控制与决策
Control and Decision

求解非线性方程组系统的改进差分进化算法

王开， 龚文引

引用本文:
王开, 龚文引. 求解非线性方程组系统的改进差分进化算法[J]. 控制与决策, 2020, 35(9): 2121−2128.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2018.1739

求解非线性方程组系统的改进差分进化算法

王 开, 龚文引†

(中国地质大学 计算机学院，武汉 430074)

摘 要: 针对基于邻域拥挤的差分进化算法求解非线性方程组系统时存在丢根、陷入局部最优等不足,提出一种改进的差分进化算法. 首先,提出一种个体预判机制,判断当前群体的个体属于哪一类,并分别采取不同的操作;其次,设计一种新的混合差分变异算子,以增强算法跳出局部最优的能力;然后,改进外部存档策略,延长了父代优秀个体在种群的保存时间,有利于搜索该优秀个体附近的根. 在所选测试函数集上的实验结果表明,所提出的算法能有效搜索到非线性方程组系统的多个根,并与当前5种算法进行对比,所提出算法在找根率和成功率上更具优越性.

关键词: 非线性方程组系统；差分进化算法；个体预判；差分变异

中图分类号: TP273 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2018.1739

引用格式: 王开, 龚文引. 求解非线性方程组系统的改进差分进化算法[J]. 控制与决策, 2020, 35(9): 2121-2128.

Solving nonlinear equations system with an improved differential evolution

WANG Kai, GONG Wen-yin†

(School of Computer Science，China University of Geosciences，Wuhan 430074，China)

Abstract: In order to remedy the drawbacks of neighborhood-based crowding differential evolution for losing the roots and trapping into the local optima when solving nonlinear equations systems (NESs), this paper presents an improved differential evolution, which can be featured as follows: 1) An individual pre-judgement mechanish is proposed, which is used to divide the individuals into different groups, and different operations are used for different groups. 2) An improved hybrid differential mutation is developed to make the algorithm escape the local optima. 3) An improved archive strategy is presented to enhance the algorithm to find more roots. The experimental results on the selected test functions show that the proposed method can locate mutiple roots of the NES efficiently. Compared with other state-of-the-art methods, the propsoed method obtains better results in terms of both the root rate and the success rate.

Keywords: nonlinear equations system；differential evolution；individual pre-judgment；differential mutation

0 引 言

非线性方程组系统(nonlinear equations system, NES)在数学、物理、工程等多个领域广泛存在. NES问题往往都比较复杂,并且很多实际问题都有多个根. 传统数学方法比如牛顿法、拟牛顿法、区间估计等,十分依赖初值的选取;另外,对于某些导数不存在或是导数难求的方程,传统数学方法例如牛顿改进法[1]也具有一定的局限性. 与传统数学方法相比,由于进化算法(evolutionary algorithms, EAs)作为一种全局优化算法,具有自学习、自适应的特性,能够有效地解决传统数学方法难以解决的复杂问题. 除此之外,进化算法对优化函数的特点不敏感,如非凸性、不连续性. 因此,近年来进化算法被用于非线性方程组系统的求解,并获得了良好的结果.

在利用进化算法求解非线性方程组系统时需要考虑以下两个重要问题.

1) NES与优化问题转换, 即如何将非线性方程组系统转换为一种优化问题. Song等[2]将转换技术分为单目标优化技术[3-5]、约束优化技术[6]和多目标优化技术[2,7-8]3种.

2) 优化算法的选取,即选择或设计一个优化算法去处理将非线性方程组系统转化后的优化问题. 近年来利用进化算法处理非线性方程组系统受到越来越多的重视,比如遗传算法[6]、粒子群算法[4,9]、差分进化算法[7,10-11]和帝国主义竞争算法[12]等.

Qu等[5]提出一种基于邻域小生境技术的差分进

收稿日期: 2018-12-20；**修回日期:** 2019-01-14.
基金项目: 国家自然科学基金项目(61573324).
责任编委: 陈家伟.
†通讯作者. E-mail: wygong@cug.edu.cn.

化算法(neighborhood-based crowding DE, NCDE)求解多模态问题,该方法将原始种群划分为多个子种群,从而扩大搜索范围和保持种群多样性,实验表明了该算法在求解多模态问题时的有效性. 与多模态问题类似,NES多根求解问题也是需要算法同时找出多个根. 然而,将NCDE直接用于求解非线性方程组系统存在一些不足,例如算法容易陷入局部最优,算法存在丢根的可能性等. 基于此考虑,本文提出一种改进的差分进化算法同时求解NES多个根. 实验结果表明,所提出的算法能够有效提高求非线性方程组的找根率和成功率.

1 问题提出及转化

一般来说,NES由m个方程组成,其中至少包含一个非线性方程. $x=(x_1,\dots,x_n)^{\mathrm{T}}\in X$是一个$n$维向量,$X$是该NES的搜索空间. NES的数学形式可以表示为

$$\begin{cases} f_1(x_1,\dots,x_n)=0,\\ \quad\vdots\\ f_m(x_1,\dots,x_n)=0. \end{cases} \tag{1}$$

在对NES求根的过程中,本文将其转换为如下所示的最小化单目标优化问题

$$\min\sum_{i=1}^{m}f_i^2(x). \tag{2}$$

2 标准差分进化算法

差分进化算法(DE)[10-11]是一种基于群体全局启发式搜索的算法,由于其具有良好的鲁棒性、自学习性、自适应性、收敛速度快、操作简单等特性而被广泛使用. 该算法已经在多个领域取得了成功应用,如电力系统经济负荷分配[13]、作业车间调度[14]、生物学、经济学等[15].

DE算法的基本流程如下.

step 1: 初始化. 根据定义域随机产生NP个个体作为初始种群,并计算它们的适应值.

step 2: 变异. 利用差分变异算子产生实验向量.

step 3: 交叉. 将实验向量和基向量按一定的概率进行杂交,产生子代.

step 4: 选择. 比较子代与父代的适应值,如果子代的适应值优于父代的适应值,则替换,否则保留父代.

step 5: 当前一代已经迭代完成,判断是否达到循环终止条件,若是则退出,否则代数加1,继续执行step 2到step 5.

3 基于邻域拥挤的差分进化算法

基于邻域拥挤的差分进化算法(neighborhood-based crowding DE, NCDE)是2012年Qu等[5]提出的一种基于邻域小生境技术的差分进化算法,用于求解多模态问题. 其主要思想是将种群分割成各个小种群,在邻域范围内变异交叉,从而有效地保持种群多样性. 其主要的算法步骤如下.

step 1: 初始化. 根据定义域随机产生NP个个体作为初始种群,并计算它们的适应值.

step 2: 从$i=1$ to NP.

step2.1: 组成子群. 选择离当前个体i最近的m个个体组成子种群sub_i.

step2.2: 产生子代个体. DE处理sub_i产生子代个体u_i.

step 2.3: 选择. 比较u_i与欧氏距离最近的父代的函数适应值,适者生存.

step 3: 当前一代已经迭代完成,判断是否达到循环终止条件,若是则退出,否则代数加1,继续step 2.

在迭代过程中,对于每一个个体,NCDE是先找其最近的部分个体形成子群,然后利用差分进化算法产生子代,选择替换,通过逐代进化,直到达到停止条件. 然而,针对NES问题如果目标函数(2)中存在局部最优的情况,则NCDE容易陷入局部最优. 如图1所示,“*”是根(最优解),“★”是局部最优,“○”是当前种群的个体分布. 可以看出,种群中有很多个体分布在中间局部最优的附近,与多模态优化问题不同,NES根的求解应该尽可能避免陷入局部最优解,这样就浪费了很多计算资源. 因此,如何在算法里检测局部最优并进行有效处理具有重要的意义.

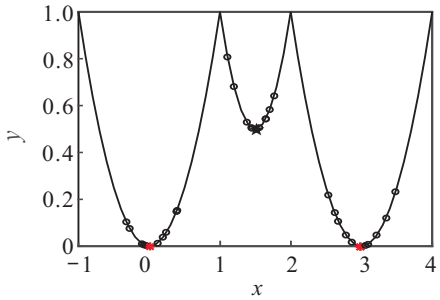

图1 NCDE搜索过程中陷入局部最优

此外,NCDE还存在另一个问题. 如图2所示,“*”是NES问题的根,而“○”是算法已找到的根,当两个根靠得非常近时,NCDE有时只能找到其中一个根,而丢失了与其相近的另一个根. 究其原因,NCDE在子代与父代进行替换时,根据子代个体选择距离其最近的父代个体比较,此时如果子代个体和父代个体

是两个不同的根,而且子代的适应值优于其最近的父代个体,则父代个体就会被替代,即丢失了一个根,从而导致如图2所示的情况.

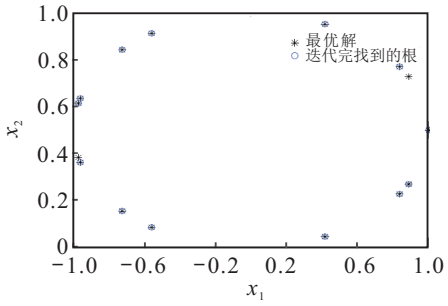

图 2　CDE丢失相邻根

4　改进的差分进化算法

为了弥补上述NCDE的缺陷,提出一种改进的差分进化算法,简称为CADE(counter-archive aided NCDE),该算法在NCDE基础上加入个体预判断机制、混合算子机制和改进外部存档机制. 通过个体预判断机制可以让算法自行掌握种群每一代个体的进化程度,混合算子机制一方面通过局部收敛达到全局收敛的效果,另一方面避免陷入局部最优的情况. 而外部存档机制用来保存种群里已经达到方程组根的精度的个体,避免浪费计算资源. 下面详细介绍3种机制.

4.1　个体预判机制

受文献[16]启发,将种群的每个个体与一个计数器相关联,其最大值为T,记录当前个体未被子代个体替换的次数. 然后,通过该计数器判断个体是属于哪一类,即普通个体,陷入局部最优的个体,候选根的个体.

1) 如果个体的计数器未达到最大值T,即认为其是普通个体,则与原始NCDE算法一样处理.

2) 如果是个体的计数器达到最大值T,但是未达到候选根的精度,此时有两种情况如下:

①由NCDE可知,如图3所示“*”为方程组的根,“○”为初始化之后相对比较靠近的根,因此具有较好的适应值,被子代替换的可能性较小,所以该个体的计数器会迅速增加到最大值T,此种情况在算法执行前期出现的概率较大.

②随着迭代的进行,种群逐渐收敛,如果后期出现该个体未被子代个体替换的次数达到计数器最大值T,则其有很大概率成为已经陷入局部最优的个体,故此情况在算法执行后期出现的概率大.

在条件2)下,因为算法前期FES/MaxFES(其中

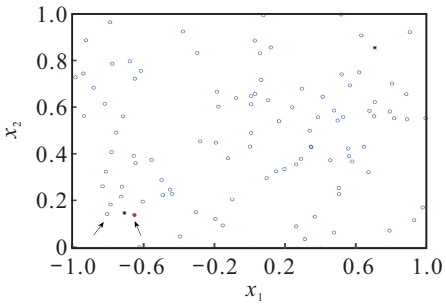

图 3　初试群体个体分布图

FES为当前适应值评价次数,MaxFES为初始给定的最大适应值评价次数)比较小接近0,但随着种群进化,该比值会一直增加,直至到1. 所以可将FES/MaxFES作为①和②概率事件的判断标准. 本文在[0, 1]内产生一个随机数rand,将满足rand > FES/MaxFES条件时期的个体认为是初始适应值较好的个体,并当作普通个体处理. 相反,将满足rand ⩽ FES/MaxFES条件时期的个体认为是陷入局部最优的个体,该类个体会削弱算法的性能,特别在求解非线性方程组的多根问题上表现尤为明显. 因此,如何处理这种个体,也是一个急需解决的问题. 最简单的方法是直接重新初始化,但是NCDE采用的是邻域变异,因此很有可能再次陷入局部最优,故简单的初始化并不能很好地解决问题,并且迭代过程又消耗了部分计算资源. 如图4所示,为了解决这个问题,通过选择与该个体距离最远的个体组成子种群,然后通过交叉变异使个体跳出局部最优.

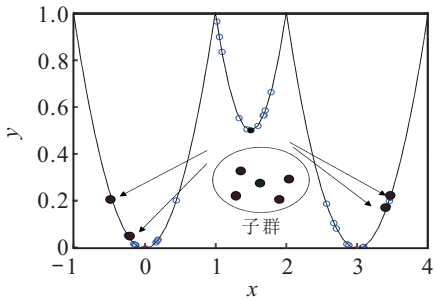

图 4　局部最优处理办法

3) 如果是个体的计数器达到最大值T,也达到候选根的精度,即认为其是候选根,则将该个体放入外部存档,个体重新初始化,此时该位置的个体就是普通个体,与NCDE一样处理,并将该个体的计数器重置为0,这样就避免了候选根浪费计算资源的情况.

4.2　混合算子

通过个体预判机制反馈的信息,对普通个体选取最近的m个体形成子群,采用的变异算子为

$$u_i = x_{r_1} + F(x_{r_2} - x_{r_3}). \tag{3}$$

对陷入局部最优的个体x_i选取最远的m个个体形成子群,且采用的变异算子为

$$u_i = x_i + F(x_{r_1} - x_{r_2} + x_{r_3} - x_{r_4}). \tag{4}$$

子群大小m是与NCDE一样的设置方式,大小跟种群大小相关,并随着迭代过程从大变小;

这样可以使陷入局部最优的个体,从自己的邻域外搜索,从而跳出局部最优的困境,节省了计算资源. 以上的r_1, r_2, r_3, r_4为按照NCDE方法从子种群里随机选出且互不相同的个体.

4.3 外部存档

采用类似文献[17]根外部存档策略,如果个体x是一个根,则它可以保存在外部存档中的概率很大,否则它将被舍弃. 与文献[17]不同的是,本文对其进行改进:首先判断当前个体的计数器的值是否达到最大值T,然后判断当前个体是否可以放入外部存档. 因为NCDE是邻域搜索,若方程组的两个根A和B非常靠近,则将B放入外部存档,会导致找到A的过程变得相当困难,而通过计数器来判断可以让B在原始种群多存留一段时间,从而有利于辅助找到其邻域根A. 改进的外部存档步骤如下.

1) 个体预判断阶段:如果个体的计数器达到最大值T且个体达到候选根的精度,则将个体放入外部存档中.

2) 选择阶段:

① 如果子代个体比父代个体适应值更好,且两者是方程组的两个互不相同的根时,为了保证方程组的根不丢失,将父代个体放入外部存档;子代个体替换父代个体,计数器重置为$\text{counter} = 0$.

②如果父代个体比子代个体适应值更好,且两者是方程组的两个互不相同的根时,则将子代个体放入外部存档,计数器counter加1.

4.4 算法详细步骤

CADE算法具体步骤如下.

step 1: 初始化. 随机产生NP个个体作为初始种群,并进行适应值评价,每个个体关联一个计数器$\text{counter} = 0$.

step 2: 从$i = 1$ to NP.

step 2.1: 个体预判机制. 根据个体关联的计数器和预先给定的候选根的精度判断个体属于何种类型,并作相应的处理.

step 2,2: 组成子群. 根据预判断机制反馈的信息,从原始种群选择m个个体形成子种群sub_i.

step 2.3: 变异. 根据预判断机制判断父代个体x_i是否陷入局部最优,并据此选择不同变异算子(对普通个体所在子群采用的变异算子为式(3),陷入局部最优个体所在子群采用的变异算子为式(4)产生实验向量).

step 2.4: 交叉. 将实验向量和基向量按一定的概率交叉一部分属性值,产生子代u_i.

step 2.5: 选择. 比较u_i与欧氏距离最近的父代的函数适应值时,也要考虑两者是否是候选根,是否是相同的候选根,如果是候选根,则考虑放入外部存档等情况,并更新个体的计数器,避免迭代中子代父代个体是根却被丢弃的情况.

step 3: 判断是否达到循环终止条件,若是则退出,否则进化代数加1,继续循环.

5 实验与结果

5.1 测试函数描述

为了验证CADE的有效性,本次测试集采用文献[17]中高维或含有较多根的8个方程组,这里表示为F1 ~ F8.

这8个非线性方程组系统如下:

$$\text{F1}\begin{cases}\cos(2x_1) - \cos(2x_2) - 0.4 = 0;\\ 2(x_2 - x_1) + \sin(2x_2) - \sin(2x_1) - 1.2 = 0.\end{cases} \tag{5}$$

其中:$x_i \in [-10, 10], i = 1, 2$,有13个根如表1所示.

表 1 F1的根

x_1	x_2
−9.268 258	−8.931 402
−8.744 542	−7.164 787
−6.126 665	−5.789 809
−5.602 950	−4.023 195
−2.985 073	−2.648 216
−2.461 357	−0.881 602
0.156 520	0.493 376
0.680 236	2.259 991
3.298 113	3.634 969
3.821 828	5.401 583
6.439 705	6.776 562
6.963 421	8.543 176
9.581 298	9.918 154

$$\text{F2}\begin{cases}x_1 - 0.25 = 0,\\ x_1\sin(4\pi x_2^2) + 0.75x_1 - 0.25 = 0.\end{cases} \tag{6}$$

其中: $x_i \in [-1,1], i=1,2$, 有8个根如表2所示.

表 2　F2的根

x_1	x_2
0.250 000	−0.854 337
0.250 000	−0.721 185
0.250 000	−0.479 471
0.250 000	−0.141 801
0.250 000	0.141 801
0.250 000	0.479 471
0.250 000	0.721 185
0.250 000	0.854 337

$$\mathrm{F3}\begin{cases}(1-R)\Big[\Big(\dfrac{D}{10(1+\beta_1)}-x_1\Big)\mathrm{e}^{\frac{10x_1}{1+\frac{10x_1}{\gamma}}}\Big]-x_1=0,\\ (1-R)\Big[\Big(\dfrac{D}{10}-\beta_1x_1-(1+\beta_2)x_2\Big)\mathrm{e}^{\frac{10x_1}{1+\frac{10x_1}{\gamma}}}\Big]+\\ x_1-(1+\beta_2)x_2=0.\end{cases}\tag{7}$$

其中: $x_i \in [0,1], i=1,2, R=0.96, D=22, \gamma=1\,000$, 且 $\beta_1=\beta_2=2$, 有7个根如表3所示.

表 3　F3的根

x_1	x_2
0.042 100	0.061 813
0.042 100	0.268 723
0.266 600	0.178 430
0.266 600	0.327 267
0.266 600	0.461 111
0.042 318	0.686 779
0.719 074	0.244 197

$$\mathrm{F4}\begin{cases}\sin(x_1^3)-3x_1x_2^2-1=0,\\ \cos(3x_1^2x_2)-|x_2^3|+1=0.\end{cases}\tag{8}$$

其中: $x_i \in [-2,2], i=1,2$, 有10个根如表4所示.

表 4　F4的根

x_1	x_2
−1.810 885	−0.349 092
−1.810 885	0.349 092
−1.502 221	−0.409 077
−1.502 221	0.409 077
−1.791 302	0.301 926
−1.791 302	−0.301 926
−0.947 268	0.785 020
−0.947 268	−0.785 020
−0.213 057	1.256 845
−0.213 057	−1.256 845

$$\mathrm{F5}\begin{cases}4x_1^3+4x_1x_2+2x_2^2-42x_1-14=0,\\ 4x_2^3+2x_1^2+4x_1x_2-26x_2-22=0.\end{cases}\tag{9}$$

其中: $x_i \in [-5,5], i=1,2$, 有9个根如表5所示.

表 5　F5的根

x_1	x_2
−0.127 961	−1.953 715
−0.270 845	−0.923 039
0.086 678	2.884 255
3.385 154	0.073 852
3.584 428	−1.848 127
3.000 000	2.000 000
−3.779 310	−3.283 186
−3.073 026	−0.081 353
−2.805 118	3.131 313

$$\mathrm{F6}\begin{cases}-\sin(x_1)\cos(x_2)-2\cos(x_1)\sin(x_2)=0,\\ -\cos(x_1)\sin(x_2)-2\sin(x_1)\cos(x_2)=0.\end{cases}\tag{10}$$

其中: $x_i \in [0,2\pi], i=1,2$, 有13个根如表6所示.

表 6　F6的根

x_1	x_2
0.000 000	0.000 000
3.141 593	0.000 000
1.570 796	1.570 796
6.283 185	0.000 000
0.000 000	3.141 593
4.712 389	1.570 796
3.141 593	3.141 593
1.570 796	4.712 389
6.283 185	3.141 593
0.000 000	6.283 185
4.712 389	4.712 389
3.141 593	6.283 185
6.283 185	6.283 185

$$\mathrm{F7}\begin{cases}\beta_{11}+\beta_{12}x_2^2+\beta_{13}x_3^2+\beta_{14}x_2x_3+\beta_{15}x_2^3x_3^2=0,\\ \beta_{21}+\beta_{22}x_3^2+\beta_{23}x_1^2+\beta_{24}x_3x_1+\beta_{25}x_3^3x_1^2=0,\\ \beta_{31}+\beta_{32}x_1^2+\beta_{33}x_2^2+\beta_{34}x_1x_2+\beta_{35}x_1^3x_2^2=0;\end{cases}\tag{11}$$

$$\beta_{ij}=\begin{bmatrix}-13,-1,-1,24,-1\\ -13,-1,-1,24,-1\\ -13,-1,-1,24,-1\end{bmatrix}.$$

其中: $x_i \in [-20,20], i=1,2,3$, 有16个根如表7所示.

表 7 F7的根

x_1	x_2	x_3
10.857 703 600 0	0.779 548 044 9	0.779 548 045 1
−10.857 703 600 0	−0.779 548 044 9	−0.779 548 045 1
0.332 073 098 4	4.625 181 601 0	4.625 181 601 0
−0.332 073 098 4	−4.625 181 601 0	−4.625 181 601 0
0.779 548 044 9	0.779 548 044 9	0.779 548 045 1
−0.779 548 044 9	−0.779 548 044 9	−0.779 548 045 1
0.779 548 044 9	10.857 703 600 0	0.779 548 045 1
−0.779 548 044 9	−10.857 703 600 0	−0.779 548 045 1
0.779 548 044 0	0.779 548 045 7	10.857 703 600 0
−0.779 548 044 0	−0.779 548 045 7	−10.857 703 600 0
4.625 181 601 0	4.625 181 601 0	4.625 181 601 0
−4.625 181 601 0	−4.625 181 601 0	−4.625 181 601 0
4.625 181 601 0	0.332 073 098 4	4.625 181 601 0
−4.625 181 601 0	−0.332 073 098 4	−4.625 181 601 0
4.625 181 600 0	4.625 181 613 0	0.332 073 098 4
−4.625 181 600 0	−4.625 181 613 0	−0.332 073 098 4

$$\mathrm{F8}\begin{cases}3u^2+2v-3x_1+x_1x_2+x_3^2-24=0,\\ u-3v^2-x_1+2x_2-x_1x_3+10=0,\\ 2u-v+x_1-x_2^2+2x_3-5=0.\end{cases}\tag{12}$$

其中: $u=3x_1+x_2-x_3, v=x_1^2-x_2+x_3$, 且 $x_i\in[-3,3], i=1,2,3$. 它有两个解 (1.140 226, −0.448 382, 0.135 274) 和 (1, 2, 3).

5.2 评价指标

为了对比不同算法的性能,采用文献[17]中的两个性能指标: 找根率(root ratio, RR)和成功率(success rate, SR)来判断.

RR的计算公式如下:

$$\mathrm{RR}=\frac{\sum\limits_{i=1}^{N_r}\mathrm{NOR}_i}{\mathrm{NOR}\cdot N_r}.\tag{13}$$

其中: N_r 是算法独立运行的总次数, NOR_i 为当前第 i 次独立运行找到根的个数, NOR为当前NES已知根的个数.

SR的计算公式如下:

$$\mathrm{SR}=\frac{N_{sr}}{N_r},\tag{14}$$

其中 N_{sr} 为成功找到NES所有根的次数.

5.3 参数设置

1) 候选根精度: 如果方程组的维数 $n\leqslant 5$, 则 $\theta=10^{-6}$; 否则, $\theta=10^{-4}$.

2) 距离半径(保证两候选根不要靠得太近,即是不同的根): 如果方程组的维数 $n\leqslant 5$, 则 $\delta=10^{-3}$; 否则, $\delta=10^{-2}$.

3) 外部存档容纳最大个体数 $\max_{s_a}=\mathrm{NP}$, 对所有问题,算法独立运行30次.

由于计数器的最大值 T 是一个参数,讨论了当 $T=5,10,15,20$ 时对CADE在找根率(RR)和成功求得全部根的成功率(SR). 如表8所示, $T=5$ 与 $T=10$ 相比,二者的找根率差别不大,但是求全部根的成功率上 $T=10$ 更好. 从 $T=10$ 到 $T=20$ 算法在找根率和成功率上呈现依次下降的趋势,因此 $T=10$ 时CADE算法总体性能最好.

表 8 计数器 T 的不同取值对CADE的影响

	$T=5$		$T=10$		$T=15$		$T=20$	
	RR	SR	RR	SR	RR	SR	RR	SR
F1	**0.951 3**	**0.500 0**	0.933 3	0.466 7	0.897 4	0.300 0	0.800 0	0.00 00
F2	0.995 8	0.966 7	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	0.987 5	0.900 0
F3	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	0.990 5	0.933 3	0.985 7	0.900 0
F4	**0.740 0**	**0.066 7**	0.703 3	0.066 7	0.686 7	0.066 7	0.650 0	0.000 0
F5	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	0.996 3	0.966 7
F6	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	0.994 9	0.933 3
F7	0.925 0	0.266 7	0.939 6	0.400 0	**0.966 7**	**0.566 7**	0.962 5	0.566 7
F8	0.950 0	0.900 0	0.950 0	0.900 0	0.966 7	0.9333	**1.000 0**	**1.000 0**
均值	**0.945 3**	0.712 5	0.940 8	**0.729 2**	0.938 5	0.725 0	0.922 1	0.658 3

5.4 算法对比

为了验证CADE算法的有效性,将其5种具有较好性能的算法进行比较,对比算法如下.

1) NCDE: 该算法主要是基于邻域小生境技术求解多模态问题,也可转换为求解非线性方程组的根.

2) RADE[17]: 该算法主要用带有排斥法的外部存档和自适应的参数法求解非线性方程组的根.

3) MONES[2]: 该算法主要用多目标进化算法求解非线性方程组的根.

4) A-Web[8]: 该算法主要用加权多目标算法求解非线性方程组的根.

5) DR-JADE[18]: 该算法主要用动态半径的外部

存档和自适应参数求解非线性方程组的根.

对于上述6种算法,参数设置如表9所示. 为了对比不同算法的结果,使用相同的初始种群、最大适应值评价次数去评估不同的算法,并且每种算法独立运行30次,最后结果取平均值. 所有算法都在Matlab 2016b中实现,所有对比实验都是在Windows 10 64位操作系统下使用Intel Core i5-4590M处理器 @ 3.30 GHz, 8 GB RAM的台式PC上执行.

表 9　参数设置

算法	参数设置
CADE	$\mathrm{NP}=100, F=0.5, \mathrm{CR}=0.9, T=10$
NCDE	$\mathrm{NP}=100, F=0.5, \mathrm{CR}=0.9$
RADE	$\mathrm{NP}=100, H_m=200$
MONES	$\mathrm{NP}=100, H_m=200$
A-Web	$\mathrm{NP}=100, H_m=200$
DREA	$\mathrm{NP}=10, c=0.1, \mu_{\mathrm{CR}}=0.9, \mu_{\mathrm{CR}}=0.1$

5.5　结果对比与分析

为了将算法性能的优越性更直观地显示出来,将CADE算法与NCDE、RADE、MONES、A-Web、DR-JADE算法在找根率(RR)上进行比较,如表10所示. 由表10可以看出,CADE可以获得最大的平均RR值,且在F2、F3、F5、F6、F7这5个问题上可以获得最优RR值. 本文提出的CADE算法充分研究了算法NCDE存在丢根、陷入局部最优等不足,分别加入了个体预判机制,新的混合差分变异算子,改进的外部存档策略,所以提高了算法求根效果. 从实验结果来看,DR-JADE排名第2, RADE排名第3, NCDE排名第4, MONES由于较依赖方程组第1个自变量而导致求根的效果并不好. 所提出的CADE算法比其余5种算法在搜索方程组的根上性能更具优势.

为了避免单独比较找根率造成的偏差,再与5种算法在求解方程组全部根的成功率(SR)上进行了比较,结果如表11所示,CADE算法在成功率远高于其他5种对比的算法.

综上,实验结果表明CADE算法在求解非线性方程组问题上具有良好的效果.

表 10　不同算法的root ratio对比

算法	F1	F2	F3	F4	F5	F6	F7	F8	均值
CADE	0.933 3	**1.000 0**	**1.000 0**	0.703 3	**1.000 0**	**1.000 0**	**0.939 6**	0.950 0	**0.940 8**
DR-JADE	0.726 7	**1.000 0**	**1.000 0**	**1.000 0**	0.770 0	**1.000 0**	0.872 9	**1.000 0**	0.921 2
RADE	0.901 5	0.990 0	0.997 1	0.631 0	0.986 7	0.995 4	0.561 9	0.835 0	0.862 3
NCDE	0.756 4	0.970 8	0.961 9	0.590 0	0.992 6	0.912 8	0.539 6	0.933 3	0.832 2
A-Web	**1.000 0**	0.940 0	0.837 1	0.888 0	0.973 3	1.000 0	0.003 8	0.030 0	0.709 0
MONES	0.934 6	0.125 0	0.438 6	0.436 0	0.990 0	0.436 9	0.146 3	0.315 0	0.477 8

表 11　不同算法的success rate对比

算法	F1	F2	F3	F4	F5	F6	F7	F8	均值
CADE	0.466 7	**1.000 0**	**1.000 0**	0.066 7	**1.000 0**	**1.000 0**	**0.400 0**	0.900 0	**0.729 2**
DR-JADE	0.000 0	**1.000 0**	**1.000 0**	**1.000 0**	0.000 0	**1.000 0**	0.000 0	**1.000 0**	0.625 0
RADE	0.310 0	0.930 0	0.980 0	0.000 0	0.890 0	0.940 0	0.000 0	0.670 0	0.590 0
NCDE	0.000 0	0.766 667	0.833 3	0.000 0	0.933 3	0.266 7	0.000 0	0.866 7	0.458 3
A-Web	**1.000 0**	0.600 0	0.120 0	0.120 0	0.760 0	**1.000 0**	0.000 0	0.020 0	0.452 5
MONES	0.670 0	0.000 0	0.000 0	0.000 0	0.920 0	0.000 0	0.000 0	0.070 0	0.207 5

6　结　论

本文主要介绍了非线性方程组的求解,首先将非线性方程组转换为一种单目标优化问题,然后根据已经提出的NCDE算法的问题,提出了一种改进的差分进化算法. 该算法的主要改进是考虑了NCDE陷入局部最优造成浪费计算资源以及迭代过程存在丢根的情况,然后相应地进行了处理. 所提出的改进算法一方面可以增加找到根的多样性,另一方面增加了算法的稳定性. 为了验证CADE算法的有效性,对8个测试非线性方程组进行了独立运行30次的实验, 与当前已经提出的5种算法得出的结果作了比较,进一步验证了所提出算法的优越性. 下一步将研究参数自适应[19]、多算子协同进化[20]、多目标相关技术[21-22]等,以进一步提高算法性能,使之能运用到复杂的科学工程问题中.

参考文献(References)

[1] 吕巍, 魏良亭, 冯恩民. 一类求解非线性奇异方程组的牛顿改进算法[J]. 控制与决策, 2017, 32(12): 2240-2246.
(Lv W, Wei L T, Feng E N. A modification of Newton's method solving non-linear equations with singular Jacobian[J]. Control and Decision, 2017, 32(12): 2240-2246.)

[2] Song W, Wang Y, Li H, et al. Locating multiple optimal solutions of nonlinear equation systems based on multiobjective optimization[J]. IEEE Transactions on Evolutionary Computation, 2015, 19(3): 414-431.

[3] Oliveira H A Jr, Petraglia A. Solving nonlinear systems of functional equations with fuzzy adaptive simulated annealing[J]. Applied Soft Computing, 2013, 13(11): 4349-4357.

[4] Jaberipour M, Khorram E, Karimi B. Particle swarm algorithm for solving systems of nonlinear equations[J]. Computers & Mathematics with Applications, 2011, 62(2): 566-576.

[5] Qu B, Suganthan P, Liang J. Differential evolution with neighborhood mutation for multimodal optimization[J]. IEEE Transactions on Evolutionary Computation, 2012, 16(5): 601-614.

[6] Pourrajabian A, Ebrahimi R, Mirzaei M, et al. Applying genetic algorithms for solving nonlinear algebraic equations[J]. Applied Mathematics and Computation, 2013, 219(24): 11483-11494.

[7] 许伟伟, 梁静, 岳彩通, 等. 多模态多目标差分进化算法求解非线性方程组[J]. 计算机应用研究, 2019, DOI: 10.3969/j.issn.1001-3695.2017.11.0734.
(Xu W W, Liang J, Yue C T, et al. Multimodal multi-objective differential evolution algorithm for solving nonlinear equations[J]. Application Research of Computers, 2019, DOI: 10.3969/j.issn.1001-3695.2017.11.0734.)

[8] Gong W, Wang Y, Cai Z, et al. A weighted biobjective transformation technique for locating multiple optimal solutions of nonlinear equation systems[J]. IEEE Transactions on Evolutionary Computation, 2017, 21(5): 697-713.

[9] Mo Y, Liu H, Wang Q. Conjugate direction particle swarm optimization solving systems of nonlinear equations[J]. Computers & Mathematics with Applications, 2009, 57(11/12): 1877-1882.

[10] 刘波, 王凌, 金以慧. 差分进化算法研究进展[J]. 控制与决策, 2007, 22(7): 721-729.
(Liu B, Wang L, Jin Y H. Adcances in differential evolution[J]. Control and Decision, 2007, 22(7): 721-729.)

[11] Storn R, Price K. Differential evolution — A simple and efficient heuristic for global optimization over continuous spaces[J]. Journal of Global Optimization, 1997, 11(4): 341-359.

[12] Abdollahi M, Isazadeh A, Abdollahi D. Imperialist competitive algorithm for solving systems of nonlinear equations[J]. Computers & Mathematics with Applications, 2013, 65(12): 1894-1908.

[13] 王凌, 黄付卓, 李灵坡. 基于混合双种群差分进化的电力系统经济负荷分配[J]. 控制与决策, 2009, 24(8): 1156-1166.
(Wang L, Huang F Z, Li L P. Economic distribution of power systems based on hybrid differential evolution with double populations[J]. Control and Decision, 2009, 24(8): 1156-1166.)

[14] 潘全科, 王凌, 高亮, 等. 基于差分进化与块结构邻域的作业车间调度优化[J]. 机械工程学报, 2010, 46(22): 182-188.
(Pan Q K, Wang L, Gao L, et al. Differential evolution algorithm based on blocks on critical path for job shop scheduling problems[J]. Journal of Mechanicial Energing, 2010, 46(22): 182-188.)

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

[16] Wang Z, Zhan Z H, Lin Y, et al. Dual-strategy differential evolution with affinity propagation clustering for multimodal optimization problems[J]. IEEE Transactions on Evolutionary Computation, 2018, 22(6): 894-908.

[17] Gong W, Wang Y, Cai Z, et al. Finding multiple roots of nonlinear equation systems via a repulsion-based adaptive differential evolution[J]. IEEE Transactions on Systems, Man, and Cybernetics Systems, 2018, DOI: 10.1109/TSMC.2018.2828018.

[18] Liao Z W, Gong W S, Yan X, et al. Solving nonlinear equations system with dynamic repulsion-based evolutionary algorithms[J]. IEEE Transactions on Systems, Man, and Cybernetics Systems, 2018, DOI: 10.1109/TSMC.2018.2852798.

[19] Zhang J, Sanderson A. JADE: Adaptive differential evolution with optional external archive[J]. IEEE Transactions on Evolutionary Computation, 2009, 13(5): 945-958.

[20] 王凌, 沈婧楠, 王圣尧, 等. 协同进化算法研究进展[J]. 控制与决策, 2015, 30(2): 193-202.
(Wang L, Shen J N, Wang S Y, et al. Advances in co-evolutionary algorithms[J]. Control and Decision, 2015, 30(2): 193-202.)

[21] 孙靖, 巩敦卫, 季新芳. 基于偏好方向的区间多目标交互进化算法[J]. 控制与决策, 2013, 28(4): 542-546.
(Sun J, Gong D W, Ji X F. Interactive evolutionary algorithms for interval multi-objective optimization problems based on preference direction[J]. Control and Decision, 2013, 28(4): 542-546.)

[22] 巩敦卫, 季新芳, 孙晓燕. 基于集合的高维多目标优化问题的进化算法[J]. 电子学报, 2014, 42(1): 77-83.
(Gong D W, Ji X F, Sun X Y. Solving many-Objective optimization problems using set-based evolutionary algorithms[J]. Acta Electronica Sinica, 2014, 42(1): 77-83.)

作者简介

王开 (1996–), 男, 硕士生, 从事基于智能优化方法的非线性方程组求解的研究, E-mail: 2489009174@qq.com;

龚文引 (1979–), 男, 教授, 博士生导师, 从事智能优化方法及其相关应用等研究, E-mail: wygong@cug.edu.cn.

(责任编辑：孙艺红)