第 34 卷 第 4 期 计 算 机 工 程 2008 年 2 月
Vol.34 No.4 Computer Engineering February 2008
·人工智能及识别技术· 文章编号：1000—3428(2008)04—0231—02 文献标识码：A 中图分类号：TP18

基于改进郭涛算法的 CCEA 函数优化问题

张 萍，李 涛，李振华
(中国地质大学计算机学院，武汉 430074)

摘 要：郭涛算法在求解函数优化问题方面具有独特的优势，其核心在于多父体杂交。鉴于郭涛算法只有杂交操作而没有变异操作，该文引入高斯正态分布变异算子，提高了对复杂问题的求解效率。分析合作式协同演化算法(CCEA)，采用多种群相互作用协同进化的策略求解复杂问题。同时在合作式协同演化模型中引入了郭涛算法，求解复杂高维的函数优化问题。实验结果表明，该模型的效率优于其他模型。
关键词：郭涛算法；高斯变异算子；合作式协同演化算法

Improved-GT-Operator-Based Cooperative Co-evolutionary Algorithm to Function Optimization

ZHANG Ping, LI Tao, LI Zhen-hua
(Department of Computer, China University of Geosciences, Wuhan 430074)

【Abstract】 Guo Tao(GT) algorithm is highly effective in solving function optimization problems. The core of the algorithm is multi-parent recombination, but it only has crossover operator, no mutation. Gauss mutation operator of Evolution Strategies(ES) is introduced. Cooperative Co-Evolution Algorithm(CCEA) uses multi-population strategy, which means that the fitness of an individual depends on the relationship between that individual and other individuals. CCEA is high-efficient in solving complicated problem. A cooperative co-evolution model to function optimization is proposed, based on an improved Guo Tao operator, which employs a Gauss mutation operator to enhance its exploring ability. Experimental results show that the model is more effective than others.
【Key words】 Guo Tao algorithm; Gauss mutation operator; cooperative co-evolution algorithm

郭涛算法是郭涛 1999 年提出的一种基于子空间搜索(多父体杂交)的群体随机搜索算法，主要用来求解复杂函数优化问题。协同演化算法(Co-Evolution Algorithm, CEA)是在现有演化算法(如GA)的基础上形成的一种解空间分离编码的动态搜索和优化技术，采用了多种群(而不是一个种群)相互作用协同进化的策略[1-3]，CEA的一个最显著的特点是个体适应值与其他种群的个体相关，通常包括多个相关的进化种群。CEA 按协同演化关系可分为竞争式协同演化算法(Competitive CEA)与合作式协同演化算法(Cooperative CEA)，总称CCEA，文献[3]将合作式协同演化方法应用于函数优化问题，效果良好。

1 算法分析

1.1 改进的郭涛算法

本文只考虑如下函数优化问题:

$$\min_{x\in D} f(X)\, x\in D$$

其中，$X=\{x_1,x_2,\cdots,x_n\}\in R^n$；$D=\{X\in R^n \mid l_i\le x_i\le u_i, i=1,2,\cdots,n\}$；记 D 中的 M 个点为 X。$X'_j=\left(x'_{1j},x'_{2j},\cdots,x'_{nj}\right)\quad j=1,2,\cdots,M$。

所张成的子空间为

$$V=\{X\in D \mid X=\sum_{i=1}^{M} a_j X_j\}$$

其中，a_j满足 $\sum_{i=1}^{M} a_i=1, -0.5\le a_i\le 1.5$。

郭涛算法在求解函数优化问题方面具有搜索空间大，运行一次可找出多个最优解的优势[4]。但郭涛算法中只有杂交操作，而没有变异操作。针对该特点，笔者对郭涛算法做了一些改进[5]，即算法中引入了高斯正态分布变异算子：$N(0,\sigma_i)$为高斯随机变量，均值为 0，标准方差为 σ_i。即正态分布随机变量，一般 σ 与 $\sqrt{|f(x)|}$ 成正比，经验值为

$$\sigma=1.224\times\sqrt{|f(x)|}/n$$

其中，n 为群体规模。高斯变异算子能对被选中的个体进行微调，这对求一些抖动剧烈的函数具有很好的效果。算法采用实数编码，易实现，而且求解精度比采用二进制编码高。

1.2 协同演化算法

一种常用的基于合作式协同演化框架(CCEA)[1]用于求解高维函数优化问题。CCEA的基本框架描述如下：

初始化子种群集合{ $P(i)\mid 1\le i\le N$ }，N 为子种群数量;
计算各子种群的初始适应度;
当终止条件没有满足时
Begin
　对于每个子种群 $P(i)$
　Begin
　　进行选择、交叉、变异等演化操作;
　　对于每一个体 $X(i,j)\in P(i)$,
　　根据其他子种群提供的合作个体计算适应度;
End;

作者简介：张 萍(1984－)，女，硕士研究生，主研方向：演化计算与算法设计；李 涛，硕士研究生；李振华，副教授、博士
收稿日期：2007-04-15 **E-mail：**apple_cug@126.com

End;

协同演化采用多个种群共同进化。在应用 CCEA 框架求解优化问题时，除了考虑传统演化算法中解的编码和解码方式、种群的演化机制、演化算子的设计和算法的终止条件等问题外，还需要解决譬如问题分解、子种群间的约束和协同关系、子种群中个体适应度的计算方法等问题，具体方法参考文献[6]。

1.3 新算法步骤和特点

新算法步骤如下：

初始化子种群集合{ $P(i)|1\le i\le N$ }，N 为子种群数量;

计算各子种群的初始适应度;

当终止条件没有满足时

Begin

对于每个子种群 $P(i)$

Begin

利用改进郭涛算法进行演化操作;

对于每一个体 $X(i,j)\in P(i)$,

根据适应值评估方法，从其他子种群提供的合作个体计算适应值;

End;

End;

CCEA 的一个最显著的特点是个体适应值与其他种群的个体相关，通常包括多个相关的进化种群，其个体适应值的是由该个体与从其他子种群提供的合作个体组成完整解，计算得到的。本程序针对测试函数设计了特殊的适应值评估方法：每一代的个体与从其他子种群中随机选择的个体组成完整解，计算得到一个适应值；与从其他子种群中选择适应值最大的个体组成完整解，计算得到另一个适应值；这 2 个适应值中较好的一个被选作为该个体的适应值。这一改进使得算法能够更快地找到最优解。

综合改进郭涛算法与 CCEA 的特点，该混合算法兼具协同演化算法对复杂问题求解的高效性，也具有郭涛算法对函数优化问题求解的精确性优势，实验结果表明对于复杂高维的函数优化问题，该算法是有效的。

2 数值实验

为了测试该算法的有效性，也为了方便与郭涛算法的比较，本文参考了文献[5]中设计的 6 个高度复杂的测试函数，并将结果与其做了对比分析。

$$(1)\ \min f_1(x)=\begin{cases} f_1(x_i)=-\left[x_1\sin(9\pi x_2)+x_2\cos(25\pi x_1)+20\right] \\ D=\{x_i\,|-10\le x_1\le 10,-10\le x_2\le 10\}\end{cases}$$

文献[1]中引用的最小值为

$f(-10, 9.9445695) = -39.944506953367$

笔者找到的最小值为

$\min f(-10, 9.9445691518) =-39.944506987870454$

其优于文献[5]中的最好结果。

$$(2)\ \min f_2(x)=\begin{cases} f_2(x_i)=-\left[21.5+x_1\sin(4\pi x_1)+x_2\sin(20\pi x_2)\right] \\ D=\{x_i\,|-3\le x_1\le 12.1,4.1\le x_2\le 5.8\}\end{cases}$$

文献[4]引用的最小值为

$f(11.625\ 5448, 5.7250441) = -38.8502944790207$

文献[5]引用的最小值为

$f(11.6255446916, 5.7250442533) =-38.850294499272380$

笔者找到的最小值为

$\min f(11.6254861056, 5.7250659970)=-38.850285980145159$

该结果不如文献[4]和文献[5]中的最好结果。

$$(3)\ \min f_3(x)=\begin{cases} f_3(x_i)=-(20+x_1\cos x_2+x_2\sin x_1) \\ D=\{x_i\,|\,0\le x_1\le 10,-10\le x_2\le 0\}\end{cases}$$

文献[5]引用的最小值为

$f(10.000\ 000\ 000\ 0, -6.337\ 616\ 197\ 3)=-33.432\ 987\ 051\ 984\ 455$

找到的最小值为

$\min f(10.000\ 000\ 000\ 0, -6.337\ 612\ 764\ 7)=-33.432987052078047$

优于文献[5]中的最好结果。

$$(4)\ \min f_4(x)=\begin{cases} f_4(x_i)=-\left(|x_1|+|x_2|+|x_1x_2|\right) \\ D=\{x_i\,|-10\le x_1,x_2\le 10\}\end{cases}$$

文献[4]中引用的最小值为

$f(10,10)=f(-10, 10)=f(10, -10) = f(-10, -10)=-119.999999935647$

文献[5]中引用的最小值为

$f(10,10)=f(-10,10)=f(10, -10)=f(-10, -10)=-119.99999999980726$

笔者找到的最小值为

$\min f(10,10)=f(-10,10)=$
$f(10,-10)=f(-10,-10)=-120.000000000000000$

与文献[4]和文献[5]中结果相当。

$$(5)\ \min f_5(x)=\begin{cases} f_5(x_i)=-\left(x_1^2+x_2^2+3x_1x_2\right) \\ D=\{x_i\,|-10\le x_1,x_2\le 10\}\end{cases}$$

文献[4]中引用的最小值为

$f(10,10)=f(-10,-10) = -499.999999995314$

找到的最小值为

$\min f(10,10) =f(-10,-10) =-500.000000000000000$

与文献[4]中结果相当。

$$(6)\ \min f_6(x)=\begin{cases} f_6(x_i)=-\left[x_1\sin x_1+x_1x_2\cos(5\pi x_2)\right] \\ D=\{x_i\,|-10\le x_1,x_2\le 10\}\end{cases}$$

文献[4]引用的最小值为

$f(-8.2359638,-10) = f(8.2359638,10)=-146.301886597396$

笔者发现在点(-8.235 963 8, -10)与点(8.235 963 8, 10)处的正确函数值应为-90.002 016 005 279 2；找到的最小值为

$\min f(10,10)= f(-10,-10) = -94.559788891106308$

该结果优于文献[4]中的最好结果。

实例结果统计、实验结果比较见表 1、表 2。

表 1 实例结果统计

种群规模	搜索空间/MB	程序运行 20 次	平均值	最小值
100	10	$f1$	-39.944 506 06	-39.944 506 987 870 454
100	10	$f2$	-38.850 120 17	-38.850 285 980 145 159
100	10	$f3$	-33.432 987 05	-33.432 987 052 078 047
100	10	$f4$	-120.000 000 00	-120.000 000 000 000 000
100	10	$f5$	-500.000 000 00	-500.000 000 000 000 000
100	10	$f6$	-94.559 788 90	-94.559 788 891 106 308

表 2 实验结果比较

测试函数	文献[1-2]中引用的最好结果	实验所得结果
$f1$	-39.944 506 953 367 120	-39.944 506 987 870 454
$f2$	-38.850 294 499 272 380	-38.850 285 980 145 159
$f3$	-33.432 987 051 984 455	-33.432 987 052 078 047
$f4$	-119.999 999 999 807 260	-120.000 000 000 000 000
$f5$	-499.999 999 995 314 000	-500.000 000 000 000 000
$f6$	-90.002 016 005 279 200	-94.559 788 891 106 308

3 数值实验分析

以上都是复杂的、多峰值高频振荡函数，函数图像场景十分复杂，而且搜索空间都很大，常规方法很难奏效，极易陷入局部最优。数值测试表明：(1)该算法适合于求解复杂函数、多解函数优化问题；(2)算法运行速度快，求解精度极高。

4 结束语

将合作式协同演化方法与改进的郭涛算法相结合，应用于函数优化方面，旨在找出一种更高效的算法来求解复杂高维的函数优化问题。改进有：(1)针对郭涛算法没有变异操作的特点，引入高斯正态分布变异算子；(2)结合协同演化算法在求解复杂问题方面的潜质，利用实数编码进行复杂高维函数优化问题的求解。数值实验结果表明该算法是有效的，所求测试函数的全局最小值也优于文献[4-5]中的最好记录。

(下转第 249 页)