,其交叉组合有多种方式存在,随着交叉点的增多,组合方式的数量将会急剧增长.自适应正交交叉算子通过分析父代个体的相似度和其确定的空间区域信息,自适应地调整交叉点的个数和交叉操作的位置,交叉点的个数即为正交表的因素个数,交叉点的位置即对父代个体进行因素分割的位置,然后构造正交表,进行正交实验设计产生具有代表性的组合生成子代个体,极大地提高了搜索效率.

算法 2. 自适应正交交叉算子(self-adaptive orthogonal crossover,简称 SOC).

Step 1. 设 $p_1=(p_{1,1},p_{1,2},...,p_{1,N})$,$p_2=(p_{2,1},p_{2,2},...,p_{2,N})$ 为参与交叉操作的两个父代个体,由 p_1 和 p_2 所确定的可行解空间为 $[l_{parent},u_{parent}]$.接着把空间 $[l_{parent},u_{parent}]$ 中的第 i 维离散化为 Q 个水平,即 $\beta_{i,1},\beta_{i,2},\ldots,$

$\beta_{i,Q}$,$i\in\{1,2,\ldots,N\}$,记 $\beta_i=(\beta_{i,1},\beta_{i,2},...,\beta_{i,Q})$,其中,

$$\beta_{ij}=\begin{cases}\min(p_{1,i},p_{2,i}), & j=1\\ \min(p_{1,i},p_{2,i})+(j-1)\left(\dfrac{|p_{1,j}-p_{2,j}|}{Q-1}\right), & 2\le j\le Q-1\\ \max(p_{1,i},p_{2,i}), & j=Q\end{cases} \tag{9}$$

Step 2. 令向量 $k=[k_1,k_2,...,k_t]$,并且满足:$k_i\in J$ 且 $1\le k_1<k_2<...<k_t\le N$,j=1,2,…,t,集合 J 的定义见式(5),t 为 p_1,p_2 中相似度低的分量的个数,其中 δ_0 是给定的接近于 0 的正实数.向量 k 保存了相似度低的分量在 p_1,p_2 中的位置,即进行因素分割的位置.设 x 为 p_1,p_2 中的任一个体,把个体 $x=(x_1,x_2,...,x_N)$ 分成 t 份,其中每一份表示个体 x 的一个因素:

$$\begin{cases}f_1=(x_1,...,x_{k_1})\\ f_2=(x_{k_1+1},...,x_{k_2})\\ ...\\ f_t=(x_{k_{t-1}+1},...,x_N)\end{cases} \tag{10}$$

令 k_0=0,则第 i 个因素 f_i 的 Q 个水平可以表示为

$$\begin{cases}f_i(1)=(\beta_{k_{i-1}+1,1},\beta_{k_{i-1}+2,1},...,\beta_{k_i,1})\\ f_i(2)=(\beta_{k_{i-1}+1,2},\beta_{k_{i-1}+2,2},...,\beta_{k_i,2})\\ ...\\ f_i(Q)=(\beta_{k_{i-1}+1,Q},\beta_{k_{i-1}+2,Q},...,\beta_{k_i,Q})\end{cases} \tag{11}$$

Step 3. 根据算法 1 构造正交表 $L_M(Q^F)=[b_{i,j}]_{M\times F}$,其中 F=t.利用正交表 $L_M(Q^F)$来对式(10)中确定的 t 个因素和式(11)中确定每一个因素所对应的 Q 个水平进行正交实验设计,将产生 M 个子代个体:

$$\begin{cases}(f_1(b_{1,1}),f_2(b_{1,2}),...,f_F(b_{1,F}))\\ (f_1(b_{2,1}),f_2(b_{2,2}),...,f_F(b_{2,F}))\\ ...\\ (f_1(b_{M,1}),f_2(b_{M,2}),...,f_F(b_{M,F}))\end{cases} \tag{12}$$

我们以个体 $p_1=(2,1,6,4,2,2)$, $p_1=(0,3,8,4,2,2)$ 为例,取水平个数 Q=3.执行算法 2,对个体 p_1,p_2 进行自适应正交交叉操作,其详细过程如下:

Step 1. 计算 p_1,p_2 所确定的可行解空间 $[l_{parent},u_{parent}]$=[(0,1,6,4,2,2),(2,3,8,4,2,2)],把空间 $[l_{parent},u_{parent}]$ 的第 i 维离散化为 β_i:

$$\begin{cases}\beta_1=(0,1,2)\\ \beta_2=(1,2,3)\\ \beta_3=(6,7,8)\\ \beta_4=(4,4,4)\\ \beta_5=(2,2,2)\\ \beta_6=(2,2,2)\end{cases} \tag{13}$$

Step 2. 取 δ_0=0.05,由公式(5)得集合 J={1,2,3},t=3,F=3,则 k=[1,2,3].根据公式(10),将 $x=(x_1,x_2,x_3,x_4,x_5,x_6)$ 分割成 3 个因素: $f_1=(x_1)$, $f_2=(x_2)$, $f_3=(x_3,x_4,x_5,x_6)$.

Step 3. 构造正交表 $L_9(3^3)$,进行正交实验,产生 9 个子代个体:

$$\begin{cases}(0,1,6,4,2,2)\\(0,2,7,4,2,2)\\(0,3,8,4,2,2)\\(1,1,7,4,2,2)\\(1,2,8,4,2,2)\\(1,3,6,4,2,2)\\(2,1,8,4,2,2)\\(2,2,6,4,2,2)\\(2,3,7,4,2,2)\end{cases} \tag{14}$$

2.2 种群初始化

在对一些高维多模态的函数进行全局优化时,函数本身具有多个极点,而函数全局最优点的位置是未知的,因此在群体初始化过程中,应尽可能地使初始种群均匀地覆盖整个可行域.正交实验设计是研究多因素、多水平的一种数学实验方法,它是根据正交表,从全面实验中挑选出部分具有代表性的点进行实验,这些点具备了均匀分散,齐整可比的特点,所以本文选择采用正交实验法生成初始种群.初始种群分布的均匀性,保证了初始群体的多样性和较丰富的模式,从而使算法能在全局范围内以较快的速度收敛.采用正交实验方法初始化种群时,当可行解空间$[l,u]$较大时,为了提高搜索效率和精度,将可行解空间$[l,u]$分割成S个子空间,分割方法见本节算法3,然后构造正交表 $L_M(Q_0{}^F)$,根据正交表 $L_M(Q_0{}^F)$对每一个子空间利用所提出的 SOC 算子进行交叉操作,生成初始种群.

算法 **3**. 子空间分割.

Step 1. 选择可行解空间的第 s 维,s 满足下式:

$$u_s - l_s = \max_{1\le i\le N}\{u_i - l_i\} \tag{15}$$

Step 2. 将可行解空间$[l,u]$在第 s 维处分割成 S 个子空间 $[l(1),u(1)],[l(2),u(2)],...,[l(S),u(S)]$.

$$\begin{cases} l(i) = l + (i-1)\left(\dfrac{u_s - l_s}{S}\right)I_s \\ u(i) = u - (S-1)\left(\dfrac{u_s - l_s}{S}\right)I_s \end{cases}, i = 1,2,...,S \tag{16}$$

其中 $I_s = [c_{1,j}]_{1\times N}$, $c_{1,j} = \begin{cases}1, j = s\\ 0, j = 0\end{cases}$.

2.3 聚类局部搜索策略

为了增强算法学习能力,提高其收敛速度,受文献[11]的启发,本文还引入了局部搜索策略.先对群体进行聚类,将种群分割为互不相交的局部邻域,使每个子种群仅覆盖搜索空间一个较小的邻域.对每个子种群,即局部邻域,利用单形交叉算子(SPX)[6,7]进行局部搜索.将种群分割为互不相交的局部邻域,以提高单形搜索的局部收敛速度,同时对不相交的局部邻域进行并行搜索,以实现快速的全局搜索.种群分割方法使该算法具有搜索空间结构自学习能力,能够充分地利用搜索空间的局部信息.

设 N 维最优问题的可行解空间为$[l,u]$,种群 P 的规模为 n,$\varnothing$表示空集,o 表示可行解空间中随机选择的一个参考点,种群 P 经局部搜索后生成新的种群 P'.局部搜索的执行过程是:首先,将 $P=\{p_1,p_2,...,p_n\}$ 中的 n 个点,按照相似个体聚类的规则分解成若干个子种群,每个子种群由 m 个个体组成,本文 m=3.接着,利用 SPX 算子对每一个子种群中的 m 个个体(即父代群体,记为 S)进行交叉操作,产生新的 g 子代个体(即子代群体,记为 S'),本文 g=10,最后把每一个子代种群加入到群体 P'中.其算法描述见算法 4,其中$\lfloor\bullet\rfloor$表示取整.

算法 **4**. 局部搜索算法.

$o=(x_1,x_2,...,x_N), k=1, P'=\varnothing;$

while($k<\lfloor n/m \rfloor$)

$S=\varnothing, S'=\varnothing$;

确定群体 P 中离参考点 o 最近的一点 $p_j, j\in\{1,2,\ldots,n_k\}$;

$S=\{p_j\}\cup\{P$ 中离 p_j 最近的 $m-1$ 个点$\}$;

$P=P-S$,此时 $P=\{p_1,p_2,...,p_{n_k}\}$;

$S'=\mathrm{SPX}(S)$;

$P'=P'\cup S'$;

$k=k+1$;

End

2.4 算法步骤

Step 1. 种群初始化.

将可行解空间$[l,u]$分割成 S 个子空间,分割方法见算法 3.取正交表的水平个数为 Q_0,利用 SOC 算子对每一个子空间进行交叉操作,产生新的群体 P,计算其适应度值,从群体 P 中选择适应度值最好的 n 个个体生成初始种群 P_0.

Step 2. 生成临时种群 P'_{gen} .

设 gen 为进化代数,群体 P_{gen} 中的每一个个体,以概率 p_c 被选择进入临时群体 P'_{gen} .若群体 P'_{gen} 中个体的数目为奇数时,再从 P_{gen} 中随机选择一个个体加入到种群 P'_{gen} 中.

Step 3. 交叉操作.

对群体 P'_{gen} 的个体进行随机配对,每一对个体经过 SOC 交叉操作后,产生新的后代个体,从新产生的个体中选择一个适应度值最好的个体加入到群体 C_{gen} 中.其中,取正交表的水平个数为 Q.

Step 4. 局部搜索.

种群 P'_{gen} 经局部搜索后,生成新的种群 L_{gen} 局部搜索的方法见算法 4.

Step 5. 变异操作

种群 P'_{gen} 的任一个体 $p_i=(p_{i,1},p_{i,2},...,p_{i,N}), i\in\{1,2,...,n\}$,以概率 p_m 参与变异操作.具体操作是:① 产生一个随机整数 $j\in[1,N]$,和一个随机数实数 $r\in[0,1]$;② 令 $p_{i,j}=l_j+r(u_j-l_j)$.群体 P 经变异后生成的新种群记 G_{gen}.

Step 6. 选择操作

为了保持群体的多样性,从种群$(P_{gen}+C_{gen}+L_{gen}+G_{gen})$中选择适应度值最好的$\lfloor n\times 70\% \rfloor$个个体进入下一代种群 P_{gen+1},再从种群$(P_{gen}+C_{gen}+L_{gen}+G_{gen})$剩余的个体中,随机选择 $n-\lfloor n\times 70\% \rfloor$个个体进入到下一代种群 P_{gen+1}.

Step 7. 终止条件判断:若达到规定的代数或得到满意的结果,则结束并输出结果,否则转 Step2.

3 实验结果与分析

为了测试本文提出的混合自适应正交遗传算法(HSOGA)的性能,我们选取了 14 个高维的 Benchmark 函数作为测试集,并与其他 2 种已有的性能较好的算法进行了实验结果的比较.对函数 f_1–f_6 及函数 f_{10}–f_{14},N=30 维,对函数 f_7–f_9,N=100 维函数.函数 f_1–f_8 属多峰函数,每一个函数具有多个局部极值点,其中函数 f_7 具有 $100!=9.33\times10^{157}$ 个局部极值点,搜索过程容易陷入局部最优,这些函数能检验算法的多峰搜索能力.

$$f_1=\sum_{i=1}^{N}\left(-x_i\sin\left(\sqrt{|x_i|}\right)\right),\ -500\le x_i\le 500\ .$$

$$f_2=\sum_{i=1}^{N}(x_i^2-10\cos(2\pi x_i)+10),\ 5.12\le x_i\le 5.12.$$

$$f_3=-20\exp\left(-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N}x_i^2}\right)-\exp\left(\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi x_i)\right)+20+\exp(1),\quad -32\le x_i\le 32.$$

$$f_4=\frac{1}{4000}\sum_{i=1}^{N}x_i^2-\prod_{i=1}^{N}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1,\quad -600\le x_i\le 600.$$

$$f_5=\frac{\pi}{N}\left\{10\sin^2(\pi y_1)+\sum_{i=1}^{N-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right]+(y_N-1)^2\right\}+\sum_{i=1}^{N}u(x_i,10,100,4),\quad -5.12\le x_i\le 5.12.$$

其中, $y_i=1+\frac{1}{4}(x_i+1)$, $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & x_i>a\\ 0, & -a\le x_i\le a\\ k(-x_i-a)^m, & x_i<a\end{cases}$.

$$f_6=\frac{1}{10}\left\{\sin^2(3\pi x_1)+\sum_{i=1}^{N-1}(x_i-1)^2\left[1+\sin^2(3\pi x_{i+1})\right]+(x_N-1)^2\left[1+\sin^2(2\pi x_N)\right]\right\}+\sum_{i=1}^{N}u(x_i,5,100,4),\ -50\le x_i\le 50.$$

$$f_7=-\sum_{i=1}^{N}\sin(x_i)\sin^{20}\left(\frac{i\times x_i^2}{\pi}\right),\quad 0\le x_i\le \pi.$$

$$f_8=\frac{1}{N}\sum_{i=1}^{N}\left(x_i^4-16x_i^2+5x_i\right),\quad 0\le x_i\le 5.$$

$$f_9=\sum_{j=1}^{N-1}\left[100\left(x_j^2-x_{j+1}\right)^2+\left(x_j-1\right)^2\right],\quad -5\le x_i\le 10\ .$$

$$f_{10}=\sum_{j=1}^{N}x_i^2,\quad -100\le x_i\le 100.$$

$$f_{11}=\sum_{i=1}^{N}x_i^4+\text{random}[0,1),\quad -1.28\le x_i\le 1.28.$$

$$f_{12}=\sum_{i=1}^{N}|x_i|+\prod_{i=1}^{N}|x_i|,\quad -10\le x_i\le 10.$$

$$f_{13}=\sum_{i=1}^{N}\left(\sum_{j=1}^{i}x_j\right)^2,\quad -100\le x_i\le 100.$$

$$f_{14}=\max\left\{|x_i|,i=1,2,...,N\right\},\quad -100\le x_i\le 100.$$

数值实验在 MATLAB 中完成**,对所有的测试函数,种群的规模 n=200,Q_0=N−1,Q=2,S=5,P_c=0.6,P_m=0.1,δ_0=0.05,当 HSOGA 算法找到最优解或运行 gen=120 代,则终止运行.对于每一个函数,我们在相同的条件下独立运行 50 次,记录其平均函数评价次数(M-num-fun),最优平均值(M-best)和标准差(St. dev).为了对比实验结果,将本文算法(HSOGA)与经典的算法 OGA/Q[3]和较新的算法 LEA[8]进行比较.从表 1 可知,算法对于函数 f_5−f_9 找到的解接近于全局最优解,对其他函数都能找到全局最优解.对函数 f_7 评价 1 500 000 次,我们找到的最好解 x*的函数值为 $f_7(x^*)$=−99.618,x*的值见表 3,而文献[3,8−10]公布的最好解的函数值为−99.278.表 1 列出了实验中 3 种算法在 14 个测试函数上的函数平均计算次数,最优平均值及标准方差比较的结果.与 OGA/Q 算法比,HSOGA 算法在函数 f_1, f_3-f_5, f_8, f_9, f_{11} 的最优平均值、平均函数评价次数、标准差方面明显优于 OGA/Q 算法,对函数 f_2, f_{10}, $f_{12}-f_{14}$,HSOGA 算法和 OGA/Q 算法都能找到了最优解.对函数 f_6,f_7,HSOGA 算法得到的标准差大于 OGA/Q 算法,但找到的解的质量显著优于 OGA/Q 算法.与 LEA 算法相比,HSOGA 算法在函数 f_1-f_3, f_5-f_6, f_8-f_{14} 的最优平均值、平均函数评价次数、标准差方面明显优于 LEA 算法.对函数 f_7,虽然 HSOGA 算法得到的标准差大于 LEA 算法,但找到解的质量显著优于 LEA 算法,对函数 f_4,HSOGA 算法和 LEA 算法都找到了最优解.

** 若需要实验代码可与作者联系.

从上述的实验结果可以看出,将本文提出来的自适应正交交叉算子和局部搜索策略有机结合起来而形成的混合自适应正交遗传算法(HSOGA),在求解复杂的高维函数优化中,显示出了良好的性能.

为了验证自适应正交交叉算子对 HSOGA 算法性能的影响,将 SOC 算子从 HSOGA 算法中移去而形成新的算法记为 HGA,并选择复杂的多峰测试函数 f_1,f_6,f_7 对 HSOGA 算法和 HGA 算法进行对比实验.实验参数设置如下:对于 HSOGA 算法,种群的规模 n=200,Q_0=N−1,Q=2,S=5,P_c=0.6,P_m=0.1,δ_0=0.05,适应度函数评价次数设定为 200 000;与 HSOGA 算法相比,HGA 算法除不含参数 Q,δ_0 以外,其他实验参数设置与 HSOGA 相同.对于每一个测试函数,在相同的条件下独立运行 50 次,记录其解的最优值(best),中间值(median)、最差值(worst)、最优平均值(M-best)和标准差(St.dev).表 3 列出了 HSOGA 算法和 HGA 算法的实验结果比较.

Table 1 Comparison of three algorithms (OGA/Q[3],LEA[8], and HSOGA) on 14 benchmark functions

表 **1** 3 种算法(OGA/Q[3],LEA[8],HSOGA)对 14 个测试函数的实验结果比较.

Function/ optimal	Status	Algorithm		
		OGA/Q	LEA	HSOGA
F_1/−12 569.5	M-num-fun	302 116	287 365	101 151
	M-best	−12 569.453 7	−12 569.454 2	−12 569.486 6
	St.dev	6.447×10^{-4}	4.831×10^{-4}	3.168×10^{-5}
F_2/0	M-num-fun	224 710	223 803	8 420
	M-best	0	2.103×10^{-18}	0
	St.dev	0	3.0359×10^{-18}	0
f_3/0	M-num-fun	105 926	112 421	8 420
	M-best	3.274×10^{-16}	4.440×10^{-16}	0
	St.dev	3.001×10^{-17}	3.989×10^{-17}	0
f_4/0	M-num-fun	130 498	134 000	8 420
	M-best	6.104×10^{-16}	0	0
	St.dev	2.513×10^{-17}	0	0
f_5/0	M-num-fun	134 556	132 642	98 745
	M-best	6.019×10^{-6}	2.482×10^{-6}	2.0808×10^{-11}
	St.dev	1.159×10^{-6}	2.276×10^{-6}	1.2544×10^{-10}
f_6/0	M-num-fun	134 143	130 213	105 518
	M-best	1.869×10^{-4}	1.734×10^{-4}	4.1316×10^{-5}
	St.dev	2.615×10^{-5}	1.205×10^{-4}	3.0987×10^{-5}
f_7/−99.619	M-num-fun	302 773	289 863	236 867
	M-best	-92.83	-93.01	-98.0987
	St.dev	0.02626	0.02314	0.27125
f_8/−78.33236	M-num-fun	245 930	243 895	161 147
	M-best	−78.3000296	−78.310	−78.332331
	St.dev	6.288×10^{-3}	6.127×10^{-3}	2.356×10^{-7}
f_9/0	M-num-fun	167 863	168 910	167 374
	M-best	0.7520	0.5609	5.941×10^{-5}
	St.dev	0.1140	0.1078	4.0216×10^{-4}
f_{10}/0	M-num-fun	112 559	110 674	8 240
	M-best	0	4.727×10^{-16}	0
	St.dev	0	6.218×10^{-17}	0
f_{11}/0	M-num-fun	112 652	111 093	8 240
	M-best	6.301×10^{-3}	5.136×10^{-3}	0
	St.dev	4.069×10^{-4}	4.432×10^{-4}	0
f_{12}/0	M-num-fun	112 612	110 031	8 240
	M-best	0	4.247×10^{-19}	0
	St.dev	0	4.236×10^{-19}	0
f_{13}/0	M-num-fun	112 576	110 604	8 240
	M-best	0	6.783×10^{-18}	0
	St.dev	0	5.429×10^{-18}	0
f_{14}/0	M-num-fun	112 893	111 105	8 240
	M-best	0	2.683×10^{-16}	0
	St.dev	0	6.257×10^{-17}	0

Table 2 The optimal solution x^* of f_7 found by HSOGA

表 2 HSOGA 算法找到函数 f_7 的最优解 x^*

x^* =(2.202905547653280,1.570796214486428,1.284991573407600,1.923058388896578,1.720469806895425,1.570796348243437,
1.454414249822540,1.756086529133708,1.655717410008324,1.570796326414277,1.497728801123797,1.696616301332517,
1.630075723629949,1.570796327164349,1.517546144807838,1.666065026898027,1.616328712211932,1.570796788345188,
1.528907225843681,1.647456357417006,1.607757297131768,1.570796326390791,1.536272526517082,1.634931506943574,
1.601901830669802,1.570796270305114,1.541435141842966,1.625925976137474,1.597647948096007,1.570796299378235,
1.545254595484606,1.619138222286553,1.594417596438591,1.570796379934915,1.548194567190928,1.526546902649965,
1.591881034428863,1.570796313343089,1.550527822715220,1.609586107340208,1.589836453853538,1.570796327219784,
1.552424882718390,1.606098595613659,1.588153300004406,1.570796326128234,1.553996272464775,1.537723975503683,
1.586743574514364,1.570796413824375,1.555320400458290,1.540293110285201,1.585545197006467,1.570796327374178,
1.556451164688094,1.598599762777015,1.584515257093225,1.570795754967596,1.557427799413277,1.596761313060471,
1.583619196955086,1.570796326921408,1.558279959922572,1.546058142407977,1.534119410225983,1.570796305000000,
1.559030026953491,1.593727644730032,1.536269493892730,1.570795943255299,1.603638560303743,1.592463298428713,
1.538181406064810,1.570796326907612,1.560289298990284,1.591328444080860,1.580962812901610,1.570796259054305,
1.560822996552620,1.590309524812736,1.580462784142368,1.570794483397516,1.561305093133228,1.589386150344377,
1.580009236001596,1.570796317098385,1.561742730786400,1.588545120286697,1.579595952974376,1.570796319851633,
1.562141789372054,1.587778362586528,1.579218838519324,1.570796322547901,1.562507164309846,1.587073730676870,
1.578872386011957,1.570796365608036,1.562842930466590,1.586429358220735), $f_7(x^*)$=−99.618006161436

Table 3 Comparison of two algorithms (HSOGA and HGA) on functions f_1, f_6, and f_7

表 3 两种算法(HSOGA,HGA)对测试函数 f_1,f_6,f_7 的实验结果比较

Function/optimal	Method	Best	Median	Worst	M-best	St.dev
f_1/−12569.5	HSOGA	−12 569.486 6	−12 569.486 6	−12 569.486 6	−12 569.486 6	2.2461×10^{-5}
	HGA	−12 569.486 6	−12 569.486 5	−12 569.438 07	−12 569.484 4	0.1497
f_6/0	HSOGA	5.6376×10^{-6}	2.40987×10^{-5}	7.6835×10^{-5}	2.9735×10^{-5}	2.5046×10^{-5}
	HGA	6.9033×10^{-6}	4.7731×10^{-5}	0.0012	1.4032×10^{-4}	2.7621×10^{-4}
f_7/−99.619	HSOGA	−98.621 5	−98.046 3	97.861 1	−98.012 5	0.322 6
	HGA	−91.458 1	−90.463 1	−89.418 1	−90.474 6	0.416 3

从表 3 中可以看出,在函数评价次数相等的情况下,HSOGA 算法在函数 f_1 的解的中间值、最差值、平均值和解的标准差方面均优于 HGA 算法;对于函数 f_6,f_7,HSOGA 算法在所比较的 5 个方面均优于 HGA 算法.因此,从 HSOGA 算法和 HGA 算法的实验结果比较可以看出,SOC 算子有效提高了 HSOGA 算法稳定性和搜索精度.

4 小 结

本文结合正交设计和局部搜索技术,提出了一种求解全局优化问题的新方法.该方法有以下优点:① 根据父代个体之间的相似度和它们所确定的区域信息,利用正交实验设计方法,在该区域产生具有代表性的子代个体,提高对空间的搜索效率,减少搜索的盲目性.② 将种群分割为互不相交的局部邻域,使每个子种群仅覆盖搜索空间一个较小的邻域,从而增加对多个局部最优解的并行搜索能力,实现快速、可靠的全局搜索.将该算法推广到约束优化和多目标优化是进一步的工作.

References:

[1] Chen GL, Wang XF, Zhuang ZQ, Wang DS. Genetic Algorithm and Its Applications. Beijing: People's Post & Telecommunications Publishing House, 1996 (in Chinese).

[2] Zhang Q, Leung YW. An orthogonal genetic algorithm for multimedia multicast routing. IEEE Trans. on Evolutionary Computation, 1999,3(1):53−62. [doi: 10.1109/4235.752920]

[3] Leung YW, Wang YP. An orthogonal genetic algorithm with quantization for global numerical optimization. IEEE Trans. on Evolutionary Computation, 2001,5(1):41−53. [doi: 10.1109/4235.910464]

[4] Zeng SY, Wei W, Kang LS, Yao SZ. A multi-objective evolutionary algorithm based on orthogonal design. Chinese Journal of Computers, 2005,28(7):1153−1162 (in Chinese with English abstract).

[5] Wang Y, Liu H, Cai Z, Zhou Y. An orthogonal design based constrained evolutionary optimization algorithm. Engineering Optimization, 2007,39(6):715−736. [doi: 10.1080/03052150701280541]

[6] Tsutsui S, Yamamura M, Higuchi T. Multi-Parent recombination with simplex crossover in real-coded genetic algorithms. In: Proc. of the Genetic and Evolutionary Computation Conf. 1999. 657−664.

[7] Cai ZX, Wang Y. A multiobjective optimization-based evolutionary algorithm for constrained optimization. IEEE Trans. on Evolutionary Computation, 2006,10(6):658−675. [doi: 10.1109/TEVC.2006.872344]

[8] Wang YP, Dang CY. An evolutionary algorithm for global optimization based on level-set evolution and Latin squares. IEEE Trans. on Evolutionary Computation, 2007,11(5):579−595. [doi: 10.1109/TEVC.2006.886802]

[9] Tsai JT, Liu TK, Chou J H. Hybrid Taguchi-genetic algorithm for global numerical optimization. IEEE Trans. on Evolutionary Computation, 2004,8(4):365−377. [doi: 10.1109/TEVC.2004.826895]

[10] Zhang Q, Sun J, Tsang E, Ford J. Hybrid estimation of distribution algorithm for global optimization. Engineering Computations, 2004,21(1):91−107. [doi: 10.1108/02644400410511864]

[11] Guo GQ, Yu SY, He SL. Theoretic analysis and accelerating of a class of self-adaptive niching genetic algorithms. Chinese Journal of Computers, 2003,26(6):753−758 (in Chinese with English abstract).

附中文参考文献:

[1] 陈国良,王熙法,庄镇泉,王东生.遗传算法及其应用.北京:人民邮电出版社,1996.

[4] 曾三友,魏巍,康立三,姚书振.基于正交设计的多目标演化算法.计算机学报,2005,28(7):1153−1162.

[11] 郭观七,喻寿益,贺素良.自适应小生态遗传算法的理论分析和加速技术.计算机学报,2003,26(6):753−758.

江中央(1982－),男,湖南衡阳人,硕士,主要研究领域为进化计算,全局优化,免疫控制.

蔡自兴(1938－),男,教授,博士生导师,CCF高级会员,主要研究领域为人工智能,计算智能,智能控制.

王勇(1980－),男,博士生,讲师,主要研究领域为进化计算,约束优化,多目标优化.